Volume 40 Number 1 March 2016 ISSN 0350-5596

# Informatica

An International Journal of Computing and Informatics

Special Issue:

**Engineering and Applications of Software Agents**

Guest Editors:

**Amelia Bădică**

**Zoran Budimac**

1977

# Editorial Boards

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

# Editors' Introduction to the Special Issue on "Engineering and Applications of Software Agents"

The aim of this Special Issue is to introduce to the readers a selection of papers from the *3rd Workshop on Applications of Software Agents – WASA 2013* in the area of agent-based modelling and simulation. WASA'2013 was held in Sinaia, Romania, during October 11-13, 2013. The WASA 2013 workshop was organized within the framework of the *17th International Conference on System Theory, Control and Computing –ICSTCC'2013*. The aim of the WASA series of workshops is to contribute to the advancement of technologies and applications of software agents' by bridging the gap between the theory and practice of software agents.

15 papers were accepted for presentation at WASA'2013. Among them, 4 papers were selected, after further extension and additional review, for inclusion in this Special Issue on "Engineering and Applications of Software Agents".

The article "AgentPlanner – agent-based timetabling system" by Rafal Tkaczyk, Maria Ganzha, and Marcin Paprzycki is in the area of *applications of software agents*. The authors propose an agent-based timetabling system called AgentPlanner. The system was evaluated using real data set of a department at the University of Gdansk. The results were quite encouraging, i.e. AgentPlanner outperformed the state-of-the-art timetabling software based on the genetic algorithms and it was capable of satisfactorily solving the problem of schedule adjustment.

The article "Jason Interpreter, Enterprise Edition" by Dejan Mitrović, Mirjana Ivanović, Rafael H. Bordini, and Costin Bădică is in the area of *engineering of software agents*. This paper presents a framework that integrates Jason – an interpreter that provides a Java implementation of the AgentSpeak programming language with the Enterprise edition of the Java platform. The contribution of the paper is bridging the gap between the agent technology and modern enterprise solutions for distributed software development.

The article "Expressing GMoDS Models into Object-Oriented Models Using the Event-B Language" by Marius Brezovan, Liana Stanescu and Eugen Ganea is in the area of *engineering of software agents*. This paper proposes a generic framework for expressing the Goal Model for Dynamic Systems (GMoDS) goal-based agent-oriented methodology for the specification of multi-agent systems, using Event-B. The mapping was achieved by adding object-oriented modelling support to Event B using the modularization plug-in of the Rodin framework.

The article "HTML5-based mobile agents for Web-of-Things" by Jari-Pekka Voutilainen, Anna-Liisa Mattila, Kari Systä, and Tommi Mikkonen is in the area of *applications of software agents*. This paper proposes a solution based on mobile agents for operating and managing Internet-connected systems composed of gadgets, sensors and actuators. The solution is supported by two proof-of-concept experiments related to agents for embedded devices interconnected in the Web-of-Things and to the management of agents in the Cloud.

We would like to thank all the reviewers for their restless reviewing effort and valuable feedback and all the authors who submitted their contributions to WASA'2013, as well as to this Special Issue.

*Amelia Bădică,*
*Zoran Budimac*

# AgentPlanner – Agent-based Timetabling System

Rafał Tkaczyk
Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland
IT Systems Department of the Vemco Co. Ltd., Sopot, Poland
E-mail: rafal.tkaczyk88@gmail.com

Maria Ganzha
Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland
Institute of Informatics, University of Gdańsk, Gdańsk, Poland
E-mail: maria.ganzha@ibspan.waw.pl

Marcin Paprzycki
Systems Research Institute of the Polish Academy of Sciences
Warsaw Management Academy, Warsaw, Poland
E-mail: marcin.paprzycki@ibspan.waw.pl

**Keywords:** software agent, multi-agent system, timetabling, negotiations, agent-based timetabling system

**Received:** July 24, 2015

*The aim of the paper is to describe the AgentPlanner, an agent-based timetabling system. After its initial implementation (described in [1]), based on results of experiments, we have modified the design (to eliminate discovered shortcomings). Here, we describe the improved AgentPlanner and compare its performance with the state-of-the-art, Free Timetabling Software (FET).*

*Povzetek: Opisan je AgentPlanner, agentni sistem za urnike.*

## 1 Introduction

Creating a timetable is a challenging problem. On the one hand, timetables are widely used in multiple application areas. On the other, timetabling is an NP-hard problem. As a result, many methods that solve this problem have been proposed (see, section 5).

Recently, we have developed an agent-based timetabling system and reported preliminary experimental results concerning its performance (in, [1]). After the publication, we have been contacted by the developers of the FET program [2]. Discussions that ensued, combined with our own assessment of the shortcomings of the initial version of the AgentPlanner, resulted in improvements in its design. Furthermore, we have made changes in the experimental setup, to make the comparison more fair. Therefore, in the current contribution, we present a completely new set of experimental results.

We proceed as follows. In the next section we summarize the state-of-the-art in timetabling. Next, we outline the reasons that shaped the specific design of our AgentPlanner system. We, then, describe details of its implementation. In the penultimate section, we present the results of performed experiments. Finally, we discuss issues related to flexibility of the AgentPlanner design and possible future research directions / improvements.

## 2 Timetabling – related work

Timetabling is a common problem, which is applied in many domains (e.g. business, industry, science, private applications, etc.). Therefore, many scientists have considered it, and many solution methods have been proposed. Below, we summarize few most common and effective methods for solving the timetabling problem. Let us note, that our work concerns scheduling courses in a "college," and this provides the context for what follows. Furthermore, due to the lack of space, details of described methods are omitted. Interested readers should consult references.

### 2.1 Heuristic methods

#### 2.1.1 Genetic algorithms (GA)

Typical approach, used when applying genetic algorithms to solve the timetabling problem, is as follows. A gene is regarded as an activity, and it is associated with a group of students and their teacher. The gene is obtained as a result of the assignment of an activity to the teacher (who leads the activity) and to a group of students (who participate in the activity). A chromosome (schedule) consists of genes (activities). The idea of scheduling is to allocate activities in a plan (genes in the chromosome), i.e. the problem to be solved is treated as a problem of assignment of entities within available slots. Chromosomes prepared in this way are evaluated against constrains and standard techniques for evolving improving solutions are applied. More specific description can be found in [5].

#### 2.1.2 Artificial Immune Systems (AIS)

Artificial Immune Systems (AIS) are based on the metaphor of the natural immune system, and work by "focusing" on anomaly detection [15]. The main idea of AIS is to operate on a population of antibodies (feasible timetables) and using proper method (immune algorithm (IA), or a hybrid, e.g. combining IA with GA) to find and replace an "anomalous entity" (bad timetable) with a better solution. In [16], application of IAS to the university timetabling has been described. Authors presented three kinds of algorithms using AIS: clonal selection algorithm, immune network algorithm and a negative selection algorithm.

#### 2.1.3 Graph coloring

To solve the timetabling problem, edge and vertex coloring can be applied. (i) Vertex coloring. In this approach, all activities are vertices. Edges indicate pairs of vertices (activities) that cannot be scheduled at the same time (e.g. when the same teacher leads them). The core of the method is to perform legal coloring of the graph representing conflicts, where colors indicate time slots. (ii) Edge coloring. Here, a very simple example of a possible approach is a bipartite graph coloring, where the first set are vertices that present teachers and the second set presents activities. The idea is to color this graph and (similarly as in the case of vertex coloring) colors indicate time slots [3].

#### 2.1.4 Simplex method

An example of using this method in scheduling we can find in [4]. The constraints are transformed into a system of linear equations. There are 3 main steps of solving the problem: (i) Generate acceptable (non-negative) solution baseline (initial). (ii) Check the optimality of the obtained solution. (iii) If it is not optimal, generate a new basic feasible solution that is not worse than the one previously obtained, and check if the obtained solution is optimal. If it is optimal, the process is completed (because better solution cannot be found). In other words, the last obtained solution is considered optimal.

#### 2.1.5 Tabu Search

The basic paradigm of this heuristic method (examples of which we can be seen in [17]) is to use the search history (distribution of activities in the timetable) to guide the local search approaches to overcome the problem of solver being stuck in local optimum (repeating suboptimal results). It is possible to combine this method with other algorithms, e.g. with graph coloring.

Regardless of how successful are these methods, **all** of them have a major disadvantage. Namely, it is practically impossible to change an already existing schedule. Observe that, when scheduling courses at a university (which is our application area) it is necessary not only to generate a "high quality schedule" (where quality is judged against one or more criteria, see below), but also to provide mechanisms that would allow to shift an individual class, add a new one into an existing schedule, (ex)change rooms, etc. Since the above described methods treat the schedule from a "holistic" perspective, re-scheduling a single class, e.g. for a teacher that got sick during the semester, is a relatively complex task. Simply said, this is not what these methods were created for. Obviously, such changes can be accomplished manually, or by using additional (separate) software, but this means that multiple approaches have to be combined. One, to generate the "initial" schedule, and one to manage it during the course of the semester. Moreover, the larger the input data set (and the more links between items in this set) the more complex is the problem. As a result, the algorithms that can solve the timetabling problem need more computational power and take more time to complete. Note that, due to the holistic approach, in each "step," these algorithms treat the complete problem at once.

Interestingly, it can be stipulated that software agents can handle *both* the schedule preparation and its management, as they are characterized by autonomy, reactiveness, and ability to communicate / negotiate (see, [6, 12], for discussion of application of agents in timetabling).

Furthermore, as will be shown, agents allow to "divide" the problem into smaller subproblems that are solved in each step; thus reducing its overall complexity. Therefore, we have developed a prototype of an agent-based timetabling system (AgentPlanner), which uses agent negotiations to create and maintain (modify) the schedule. In what follows, we describe the AgentPlanner and discuss results of its experimental evaluation, when applied to scheduling university courses.

## 2.2 Agent based methods

Before proceeding with description of the AgentPlanner, let us summarize the state-of-the-art in using agents in timetabling. We found a few agent based systems (some of them are described in [12]) that are used for planning in logistics, production, defense and insurance sectors, e.g. a scheduling system for taxi companies ([13]) or hospitals ([14]). Obviously, some of them could be reorganized for school timetabling, but it would be difficult because they are designed to solve a specific problem. However, there are also agent-based systems, designed specially for timetabling.

Authors of [9] use a divide-and-conquer approach, combined with software agent technology. Timetabling Agents generate initial solutions, where each agent is responsible for the solution of a specific subproblem. Every agent uses a different heuristic. It is a big advantage, because this approach can apply proper heuristic (appropriate to the specific problem). Moreover there is a Mediator Agent, guarding that all plans are arranged, while satisfying predefined criteria and constraints. Test data is divided into three categories: small, medium and large, and the large dataset(s) are actually big enough to be considered realistic (matching the actual situation at a university). Unfortunately, it is not described how complex are the links between the items. For example, in our test data, students can belong to multiple different groups (obligatory, elective, language group, etc.), thus avoiding a collisions of students' activities is important (but relatively difficult).

Paper [10] describes a similar approach as [9], but there are three types of guarding agents. The first is making sure that the generated sample solutions comply with the main requirements (no collision for trainers, only one class in one room, etc.). The second type agents is guarding "hard" constrains. The third type of agents guards the "soft" constraints (good to have, but not necessary). The number of second and third type of agents depends on the specific course timetabling problem. System can work in two modes: (i) where second and third type of agents evaluate proposals of the first type, and (ii) where second and third type of agents try to improve proposals put forward by the first agent. Unfortunately, in the paper the test data is not well described; just the grid of 5 days and 9 time units are specified, that gives much more space where the solution can be found than in our research (5 days x 6 time units). According to the author "the results are promising" but it is not possible to verify them because there are no actual results in the paper.

Author of [11], is firstly looking for any matching solution and then optimizes it. In the proposed approach, a larger number of agents types (than in the above described papers) is proposed. Almost every element in the plan has an agent representing it: CourseAgent, TeacherAgent, StudentAgent, RoomAgent. A potential disadvantage of this approach is that the more data is to be passed around, and the larger the number of agents in the system, the larger is the number of messages that are to be exchanged. This causes two possible problems: (i) an increased chance of a bottleneck, and (ii) problem of synchronization of communication and actions of the system. In the paper, author showed results when 40 agents were used. However, this is rather a small problem, and does not show any conclusive results concerning scalability of the proposed approach.

Overall, papers [9, 10, 11] describe very interesting approaches to application of agents to timetabling, but their are not well tested. Number of "elements" in the test data is not large and not complex enough to mimick real-world situations. Furthermore, during the our research we stumbled upon many papers of this kind; interesting approaches tested on non-realistic data sets, so we will omit them here.

There exist approaches similar to the AgentPlanner. Negotiation involving teachers' time preferences have been used in [19]. Here, authors use four classes of agents: (1) Teacher Agents, (2) Classroom Agents, (3) History Agent, and (4) two Interface Agents. The role of the Interface Agents is to initialize other agents based on the user setup. The core of the algorithm is negotiations between Teacher Agents who send propositions (prepared on the basis of Teacher preferences) to proper Classroom Agents. They consider the proposals, with help of the History Agent that contains information about all timetables and allows detection of collisions. The size of the dataset is impressive (but there is no description of its complexity) and results are very encouraging. Unfortunately it is not possible to compare effectively these results with our approach (described below) because authors adopted a different evaluation criteria. They focus on number of sent and analysed messages, which guarantee the speed of the system. Teachers' time preferences are used in scheduling but authors did not check if they have been actually fulfilled (to what extent).

Similar approach to result evaluation present authors of [20]. In their work, every weekday is a different platform, where Course Agents are run. They negotiate with each other (via a SignboardAgent – a coordinator that helps find a free time slot). Primary results found in the paper are that using a distributed architecture is better than a centralized one, because of reduction of run time.

Finally, work reported in [21] shows that many agent based systems do not deal with re-scheduling of an existing schedule (e.g. system presented in [20]). To deal with the problem, authors use the Probability Collective theory. Their experimental results are promising, but they cannot be naturally compared with our approach as they (again) have different criteria of evaluation (e.g. time of running having various sets of data or evolution of the probability collective).

Summarizing, results of our research into the state-of-the-art in timetabling have revealed three groups of results. First, large number of "global" approaches to finding the optimal schedule. Here, their main disadvantage is a difficulty to modifying the schedule in response to the changes that occur during its realization. This latter feature is particularly important when dealing with real-world schedule that has to run during a semester at a university / school / college. Second group involves agent-based solutions that were not properly tested, or tested on data sets that were "not complex enough" to represent real-world situations. Finally, agent-based approaches that were somewhat similar to our approach, but in their design and experiment focused on different aspects of timetabling than what was our main goal.

## 3 AgentPlanner – preliminary considerations

The most important attribute of our approach is take into account teachers' time preferences. In this context, we have started our work by analysing the real needs of faculty members of the Mathematics, Physics and Informatics Department of the University of Gdańsk.

The results of completed analysis allowed us to specify the requirements for the development of our agent-based course scheduling system (the AgentPlanner). First, the AgentPlanner has to deal with both scenarios: (1) to develop a timetable of academic courses in accordance with specified restrictions (creation of a new timetable), and (2) to manage it; i.e. be capable of making requested changes / modifications in the existing class schedule (timetable maintenance). It is important to note that the selected application area: scheduling of courses at a university, has guided formulation of functional and non-functional requirements for the developed system. University course scheduling means that, in addition to creation of an initial course schedule for a given semester, the AgentPlanner has to be able to deal, among others, with: change of location(s) of selected laboratory groups / lectures, sickness of a teacher (i.e. rescheduling missed classes for a later date), adding new activities (e.g. an unscheduled examination caused by multiple students failing the first attempt), etc.

After analysing the actual scheduling process that takes place at the University of Gdańsk, it was decided that only the *Planner* (human system administrator) will be able to run the AgentPlanner to create the timetable. In addition, the *Planner* is going to be the only person who will be authorized to make schedule changes in the database (in particular, during the timetable maintenance phase).

In the AgentPlanner we have introduced some restrictions on the implemented functions. In this way we were able to focus on core functionality and complete experimental evaluation of our approach. In this way, after the initial course schedule is created, both the *Planner* and the *teacher* can send two types of requests: (a) to insert a new activity (group exercises, laboratory, lecture), requiring reorganization of the plan, and (b) to change location of, already scheduled, activity(ies). Observe that both types of requests may impact other teachers. Hence, the proposed rescheduling (resulting from the work of the AgentPlanner) has to be negotiated with those teachers that are affected by the changes. In the current version of the AgentPlanner, to complete a change of the existing timetable, all affected teachers have to agree. Here, for the time being, we do not take into account the fact that the teacher may be forced to accept a change (e.g. by the Dean), and assume benevolence of teachers.

Analysis of the actual course scheduling process lead to the following extra requirements for any system similar to the AgentPlanner. (1) Scheduling should be completed in a reasonable time. (2) Used algorithms must be designed so that the system can be used on computers with limited power (i.e. personal computers). (3) The timetabling system should be easy to install (use well-known and well-documented software). (4) Ease of use (simplicity of the interface) is very important. (5) Timetable requires visualization both in the printed form, as well as in a form that can be sent to the website (to be displayed). Therefore, the system should have various data converters; from the database representation of the schedule, to the appropriate file formats. (6) The scheduling system should be reliable and resilient to possible errors. (7) For obvious reasons, data security is extremely important. Finally, (8) the timetabling system should be portable between various operating systems. This context let us stress, again, that the aim of our current work was not to develop a full-blown system. Therefore, the above "extra requirements" have been mostly omitted. For similar reasons, we have not considered the requirements involved in implementing the AgentPlanner on mobile devices (which may be a very useful – or ever required – functionality for an actual system).

Based on conversations with actual faculty members of the University of Gdańsk, we have formulated the initial "scheduling goals" for the AgentPlanner. As a result, the system aims at: (i) minimizing the number of days of teaching, and (2) locating activities as close as possible to each other (i.e. no big gaps between activities, resulting from some classes taking place in the morning and the remaining ones in the evening). However, it is also possible to control this process by incorporating teachers' preferences (for both: teaching days, and selected time-slots). Specifically, the teacher can rank her preferences concerning days of the week by assigning natural numbers from the interval $[0, 4]$, where 0 is considered to be "unacceptable" and 4 represents "the best option". Similar approach applies to ranking time-slots (each of them can be ranked individually; in this way we can capture preferences such as: I like to teach in the morning vs. I hate to wake up early). In this case, the interval depends on the number of time units per day (we consider $[0, 5]$). It has to be noted that, in the current design, there is no restriction on the number of teaching activities during a single day. Therefore it is possible for a teacher to have classes "all day long" (e.g. 5 courses at a given day; and no classes for the rest of the week). While seemingly unreasonable, this does reflect the actual preferences of faculty members. It is worth to mention that, in Polish universities, single lesson last 45 minutes while time unit usually consists of two lessons, i.e. 90 minutes.

It is very important to adopt some constraints that prevent input data that makes it impossible to create a plan, or that causes a negative, unreliable results of the evaluation function. Here, we have identified key steps of proper representation of time preferences (we have also utilized them when preparing the test data).

(1) All default (undefined by teacher) time units in the schedule are set to the highest possible rank (e.g. 5 for a day consisting of 6 time intervals).

(2) Teacher should consider, which time units are "the best option" for her/him and leave them without changing rank. Minimum number of highest ranked time units depends on the number of the teacher's activities.

(3) Next step is to set ranks less than the highest but higher than 0, represented as natural numbers from the interval $[1, \text{HIGHEST\_RANK} - 1]$, where 1 means that her/his presence is possible but inconvenient. Proceeding in this way teacher can affect allocation of his/her activ-

ities. As a result there is higher probability to achieve a plan that is better than when one does not make such precise specifications.

(4) Teacher should carefully consider, which time units are "unacceptable" for him/her and rank them as 0. However, it is obvious that the more zeros, the more difficult the problem becomes. Therefore, the teacher should use it only if presence is really impossible at that time.

As far as the representation of interests of students is concerned, the prototype takes into account (what we believe to be) the key aspects of a plan: minimization of collisions of courses, number of days of instruction, and gaps during the day. However, we have to admit that the current version of the AgentPlanner has been implemented with primary focus on teacher satisfaction.

Finally, in the current version of the AgentPlanner system, the timetable is created for a single department, located in a single building.

## 3.1 AgentPlanner as an agent-based system

Recall that the AgentPlanner has been conceptualized as an agent system. On the basis of the requirements analysis, we have envisioned it as depicted in Figure 1.

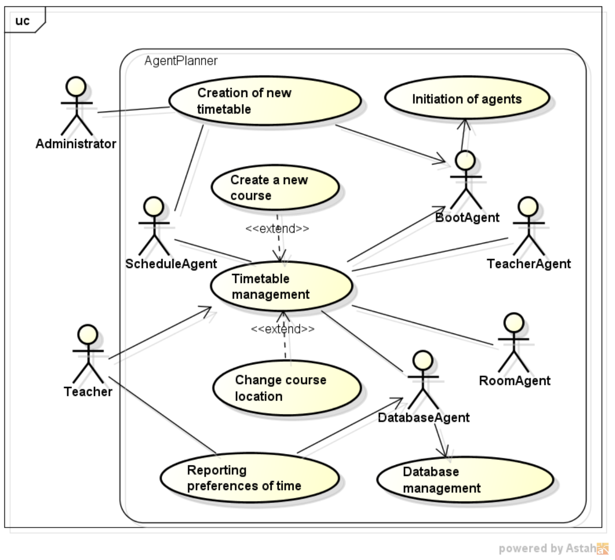


Figure 1: AgentPlanner use case diagram.

Here, we recognize the two main functions of the system, the *Creation of a new timetable*, and *Timetable management*, as well as a number of additional functions needed to complete the two main ones. The current design of the system has only two "external" actors: the *Planner* and the *teacher*. In the future, one may need to include in the design also the *student* actor, but this would lead to a system that is out of scope of our current work. Analysing functional and non-functional requirements of the AgentPlanner system, we have came to the conclusion that it should consist of the following agents:

- *BootAgent*, with the only task to create and start other agents that are required in the AgentPlanner system.

- *DatabaseAgent*, responsible for connection the system with database. All agents have access to data via the *DatabaseAgent*. This agent was found to be required to streamline and organize access to the data stored in the database.

- *RoomsAgent* represents rooms in the scheduling process. It downloads (from the database), filter and store data about rooms in the system (e.g. type of every room, seating capacity, etc.).

- *TeacherAgent* acts on behalf of a teacher (both during creation and management of the timetable). It stores: (i) information about the teacher (including personal data that has to be protected), (ii) list of activities (courses / groups) taught by the teacher, (iii) list of rooms (meeting the requirements of each group and course, e.g. laboratory group has to be scheduled in a laboratory); obtained from the *RoomsAgent*), (iv) results of the location evaluation function (described in Subsection 4.1), and (v) teacher's current timetable.

- *ScheduleAgent* is the central agent of the negotiation algorithm. It "knows" teachers involved in current negotiations (*TeacherAgents* that represent them). It also has access (read and write) to the timetable database (via the *DatabaseAgent*). Note that, all data concerning the currently considered timetable, is (after each change) saved in the database. This allows the *ScheduleAgent* to effectively issue verdicts, which room should be assigned to which requesting teacher (as it knows which rooms are already occupied and which are still available). Note that we are aware of the fact that, in a very large scheduling problem, the *ScheduleAgent* may become a bottleneck. However, solving this problem is out of scope of the current contribution. This is especially the case since the time to calculate the schedule for the (realistic) size of data used in our experiments was acceptable (it took about 1 minute to complete the scheduling task; after all agents were started and provided with their input data).

The negotiation process is between *ScheduleAgent* (a judge) and *TeacherAgents* (representing teachers) but not between *TeacherAgents*. Obviously, it is centralized model, where it is possible to run into a bottleneck (caused by limited processing capability of the *ScheduleAgent*; see, above). However, observe that very complex, time-consuming, negotiations take place only once – during early phases of creation of the initial schedule. This is "acceptable" as the time-pressure is, usually, not too-serious. At the same time, adaptations to the existing schedule, which take place during the semester do not take long time, as they involve only small number of agents (representing teachers affected by the required change(s)).

It is easy to note that the *DatabaseAgent* and the *RoomsAgent* did not have to be implemented as full-fledged agents. For instance, they could have been designed as FIPA-style services [23]. However, we have decided (for the simplicity and uniformity of implementation) to use agents "across the board". Acknowledging that this decision may seem somewhat controversial, we believe that our choice of an implementation method (i) has merit, and (ii) does not influence the experimental results supporting our approach (quality of the obtained solution).

## 4 Implementation of the AgentPlanner

Based on the above considerations, we have decided that the AgentPlanner should be implemented as a client-server-type system, where all operations concerning generation and maintenance of the timetable are going to be executed as an agent-based server application, while the client component will be responsible only for sending requests and reviewing / accessing results. It is important to keep in mind that access to the database is allowed only on the server side of the application, so every request of the client, or any other agent in system, has to be handled by the *DatabaseAgent*. This decision was based on the fact that, our software of choice (the JADE agent platform), does not provide a robust GUI for user interfaces. Therefore, following advice found in [8] we have decided to clearly separate the agent and non-agent functionality. Furthermore, in the current version of the prototype, the client application is simplified to a "line interface," while the server application has only functionalities needed for the two timetabling operations (schedule creation and maintenance). All data needed for the tests was inserted manually to the database via SQL scripts, or other scripts written for this purpose.

### 4.1 Evaluation algorithm

The core of the timetabling mechanism is the evaluation algorithm. Here, the *TeacherAgent*(s) evaluate the locations (room information received from the *RoomsAgent*) that best match the need of the teachers. The evaluation algorithm must takes into account: priority of course and lecture, links of students with other groups, teacher preferences, and the current state of the timetable. Overall, every activity has an assigned priority, which describes how important it is for the teacher (e.g. a lecture may have higher scheduling priority than a laboratory). Furthermore, some courses are "more important" than others, e.g. a core course may have a higher rank than an elective (all students have to take the core course, while they may sometimes be "forced" to take a different elective – to avoid course collision(s)). Moreover, courses related to the major (e.g. in our case, CS courses) have higher rank than non-major ones (e.g. psychology courses). Separately, when considering the current timetable, priority is given to activities that can be assigned in the time-vicinity of the already scheduled ones. In this way, the total number of gaps in the schedule of the teacher (and possibly students) can be minimized. The evaluation algorithm works as follows (1). The current

```
Data: S = priority_of_course * 10 + links_number
if day_priority or time_slot_priority is equal 0 then
    do not add room from this time slot to the list
else
    S := S * day_priority * time_slot_priority;
    if there are other lessons in this day then
        S:= S+5
    end
    if there are other lessons around time_slot then
        S:= S+5
    end
end
Result: S
```

**Algorithm 1:** Evaluation Algorithm.

version of the evaluation algorithm is quite different from the one reported in [1]. The initial value is the sum of the activity priority (multiplied by 10, because in this way, in the experiments, we have received better results) and the number of all groups, to which students from this activity belong. This should be understood as follows: the more links / dependencies between data elements, the harder it is to put the activity in the plan, because the algorithm has to avoid collisions between connected groups. Therefore, the most complex situations should be resolved in the first place. The next step is to consider the most important factor, the teacher's time preferences, i.e. rank of day and time unit. If one of them is equal 0, that means that the teacher cannot lead activity at that time, and function does not add this location to the list. The last element is the evaluation of the "vicinity". It is important to reduce the "time gaps" in the plan. Therefore, to place an activity between two others is the highest ranked situation.

### 4.2 Timetable planning algorithm

Let us now consider creation of a new timetable. Recall, that the approach is based on a single "judge" (the *ScheduleAgent*), having access to the current timetable (which initially is empty). The *ScheduleAgent* negotiates the timetable with the *TeacherAgent(s)*, using information obtained from the *RoomsAgent*. Negotiations are divided into rounds (since in each round at least one activity is places in the schedule, their number is not larger than the total number of all "activities" – courses / exercise groups / laboratories – of all teachers). Due to the lack of space, we omit the pseudo-code (it is about 4 pages long and can be found in [22]). The general idea of actions that are performed in a single round is depicted in figure 2. Each round begins with the start signal (message) from the *ScheduleAgent* to the *TeacherAgent(s)* (that still have activities to allocate). During a single round, every *TeacherAgent* considers an activity from the list of all teacher's activ-

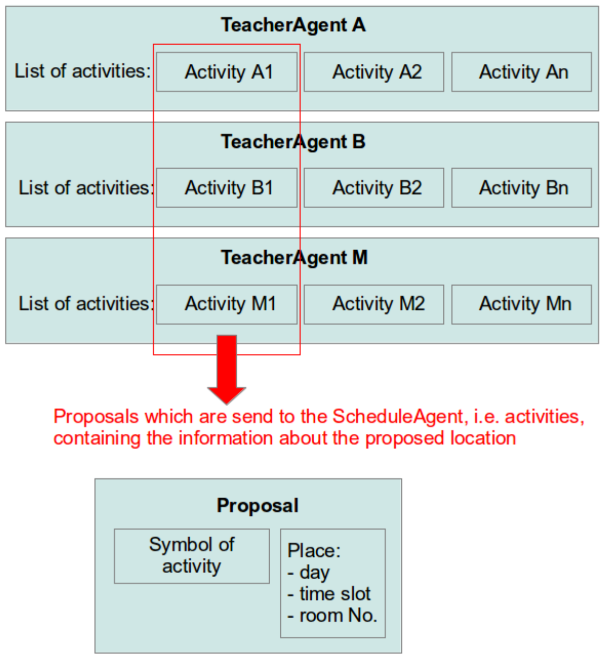


Figure 2: General schema of single round (sending proposals).

ities (initially sorted according to the priority of the activity type) and selects the "most important one". If activities list is empty then the *TeacherAgent* sends to the *ScheduleAgent* a message that it resigns from further negotiations. If not, the next step is to prepare a list of rooms acceptable for the considered activity and sort them according to the results of the evaluation algorithm (1). The *TeacherAgent* selects the most desirable location and sends the proposal to the *ScheduleAgent*. The proposal consists of: (i) symbol of the activity, (ii) number of the week day, (iii) number of the time slot, (iv) result of the evaluation algorithm. When all *TeacherAgents* send their proposals, the *ScheduleAgent* considers them and accepts the best (placing these activities into the current timetable), while rejecting others. Note that, in the current version, we omit the case when one (or more) *TeacherAgent(s)* do not send their proposals. While, in general, this is an important issue for the design of agent-based distributed systems, handling such anomaly is out of scope of our current work. The decision to accept a request depends on two factor: (1) is the requested location already occupied by another activity, and (2) does a given request involve course collisions. Obviously, it is possible that multiple *TeacherAgents* may ask for the same location. In this case, the *ScheduleAgent* selects the one that delivers the best value of the evaluation algorithm. It is also possible that proposals from multiple *TeacherAgents* "have the same value". In this case, the *ScheduleAgent* draws a winner (randomly). Next, the *ScheduleAgent* sends messages to the *TeacherAgents*, about rejected proposals. Then, the *TeacherAgents* select the next best location from the list, and create proposals for the *ScheduleAgent*. If all of its proposals are rejected, then *TeacherAgent* resigns from the given round of negotiations. The result of this decision is recorded in the database for the information of the *Planner*. The round ends when every *TeacherAgent* gets a place for its activity, or when some unscheduled requests cannot be satisfied. Note that, in a single round, the total number of evaluated requests is equal to the number of teachers with unscheduled activities and thus is relatively small and systematically decreasing (when at least some teachers have their schedules complete).

Observe that this approach is based on the assumption that all teachers have the same chances. This is because, in a single round, every *TeacherAgent* can reserve one permanent place for one of its activities. For example, if a professor has two seminar lectures, while an assistant has two exercise groups, then in the first round each one of them will "book" a room for one of their activities (regardless of their position in the academic hierarchy). However, it is not clear if such democratic approach would be sustainable in the real-life university course scheduling. If this was not the case, then the structure of academic dependencies (who, in a given moment, is more important than others) could be represented, as weights, in the evaluation function. However, exploring this possibility is out of scope of our interests.

Before the beginning of the next round, the *ScheduleAgent* receives messages from the *TeacherAgents* with resignations from the given round of negotiation. In response to these messages, it suspends the main thread of negotiations, and runs the timetable reorganization algorithm (see subsection 4.3) to deploy the rejected activities into the current schedule.

It could happen that the reorganization (adding new activity) is impossible, then the activity is added to list of rejected activities, for inspection by the human *Planner*. In this case, the *Planner* has to figure out how to improve the input data, e.g. to change the time preferences of teacher (e.g. by contacting her/him directly). After the schedule is reorganized and, previously rejected proposals are added, the *ScheduleAgent* returns to the main thread and starts the next round of negotiations. Thus, the timetabling algorithm continues from reception of the next group of proposals from those *TeacherAgents* that still have unscheduled activities. The sequence diagram of the timetable planning process is represented in figure 3. After an extended analysis of results reported in [1], we have made an important modification to this algorithm, aimed at eliminating collisions among student activities. Originally, during the negotiations, student collisions were checked against groups that were already inserted into the timetable, but not against the remaining groups that were involved in the given negotiation step. To deal with the collisions, we have decided to sort all proposals according to the results of the evaluation algorithm (see, section 4.1) and insert activities iteratively beginning from the top of the list. However, before inserting an activity into the plan, we now check for possible collisions between student activities and if there are any,

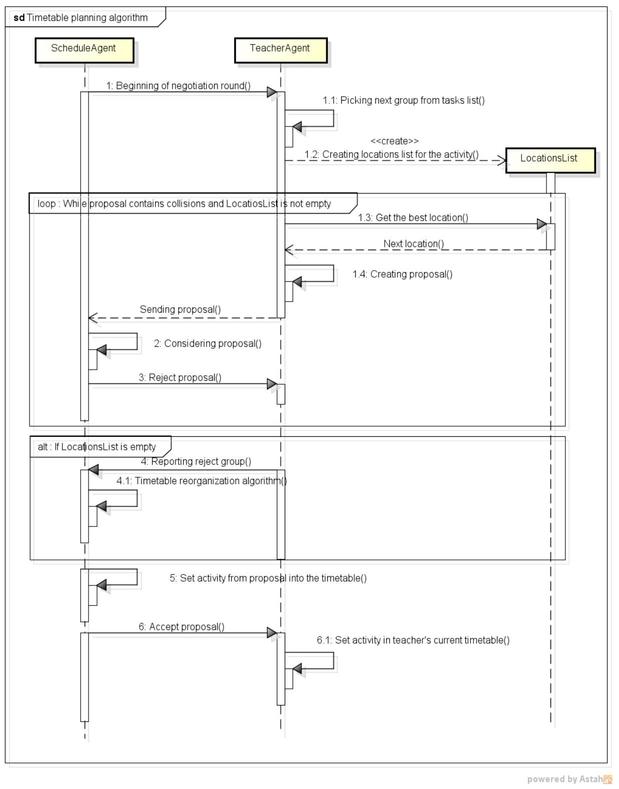

sd Timetable planning algorithm
ScheduleAgent
TeacherAgent
1: Beginning of negotiation round()
1.1: Picking next group from tasks list()
<<create>>
1.2: Creating locations list for the activity()
LocationsList
loop : While proposal contains collisions and LocatiosList is not empty
1.3: Get the best location()
Next location()
1.4: Creating proposal()
Sending proposal()
2: Considering proposal()
3: Reject proposal()
alt : If LocationsList is empty
4: Reporting reject group()
4.1: Timetable reorganization algorithm()
5: Set activity from proposal into the timetable()
6: Accept proposal()
6.1: Set activity in teacher's current timetable()
powered by Astah

Figure 3: Timetable planning algorithm sequence diagram.

we reject such proposal. This approach slows the progress of timetabling, but thanks to it we can generate timetable with no collisions between student activities.

### 4.3 Timetable reorganization algorithm

The timetable reorganization algorithm is used in two situations. First, when in a single round, all proposals of the *TeacherAgent* (concerning a given activity) were rejected by the *ScheduleAgent*. Then, the *ScheduleAgent*, has to find a place for such activity in the current timetable. Second, during the timetable maintenance phase, when the requested changes require schedule reorganization.

The list of activities that require adding, via the timetable reorganization algorithm, is based on messages received from the *TeacherAgents* and stored (by the *ScheduleAgent*) in the rejected activity list (and ordered according to their priority). The *ScheduleAgent* considers these messages one by one.

When the *TeacherAgent* T1 wants to take location that is occupied by the *TeacherAgent* T2, then the *ScheduleAgent* sends to the T2 a message with a proposal of release this location. Then, the *TeacherAgent* T2 requests a new place for its activity. If it succeeds, the *TeacherAgent* T2 accepts the proposal and the *TeacherAgent* T1 can book the requested room for its activity. If not, the *TeacherAgent* T1 has to find another place. Here, it is assumed that all *TeacherAgents* are cooperating and all have a chance to put all activities in the timetable, even in a "conflict situation". Furthermore, a simple mechanism that prevents this phase from reaching a deadlock (when all agents depend on others releasing their rooms, "in a loop") is applied by the *ScheduleAgent*. In the pessimistic situation (very difficult input dataset), activity may find no place. In that case, the algorithm cannot deal with it and the *Planner* (human being) is informed about the situation, and has to change the dataset (e.g. by convincing a teacher to change preferences) or to put the activity manually into the schedule.

On the other hand, when making changes in the existing plan (during the semester) owner of the *TeacherAgent* T2 would receive a request to accept the proposed change. Obviously, in this case, success of the schedule adjustment depends in large part on the benevolence of the involved instructors. However, let us recall that, for the time being, we assume such benevolence.

### 4.4 Technologies used in the implementation

The following technologies were used to implement the AgentPlanner:

– Agent platform: JADE (version 4.3.0) [7]

– MySQL database 5.1.69 [24]

– NetBeans 7.0.1 [25]

## 5 System testing and analysis of results

### 5.1 Test data

The test data used in our experiments was prepared on the basis of the actual organizational structure and room base of the Mathematics, Physics and Informatics Department of the University of Gdańsk (UG MFI). To evaluate the efficiency of the proposed method, the results obtained by the AgentPlanner were compared with these produced by the Free Timetabling Software (FET, version: 5.19.1) [2], which uses the GA. Each software solved the same timetabling problem.

The problem involved 5 days (Monday-Friday), each consisting of 6 time slots, and the building with 21 rooms, which results in a "grid" that contains 630 locations. There were 301 activity groups (62 courses, comprising total of 734 students). While it may seem that there is "a lot of space" to allocate activities, constraints imposed by the teachers and the student grouping limited this space considerably. Specifically, time preferences of 78 teachers were the main limiting factor, without it, both algorithms would find a solution without any problem (for the grid: 5 days x 6 time units x 21 rooms). Moreover, the problem is more difficult, because connections between students and groups are very complex, primarily due to the possibility of choosing elective courses. For example, a single student can have a few core courses (consisting of lectures for all students and exercises/laboratories for student groups) and (s)he has to choose a few elective courses (like facultatives, language(s), seminars, etc.). The most difficult situation involved an activity that consisted of 120 students, who belonged to 107 other activities. In this situation, the timetabling algorithms have to allocate these 107 activities in the schedule without collisions (for both teachers and students) and additionally take into account teachers time preferences.

Let us now stress that the input data and the setup of the FET system, reported in [1], was based on the *actual settings* used in the UG MFI department. Interestingly, the obtained results were not very impressive. However, after the publication, a co-author of the FET system contacted us and shared insights and advice how to setup the FET system better. Let us now make a few comments about the specific issues in experimental setup of the AgentPlanner and the FET.

First, it is important to explain, why we could not set 100% of constraints in the FET. In general, during preparation of the data for both systems (FET and AgentPlanner) we tried make them most similar. We have set up subjects, activities (with tags), subactivities, rooms (recall that we solve the problem for one department and one building only), teachers, students (assigned to activities). The AgentPlanner has been already designed to resolve the most important (hard) constraints like elimination of collisions, minimization number of days of instruction and gaps during the days. The only light constraint, we took into account, were time preferences, because they are the core of the AgentPlanner algorithms. The AgentPlanner takes into account teacher's time preferences by using ranks. In the FET we have similar option but we can only set the time units when teacher cannot lead any activity, and we have used this option. However, in the FET it is impossible to make a more specific ranking of teacher preferences.

It is worth mentioning that there is significant difference between the AgentPlanner and the FET rank function. In the AgentPlanner we can describe the time preference of teacher using sets of natural numbers: (1) $[0, d]$ where $d$ is maximum number of days, and describe the best option for the teacher, while 0 means that the teacher absolutely cannot have an activity in that day; (2) $[0, t]$ where $t$ is maximum number of time units during the day, and describe the best option for the teacher, and 0 means that teacher absolutely cannot have an activity in that time. In the FET, we can set the rank for only the time units, when the teacher cannot have an activity. Actually it is possible to describe the percentage value number that describes the time unit, but it is only one and we can use it to describe any time.

The next important issue is preparation of the student test data. In the AgentPlanner, we can describe students group as activity and link with it any single student that belongs to it. Due to this setup, it is very easy to detect the collision of an activity, which we try to insert into the schedule. It is possible to use similar solution in the FET. We are able to create 3 types of groups: years, groups, subgroups. In [1] we understood it literally, so set of students was divided into years (IT, Math), groups (as a lectures of subjects) and subgroups (as a parts of lectures from groups). This approach prevented linking students individually with groups and reduced the potential for effective detection of collisions. As a result, we have decided to prepare 2 types of groups, similar to the AgentPlanner: first contains activities of subjects, while in the second the students that are linked directly to these activities. As a result, the FET achieved much better results and effectively eliminated collisions between student activities. We plan to suggest this approach to the *Planner* at the Mathematics, Physics and Informatics Department of the University of Gdańsk.

## 5.2 Comparison metrics

Note that the AgentPlanner is being designed with focus on the "human factor" (convenience of teachers and students). In this way it differs from most approaches reported in work summarized in Section 2.1. Therefore, we have constructed a teacher and a student satisfaction functions that estimate satisfaction of this criterion. The teacher satisfaction is represented by the following formula:

$$S_T = \frac{s * 100}{a * n * m} \tag{1}$$

where: $a$ is the number of activities of a teacher in a given semester, $n$ is highest rank assigned to any day, $m$ is the highest rank assigned to any time slot. Furthermore, $s$ is the sum of evaluations of all time units of all activities of teacher, obtained by using formula 2.

$$s = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} D[i] * T[i][j] \tag{2}$$

Here, $D[i]$ represents the evaluation of each day of the schedule, while $T[i][j]$ (formula 3) represents evaluation of each of time slots assigned in the schedule of that day.

$$T[i][j] = \begin{cases} < 0, m > & \text{teacher's evaluation of time unit} \\ 0 & \text{when teacher has no activity} \end{cases} \tag{3}$$

In other words, in the numerator we represent the actual time slots assigned by the planing software, while in the denominator we represent the best potential schedule. In this way we introduce a measure that allows us to capture satisfaction of the teacher represented as a percent of the schedule that would be an ideal one.

The student satisfaction is assessed as follows. We start from 100% satisfaction and subtract: (1) 10% for one collision between the desired activities, (2) 10% for two collisions, (3) 20% for more than two collisions, (4) 10% for one extra gap between activities (we allow for one gap during a day), (5) 10% for two extra gaps, (6) 20% for more than two extra gaps, (7) 10% for one additional day (the situation when there is more days than necessary), (8) 10% for more than one additional day. Assessment of student satisfaction was conceptualized in this way, as it is impossible (at least in the current version of the AgentPlanner) to include in the process (and aggregate in some way) individual preferences of each student. While somewhat artificial, we believe that this function gives a reasonable way of assessing student satisfaction. Obviously, values 10%, 20% etc. are arbitrary ones, but they allow us to quantitatively capture the quality of schedule (seen from the student perspective).

## 5.3 Analysis timetable creation

Because the FET uses genetic algorithms, every generated schedule is different. Therefore, to get a reliable results of the test, we decided to run it 100 times and, in what follows, we report the best results for both teachers and students. In other words, each time we present two outcomes obtained by FET. This has to be done in this way because (in all reported cases) the best schedule for teachers is **not** the best one for students. In the case of the AgentPlanner, there is only one result (for the given set of test data). This is different than in the case of the results reported in [1], since we have resigned from the random factor used there.

In Figure 4 and Figure 5, we depict the schedule satisfaction results, for all 78 teachers, originating from the AgentPlanner and two results for the FET (the best result for the teachers and for the students). We can see that the teachers achieved higher satisfaction in the case of the AgentPlanner results. We compared the results with the average, the best, and the worst result of all test runs of the FET. The more specific conclusions follow. In Table 1 we can see

| Runs | Average | Max. | Min. |
|---|---|---|---|
| AgentPlanner | 98.03% | 100.00% | 66.67% |
| FET (all) | 93.16% | 100.00% | 30.00% |
| FET (the best) | 94.71% | 100.00% | 66.67% |
| FET (the worst) | 91.73% | 100.00% | 50.00% |
| FET | 93.95% | 100.00% | 55.00% |

Table 1: Comparison of teachers satisfaction function results.

in a row: (1) results for all teachers for the AgentPlanner, (2) the average result for the FET from all runs, (3) results of the best FET run for the teachers, (4) results of the worst FET run for the teachers, (5) results of the best FET run for the students. The columns describe: (1) “Average” the number of all results of all test runs, (2) “Max.” the best result of all test runs, (3) “Min.” the worst result of all test runs. We can see that the AgentPlanner achieves better result than the best one obtained by the FET. Note that every case has 100% of maximum satisfaction, because (in the test data) there were teachers who did not define their time preferences, so they were “happy” with what they got.

When we compare Figure 6 with Figure 7 and Figure 8 we can see that in the AgentPlanner the set of satisfied teachers was 14.1% higher than the same set obtained using the FET (for the best result) and 17.95% higher than in the case of the best result (obtained by the FET) for the students.

Next, in the figures 9 and 10, we represent student satisfaction.

The diagram in Figure 10 shows the average student schedule satisfaction for the best result for the teachers. Here, the the largest group of students belongs to the interval (70%, 80%] for the AgentPlanner (27.11% of all students), and (30%, 40%] for the FET (31.88% of all students). Observe also that, in the case of the AgentPlanner, 9.95% of students belong to the interval (90%, 100%], while in the case of the FET none student was satisfied to this extent.

In the diagram in Figure 9 we can see the average student schedule satisfaction, considering the best FET run for students. Here, the largest group of students belongs to the interval (70%, 80%] for the both systems (27.11% for the AgentPlanner and 27.80% for the FET). Observe also that, in this case the FET (only) 0.14% of students belong to the interval (90%, 100%]. In Table 2 we can see in subsequent rows: (1) results of all students for the AgentPlanner, (2) average result of the FET for all runs, (3) results of the FET run that was the best for the students, (4) results of the worst run for the students, (5) results of the best run for the teachers. We can see that the AgentPlanner achieved 6.99% better result than the best case of the FET. Moreover, the difference between the best and the worst FET result is 7.16%.

| Runs | Average | Max. | Min. |
|---|---|---|---|
| AgentPlanner | 73.27% | 100.00% | 40.00% |
| FET (all) | 59.37% | 100.00% | 40.00% |
| FET (the best) | 66.28% | 100.00% | 40.00% |
| FET (the worst) | 52.21% | 90.00% | 40.00% |
| FET | 60.41% | 100.00% | 40.00% |

Table 2: Comparison of students satisfaction function results.

In the Table 3 we depict number of gaps in students' timetables. Specifically, we depict the percentage of students who have: (1) no gaps in their schedule (the perfect situation), (2) 1 gap, (3) 2 gaps, and (4) more than 2 gaps. We present results obtained by the AgentPlanner and (as previously) two versions of the FET results (best from the point of students and teachers). In the case of the AgentPlanner, the biggest set of students (41.28%) has no gaps,

| Gaps | AgentPlanner | FET (the best) | FET |
|---|---|---|---|
| 0 | 41.28% | 33.79% | 18.26% |
| 1 | 28.34% | 26.02% | 27.79% |
| 2 | 17.57% | 14.17% | 20.71% |
| $> 2$ | 12.81% | 26.02% | 33.24% |

Table 3: Average number of gaps of students.

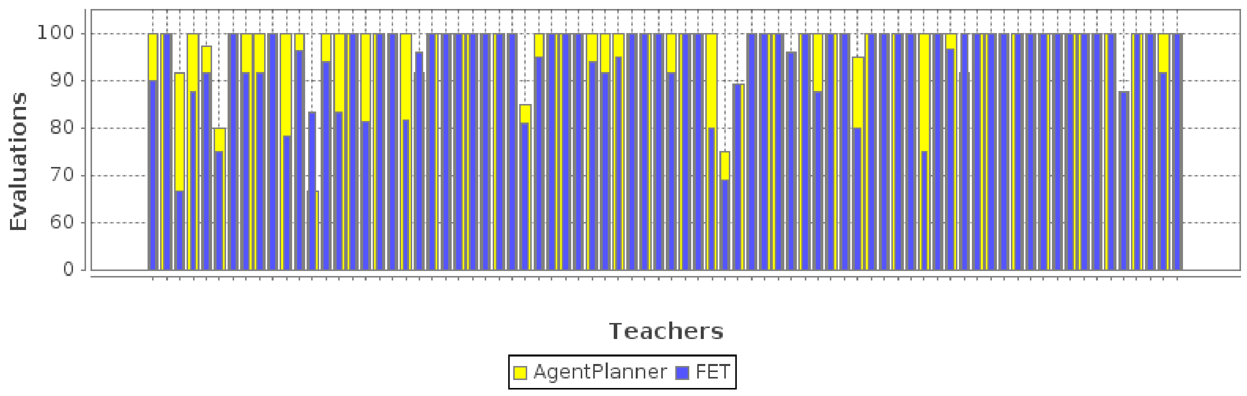


Figure 4: Satisfaction evaluations of all teachers (the best result for teachers).

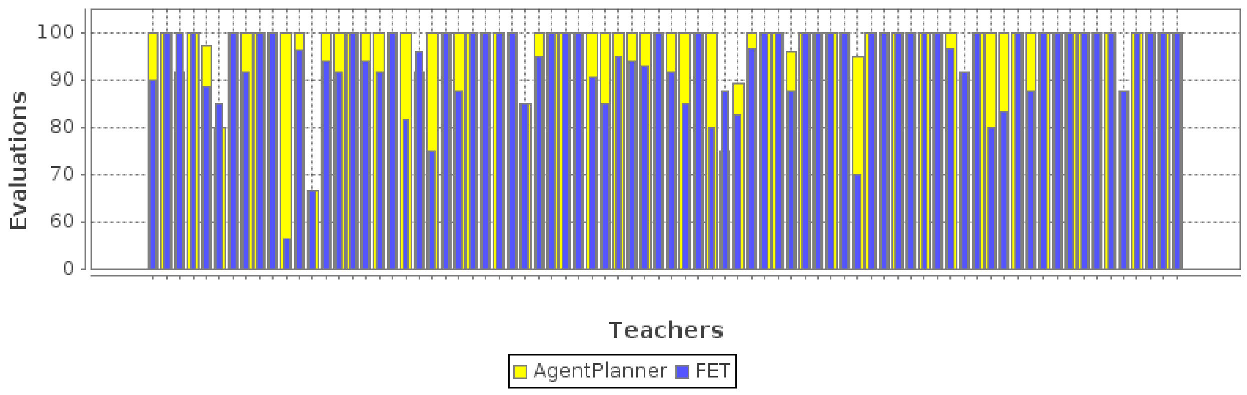


Figure 5: Satisfaction evaluations of all teachers (the best result for students).

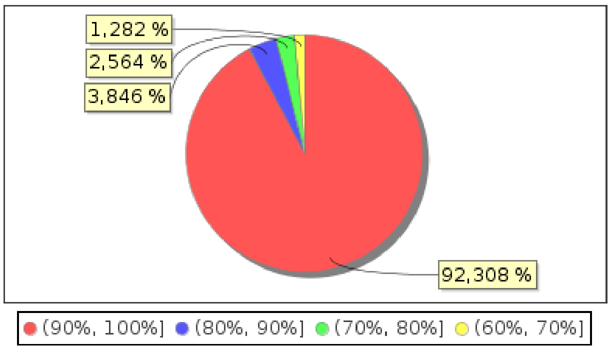


Figure 6: AgentPlanner: sets of satisfaction evaluations of teachers.

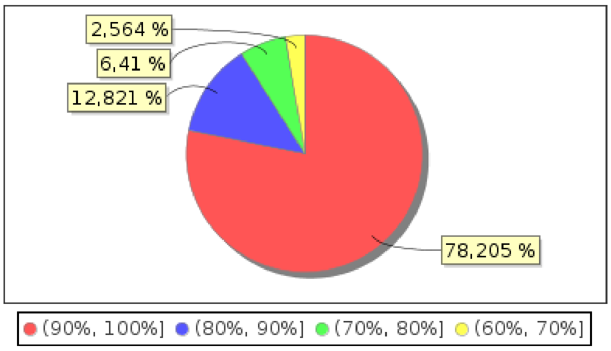


Figure 7: FET: sets of satisfaction evaluations of teachers (the best result for teachers).

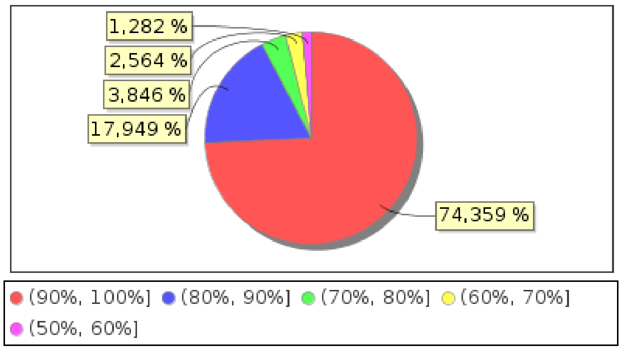


Figure 8: FET: sets of satisfaction evaluations of teachers (the best result for students).

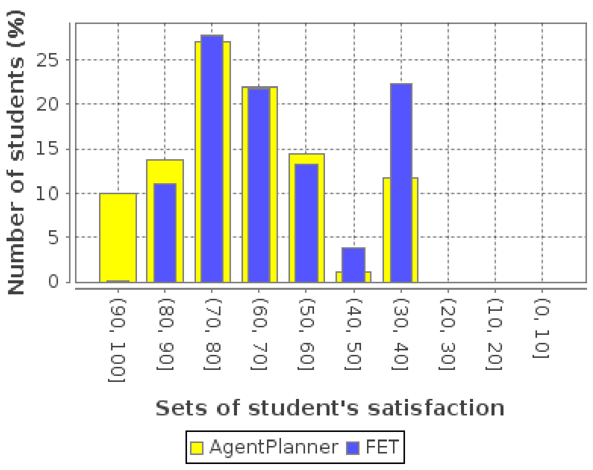


Figure 9: Average satisfaction evaluations of students (the best result for students).

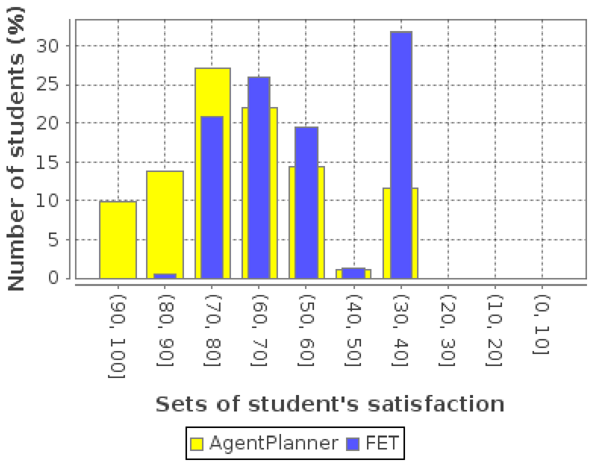


Figure 10: Average satisfaction evaluations of students (the best result for teachers).

while in the case of the "best FET" only 33.79% students reach this result, and only 18.26% for the best result for the teachers. It is very important to notice that in the case of the AgentPlanner, the number of students who have more than 2 gaps is only 12.81%, while for FET it is equal 26.02% (for the students' best result) and 33.24% (for the teachers' best result).

Table 4 captures the number of additional days (from the students' point of view). Note that multiple days of instruction are not desired (actual students of University of Gdańsk prefer to have minimal number of days of instruction as most of them are already full-time employees). Additional day is a result of a timetable "unfavorable" for the student. For example, if someone has just 4 activities during a week, the best result for her/him is a plan with all activities scheduled within a single day. But very often it is a very hard (or even impossible) task to have such schedule, because of complex data dependencies. Then, the AgentPlanner must split student's activities into more days, e.g. two (i.e. one additional day) or three (i.e. two additional days). We again consider two versions of FET results, best for the students and for the teachers.

| Days | AgentPlanner | FET (the best) | FET |
|---|---|---|---|
| 1 | 10.22% | 0.14% | 0.00% |
| 2 | 26.02% | 20.98% | 6.27% |
| $>2$ | 63.76% | 78.88% | 93.73% |

Table 4: Average number of additional days of students.

Taking into account this criterion, the AgentPlanner is quite "student friendly". Specifically, 10.21% of students have schedule with just one additional day of instruction, while the same time, in the FET, practically all students have more than one additional day. The situation is particularly "bad" in the case of FET-generated schedule which is the best for the teachers. Here, more than 93% of students have schedule with more than two additional days. Because of very complex dataset, none of the student obtained the most favorable timetable.

Let us note that the situation can be even more complex in situations, which are very natural at most universities. Consider, for instance, elective courses, which can be chosen by students from various specialties, years, and even different majors. The AgentPlanner approaches this situation in a flexible way. Since the timetable is stored in the database, there is possibility of an easy (and fast) way of checking course collisions for each student (using SQL requests). Furthermore, in our approach, we can introduce the notion of collision threshold (it is an option in the prototype application, which we have explored to a certain extent, but which is out of scope of current contribution). In other words, we can specify what percentage of collisions is acceptable. Specifically, when the tolerance threshold is exceeded, a proposal to take a given slot could be reject by the *ScheduleAgent*. This notion could be very useful, particularly in the case of very complicated and difficult to schedule timetables (e.g. involving multiple departments).

Finally, it is worthy noting that we have further checked robustness of both approaches. To make the problem more complicated we have reduced the schedule grid by one time unit (5 days x 5 time units x 21 rooms), which results in a grid that contains only 525 locations. In this case, both systems produced initial schedule when they did not consider teachers time preferences. When they were considered, the FET could not find any solution. The AgentPlanner could, but the delivered timetable contained collisions in student courses. The reason is that the links within the data set were too complex and it was not possible to create an "optimal" solution (where optimal would mean that at least there were no such collisions).

### 5.4 Testing modification of an existing plan

To test the AgentPlanner's ability to modify an existing timetable, we have experimented with insertion of an extra teacher, who is leading three activities (a single lecture, for 60 students, and two laboratories, with 30 students in each one). Note that the timetable reorganization using the FET would involve creation of a completely new schedule. Obviously, this would be "impossible" in the real–world – as it, most likely, would destroy the whole schedule (special techniques would have to be used to minimize the propagation of changes). Henceforth, the FET was omitted in this experiment.

In general, the AgentPlanner worked well. Specifically, the timetable reorganization, caused by the insertion of a new activity, marginally affected the average satisfaction of all teachers. The average satisfaction after the reorganization was 97.72% (compared to the original 98.03%; the difference of 0.31%). The average students satisfaction, after the reorganization, was 73.27% (there was no change). It is worthy to mention that no additional collisions between the activities were generated.

## 6 Flexibility of the AgentPlanner

The big advantage of the AgentPlanner is the possibility of its easy modification to use in other cases of planning, e.g. business meeting, booking of meeting rooms in company, etc. The most laborious aspect would be creation of a new database that would describe the new "reality" that the timetabling is to work with (however, its overall structure will be quite similar). Furthermore, it is very likely that the evaluation algorithm would have to be adjusted (to match the nature of the problem). After such modifications, it can be postulated that "*IndividualAgents*" (instead of *TeacherAgents*) would negotiate locations in the timetable. Note that, due to the nature of the design of the AgentPlanner, use of a different database would involve only modifications in a single agent, the *DatabaseAgent* that provides the interface to the database. Let us also stress that the current (agent-based) design and implementation of the system, which is highly modular, allows relatively easy modifications (e.g. modifications mentioned in this section).

## 7 Concluding remarks

The aim of this paper was to discuss development and experimental evaluation of an agent-based timetabling system (AgentPlanner). The proposed system was based on assumptions originating from the *actual academic settings* (class scheduling at a department at the University of Gdańsk). The results are quite encouraging. First, the AgentPlanner outperformed the state-of-the-art timetabling software based on the genetic algorithms. Second, it is capable of satisfactorily solving the problem of schedule adjustment. Finally, it is worthy to note that even though our application area was precisely defined, we believe that our systems may be successfully applied to other timetabling problems. To achieve this status, a several actions must be done.

First of all, it is necessary to design and implement: (1) GUI for the administrator (the *Planner*), to make her/him able to manage the system (i.e. to input the data, to create/change a timetable, etc.), (2) to design and implement the GUI for for teachers (to allow them to input the data, i.e. the time preferences and, possibly, other data to be specified in the future); moreover, teachers should have access to: (i) the proposed timetable, (ii) function related to a request to change the timetable, and (iii) communication with the system, e.g. to request change of the place of the activity, (3) GUI for students as a module that providing the visualisation of timetable, (4) optional, but very helpful, would be a mobile module for the teachers. The latter one could facilitate the process of *Timetabling reorganisation* (see Section 4.3).

Second, very important issues that are needed to improve the functionality of the system are: (1) introduction of further "scheduling goals" (see Section 3), i.e. constraints for both teachers and students, (2) possibility of adding more departments and buildings (considering the location and time for change a place).

The last but not least, ways of improving the AgentPlanner's core algorithms should be explored. At the moment, the most pressing problems are as follows. (1) To eliminate or reduce the bottleneck in the *Timetable planning algorithm* 4.2 in order to make it faster. (2) Consider ways of extending / modifying / improving the algorithm involved in asking the *TeacherAgents* to release the place in the *Timetabling reorganisation algorithm* (see Subsection 4.3). At the moment, the *TeacherAgent* T1 is asking the *TeacherAgent* T2 to release the location. Next, the T2 searches for a new one and accepts the proposal (to release the current location) or rejects it. This takes place in a single iteration and completes the process. However, it is easy to envision that if T2 would not be able to find a new place, it could ask T3 for an analogous operation. Obviously, this process could be repeated (with proper care taken to avoid an infinite loom of requests). This improvement would create more possibilities for adjusting the timetable.

## References

[1] Rafał Tkaczyk, Maria Ganzha, Marcin Paprzycki (2013) AgentPlanner – agent-based timetabling system – preliminary design and evaluation, *2013 17th International Conference on System Theory, Control and Computings*, Emil Petre, Marius Brezovan, Sinaia, Romania, pp. 795–800.

[2] Liviu Lalescu, Volker Dirr, FET Free Timetabling Software, `http://www.lalescu.ro/liviu/fet/`.

[3] Timothy A. Redl, On Using Graph Coloring to Create University Timetables with Essential and Preferential Conditions, `http://cms.uhd.edu/faculty/redlt/iccis09proc.pdf`.

[4] Karl Nachtigall, Jens Opitz (2007) A Modulo Network Simplex Method for Solving Periodic Timetable Optimisation Problems, *Operations Research Proceedings*, pp. 461–466.

[5] Paweł Myszkowski, Maciej Norberciak (2003) Evolutionary algorithms for timetable problems, *Annales UMCS Informatica AI 1*, pp. 115–125.

[6] Marcin Paprzycki (2003) Agenci programowi jako metodologia tworzenia oprogramowania, `http://www.e-informatyka.pl/wiki/Agenci_programowi_jako_metodologia_tworzenia_oprogramowania`.

[7] Fabio Bellifemine, Giovanni Caire, Giovanni Rimassa, Agostino Poggi, Tiziana Trucco, Elisabetta Cortese, Filippo Quarta, Giosue Vitaglione, Nicolas Lhuillier, Jereme Picault, Java Agent Development Framework, `http://jade.tilab.com/`.

[8] Maciej Gawinecki, Minor Gordon, Pawel Kaczmarek, Marcin Paprzycki (2005) The Problem of Agent-Client Communication on the Internet, Scalable Computing Practice and Experience, 6(1), pp. 111–123

[9] Joe Henry Obit, Dario Landa-Silva, Djamila Ouelhadj, Teong Khan Vun, Rayner Alfred (2011) Designing a Multi-agent Approach System for Distributed Course Timetabling, *Proceedings of the 2011 IEEE Hybrid Intelligent Systems Conference (IEEE-HIS)*, IEEE Press, Melacca Malaysia, pp. 103–108.

`http://www.dcs.kcl.ac.uk/staff/mml/publications/assets/aamas06.pdf`.

[10] Yan Yang, Raman Paranjape, Luigi Benedicenti (2006) An Agent Based General Solution Model For

the Course Timetabling Problem, *AAMAS '06 Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, ACM New York, New York, USA, pp. 1430–1432 .

[11] Curak Ivan (2008) Negotiatian–based multi-agent system for timetabling, *Annals of DAAAM & Proceedings*, DAAAM International Vienna, `http://www.freepatentsonline.com/article/Annals-DAAAM-Proceedings/225316139.html`.

[12] Roxana A. Belecheanu, Steve Munroe, Michael Luck, Terry Payne, Tim Miller, Peter McBurney, Michal Pechoucek (2006) Commercial Applications of Agents: Lessons, Experiences and Challenges, *AAMAS '06 Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, ACM New York, NY, USA, pp. 1549–1555.

[13] Andrey Glaschenko, Anton Ivaschenko, George Rzevski, Petr Skobelev (2009) Multi–Agent Real Time Scheduling System for Taxi Companies, *The Eighth International Conference On Autonomous Agents And Multiagent Systems*, Budapest, Hungary, `http://www.ifaamas.org/Proceedings/aamas09/pdf/03_Industrial_Track/13_70_it.pdf`.

[14] Anja Zöller, Lars Braubach, Alexander Pokahr, Franz Rothlauf, Torsten O. Paulussen, Winfried Lamersdorf, Armin Heinzl (2006) Evaluation of a Multi–Agent System for Hospital Patient Scheduling, *International Transactions on Systems Science and Applications*, SpringerOpen, pp. 375–380.

[15] Carlos A. Coello Coello, Daniel Cort's Rivera and Nareli Cruz Cort's (2003) Use of an Artificial Immune System for Job Shop Scheduling, *Second International Conference on Artificial Immune Systems (ICARIS'2003)*, Springer-Verlag, pp. 1–10.

[16] Muhammad Rozi Malim, Ahamad Tajudin Khader, Adli Mustafa (2006) Artificial Immune Algorithms for University Timetabling, *6th International Conference on the Practice and Theory of Automated Timetabling*, Czech Republic.

[17] A. R. Mushi (2006) Tabu search heuristic for university course timetabling problem , *African Journal of Science and Technology (AJST) Science and Engineering Series Vol. 7, No. 1*, pp. 34–40.

[18] Houssem Eddine Nouri, Olfa Belkahla Driss (2013) Tabu search heuristic for university course timetabling problem , *African Journal of Science and Technology (AJST) Science and Engineering Series Vol. 7, No. 1*, pp. 34–40.

[19] Houssem Eddine Nouri, Olfa Belkahla Driss (2013) Distributed model for university course timetabling problem, *International Conference on Computer Applications Technology (ICCAT)*, Tunisia.

[20] Yan Yang, Raman Paranjape, Luigi Benedicenti, Nancy Reed (2005) A mobile agent system for university course timetabling, *Indian International Conference on Artificial Intelligence (IICAI-05) Pune*, India.

[21] Brian Autry, Kevin Squire (2008) University course timetabling with Probability Collectives, *The 7th International Conference on the Practice and Theory of Automated Timetabling*, Canada.

[22] Rafał Tkaczyk (2013) AgentPlanner – agentowy system zarzładzania planem zajeć, *University of Gdańsk, Gdańsk, Poland*, pp. 33–36.

[23] Foundation for Intelligent Physical Agents, FIPA Services Technical Committee, `http://www.fipa.org/activities/services.html`.

[24] Oracle Corporation, MySql Website, `http://downloads.mysql.com`.

[25] Oracle Corporation, NetBeans Website, `https://netbeans.org/downloads/7.0.1`.

# Jason Interpreter, Enterprise Edition

Dejan Mitrović and Mirjana Ivanović
Department of Mathematics and Informatics, Faculty of Sciences
University of Novi Sad, Novi Sad, Serbia
E-mail: {dejan, mira}@dmi.uns.ac.rs

Rafael H. Bordini
Postgraduate Programme in Computer Science – School of Informatics (FACIN)
Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, RS, Brazil
E-mail: r.bordini@pucrs.br

Costin Bădică
Computer and Information Technology Department
Faculty of Automatics, Computers and Electronics, University of Craiova, Romania
E-mail: cbadica@software.ucv.ro

**Keywords:** intelligent agents, Java EE, Jason, AgentSpeak

**Received:** July 24, 2015

*The Enterprise edition of the Java platform has been endorsed by both small and large enterprises, as it enables the development of large-scale, reliable, and secure software solutions. In the world of agent development, on the other hand, AgentSpeak, and its practical interpreter Jason, represent one of the most popular tools for writing complex, reasoning agents. This paper presents a framework that integrates the two approaches to distributed software development, and supports a seamless deployment of Jason agents in enterprise environments. The proposed framework offers many technical advantages, including automated agent load-balancing and fault-tolerance. The end-goal of this research, however, is to try and bridge the gap between the agent technology and modern enterprise applications.*

*Povzetek: Predstavljena je izpopolnjena platforma za agente v Jasonu z namenom izdelave agentnih aplikacij.*

## 1 Introduction

*Java platform, Enterprise Edition* (Java EE), is designed to support the development of scalable, secure, and reliable software products [11]. It is built around the idea of code reuse, and incorporates many libraries, frameworks, and technical solutions. As such, Java EE is often utilized as the main software development platform by small and large enterprises.

When it comes to agent development, most existing multiagent frameworks are written using the *Standard Edition* of Java (Java SE) [6]. On the other hand, as discussed in [14, 24], the use of Java EE can significantly reduce the effort needed to develop efficient multiagent frameworks. In addition, it can simplify the process of integrating agents into enterprise applications.

*Extensible Java EE-based Agent Framework* (XJAF) [16, 14, 24] is a multiagent framework built on top of Java EE. It utilizes technical solutions of Java EE in order to support scalable and reliable multiagent systems. More concretely, XJAF runs on top of computer clusters in order to provide *high-availability* of deployed applications, which is concerned with scalability and uninterrupted delivery of services, i.e. regardless of software or hardware failures [25].

Although the Java programming language is well-suited for many scenarios, the process of writing complex, reasoning agents often requires a special, *agent-oriented* programing language (AOPL) [6]. An AOPL provides programming constructs that enable developers to apply and use advanced multiagent concepts in practice. One of the most popular AOPLs is *AgentSpeak*, which directly supports the popular *Belief-Desire-Intention* agent architecture [21, 19]. To a great degree, the language owes its popularity to the *Jason* interpreter [4, 5]. Jason is a practical and efficient Java-based interpreter for an extended version of AgentSpeak, with a highly-customizable architecture.

This paper presents our most recent research efforts aimed at extending Jason with the support for enterprise environments. The new edition of Jason, named *Jason Interpreter, Enterprise Edition* (Jason EE), is integrated into XJAF, and uses its agent-oriented abstractions of Java EE technologies. This integration results in several advantages. First and foremost, being enterprise components themselves, Jason EE agents can interact with other parts of regular enterprise applications in a straightforward fashion. For example, a Jason EE agent can easily interact with web services or expose its capabilities in form of a web service, manage data in a remote (relational or NoSQL) database, etc. This integration could, therefore, help bridging the gap between agent technology and business, enterprise applications.

On the technical side, Jason EE provides agent load-balancing, thread pooling, and fault-tolerance. Load-balancing is concerned with automatic distribution of agents across the computer cluster [15]. It spreads the computational load, and enables Jason EE to run large number of agents. Thread pooling stems from the use of XJAF as the underlying multiagent framework. In XJAF, there is no thread-to-agent mapping [16]. An agent is assigned a thread whenever it needs to perform some processing. In the worst-case scenario, when all agents need to run simultaneously, there will be as many threads as there are agents, but the underlying enterprise application server tries to reduce resource consumption otherwise. As a result, XJAF and Jason EE can run many more agents on a single machine than a Java Virtual Machine (JVM) can support threads.

Finally, fault-tolerance is concerned with state replication and error recovery. It handles not only agents, but other Jason (EE) components as well, including *Execution control* and *Environment* [5]. Whenever a state of an object changes, it is replicated to a predefined number of nodes. In case the host node fails, the object is transparently restored on one of the remaining nodes and all calls to it are redirected there.

Along with these advantages comes one disadvantage. Since Java EE is more complicated than Java SE, the process of developing Jason EE applications is inherently more complicated, when compared to the process of developing standard Jason applications.

The idea of executing Jason agents in Java EE environments was originally presented in [13]. While it only dealt with mapping agents to *Enterprise JavaBeans*, this paper proposes Jason EE as a fully-featured redesign of Jason, suitable for enterprise environments.

The rest of the paper is organized as follows. Section 2 provides more details about AgentSpeak and Jason, as well as XJAF and *Enterprise JavaBeans*. Detailed insight into the Jason EE architecture and its components is given in Section 3. Section 4 presents a case-study that demonstrates one important technical advantage of Jason EE: state replication and failover. Related work is presented in Section 5, while the final conclusions and future research directions are given in Section 6.

## 2 Technology overview

In order to fully understand the architecture of Jason EE, some basic understanding of its underlying technologies is needed. This section describes AgentSpeak and Jason, as well as *Enterprise JavaBeans*, one of the core Java EE technologies, and XJAF, a multiagent framework which provides the necessary infrastructure for Jason agents.

### 2.1 AgentSpeak and Jason

The syntax of AgentSpeak is strongly inspired by Prolog. Its main data type is the *term*, which can be a constant (an atom, a number, or a string), a variable, a structure, or a list [7, 5]. However, AgentSpeak also includes a variety of new syntactical (and semantical) elements, in order to simplify the development of goal-oriented reasoning agents.

AgentSpeak agents are defined in terms of their beliefs, goals, and plans [4, 5]. The agent's belief base consists of *predicates* and *rules*. For example, the predicate `ball(8,32)` might represent a belief of a football playing agent that the ball is at position (8,32). The rule `canKick :- me(X,Y) & ball(A,B) & dist(X,Y,A,B) < 1` might indicate that the agent can kick the ball if the distance between itself and the ball is less than some predefined value.

More information about a belief can be provided using *annotations*. Annotation is a user-defined or a built-in structure attached to the predicate. For example, `ball(8,32)[source(percept)]` indicates that the agent's belief about the ball's current position stems from the perceptual information (i.e. the agent has directly observed the ball).

AgentSpeak supports two types of negations: *strong negation*, and *negation as failure* [5]. The first type, denoted by ~, indicates that the agent explicitly believes something no to be true. In the second type, a belief preceded by the `not` operator is true if it cannot be deduced from the agent's belief base.

The language supports two types of goals: *achievement* and *test* [4, 5]. Achievement goals are expressed as logical formulae describing the state of affairs the agent would like to reach. Test goals, on the other hand, are typically used to query the belief base, and determine if certain beliefs exist.

Beliefs and goals together describe the agent's mental state. Changes in the mental state, i.e. additions or removals of beliefs and goals, trigger the execution of *plans*. A plan definition consists of a *triggering event*, a *context*, and the plan *body* [5]. Whenever the agent's mental state changes, all plans with the corresponding triggering event are marked as *relevant*. The context of each relevant plan is then evaluated in order to determine if the plan is *applicable*. The context is a logical expression which describes beliefs, e.g. about the environment, that the agent must hold, and is especially useful in dynamic environments. The final element of a plan, its body, is defined as a sequence of simple logical expressions, internal or environment actions, test or achievement goals, as well as mental notes, which add new beliefs to the belief base [5].

Jason interpreter operates in *reasoning cycles*, where each cycle consists of ten steps [5]. First, the agent perceives its environment, processes a single message received from another agent, filtering out "socially unacceptable" messages along the way, and updates its belief base accordingly. The remaining six steps represent the core of agent's reasoning and acting:

– A single event is selected to be processed;

– A set of relevant plans, i.e. plans corresponding to that event, is selected;

– Of those, a set of applicable plans (or, *options*) is determined;

– Committing to an applicable plan, creating an *intention*;

– Selecting an intention from a stack of pending intentions; and

– Executing one step of the selected intention.

Users can modify the interpreter's behavior in many of these steps. For example, the applicable plan will be selected (for the agent to commit to executing it) based on its order in the source file (similarly as in Prolog). This behavior can be changed by modifying the corresponding selection function.

The work presented in this paper deals with a different aspect of modifying Jason. The goal is to enable AgentSpeak/Jason agents to operate in enterprise applications, by providing customized multiagent infrastructure, agent architecture, execution control, and environment, as discussed in Section 3.

## 2.2 Enterprise JavaBeans

*Enterprise JavaBeans* (*EJBs*, or simply, *beans*) represent one of the core Java EE technologies. They are server-side components that encompass the business logic of an application. An EJB is described as a *managed* component, as its life-cycle, concurrent access, transactional integrity, etc. is controlled by an enterprise application server.

There exist two main categories of EJBs: *message-driven* and *session*. Message-driven beans act as message receivers in the context of the *Java Message Service* (JMS)[1], an additional Java EE technology for asynchronous communication. Session beans are further classified as *singleton*, *stateless*, and *stateful*.

As its name suggests, there is only a single instance of a singleton session bean per Java Virtual Machine (JVM). Stateless session beans do not preserve conversational state with the client, and are best-suited for operations that can be performed in a single method call. Stateful sessions beans, on the other hand, are used when the client requires an ongoing, more complex conversation.

From the point of view of runtime efficiency, the best performance is achieved with stateless beans. When a client request is received, the enterprise application server can freely construct a new stateless bean on any cluster node. Once the request is handled, the instance is destroyed. Alternatively, instead of constructing and destroying stateless bean instances with each request, the server can be configured to recycle them from a *pool*. In any case, the advantage of using stateless beans is that it becomes relatively easy to implement load-balancing techniques in order to efficiently handle large numbers concurrent requests.

Although load-balancing of client requests that target stateful beans is also applicable, here, the more important focus is on *state replication and failover*. The server maintains multiple copies of a stateful bean across the cluster. In case of a node failure, client requests are transparently redirected to an instance residing in one of the remaining nodes. This functionality enables the development of highly-available systems, i.e. systems resilient to software and hardware failures.

As discussed next, session EJBs, mainly stateless and stateful, can be used to represent agents in enterprise applications. Stateless beans have a more restricted applicability, and are well-suited for "one-off" agents – agents that perform one task, and then terminate. More complex behaviors, like those exhibited by Jason agents, can be achieved by mapping agents to stateful beans.

## 2.3 XJAF

XJAF is an enterprise-scale multiagent framework [16, 24, 14]. One of the goals behind the development of XJAF was to show how Java EE technologies can be utilized to easily provide many functional requirements of multiagent platforms. By integrating enterprise technologies and agents, XJAF could assist in bridging the gap between the two approaches to distributed software development.

It is designed as an customizable architecture, in the sense that its core components, called *managers*, are recognized and used solely by their interfaces, while the implementation details can change. Each manager is in charge of handling a distinct part of the overall architecture. Over time, several managers have been in use, but the latest version [16] includes three: agent, message, and connection manager.

The agent manager acts as an agent directory and controls the agents' life-cycles. It represents each agent as an EJB component, and passes it to the underlying enterprise application server. The server is then in charge of managing the concurrent access, maintaining the state integrity of agents, fault-tolerance, etc. Internally, the agent directory is managed using the *Java Naming and Directory Interface* (JNDI)[2].

The message manager is in charge of transporting and delivering messages. By default, messages are processed asynchronously. Several utility functions, such as blocking the receiving of a message, are provided as well, in order to simplify the agent development. For most of its functionality, the message manager relies on the Java Message Service technology described earlier.

The latest version of XJAF, available at the XJAF home-

[1] `http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html`, retrieved on October 15, 2014.

[2] `http://docs.oracle.com/javase/tutorial/jndi/`, retrieved on October 15, 2014.

page[3], is focused on clustered computing and exploiting its many benefits [16]. It operates in *symmetric* clusters, where each node is connected to and aware of every other node. A single node is recognized as the *master*, while the others are called *slaves*. The only difference between the master and a slave is that the master can be used to remotely control the cluster: start or stop slaves, deploy applications, etc.

Finally, the connection manager is in charge of connecting physically distributed XJAF clusters in a single computational framework. Basically, it creates another virtual cluster formed of master nodes. As all parts of the XJAF, the connection manager relies on another Java EE technology for its functioning, *JGroups*[4].

In the context of this paper, XJAF acts as an underlying infrastructure for Jason EE agents. The details of integrating Jason and XJAF/Java EE concepts are described in details in the next section.

## 3 Jason EE

Jason is designed as a highly-customizable system. Users can modify not only the selection functions mentioned earlier, but also some of the interpreter's core components. The most important customizable components include *Agent architecture*, *Execution control*, and *Environment* [5]. Agent architecture represents the agent's "physical body" [5]. It enables the agent to perceive its environment and act upon it, and also to send and receive messages. The main function of Execution control is to synchronize the reasoning cycles of individual agents, while the Environment component serves as the basis for simulating real-world or artificial environments.

Another important concept in Jason is the *infrastructure* [5]. Infrastructure refers to a multiagent platform that actually hosts the agents, carries out the transmission and delivery of messages, etc. By default, two infrastructures are provided: *Centralised* and *JADE*. Figure 1 depicts main components of Jason and shows how the Centralised infrastructure binds them together[5].

Jason EE provides new versions of Jason's core components, and integrates them with an XJAF-based infrastructure, as described in the rest of this section.

### 3.1 Jason EE components

Unfortunately, Jason EE and Jason are not (yet) fully compatible, due to various technical and "philosophical" differences between Java EE and Java SE. For example, Java EE components should never directly create and use threads,

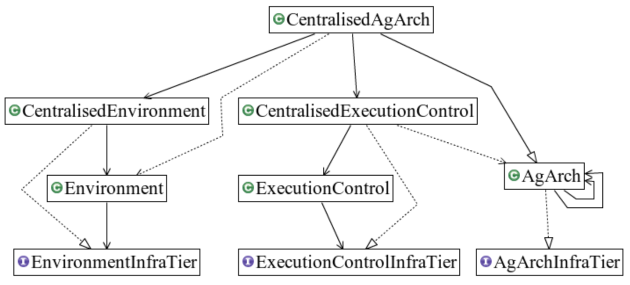


Figure 1: An overview of the main Jason components and the Centralised infrastructure that binds them together. The tight coupling and interdependencies of components might provide some difficulties in understanding the interpreter at first.

since these cannot be managed by the enterprise application server. In addition, the distributed nature of Java EE applications and the use of computer clusters pose additional requirements (e.g. component serialization).

With these differences in mind, the work of developing Jason EE consisted of three main tasks:

- Providing a new set of base classes for Agent architecture, Execution control, and Environment;
- Introducing a new set of components that support these base classes; and
- Integrating the developed architecture with XJAF, which acts as the underlying infrastructure.

The main components of Jason EE and their distribution are shown in Figure 2. The first noticeable difference from the (standard) Jason approach is in the way user-defined objects are loaded. By default, Jason relies on the Java *Reflection API* for user object construction. In case of Jason EE, however, this approach would not work. The enterprise application server views each user application as a distinct module, and loads it through a separate class-loader. This means that Jason EE components do not have direct access to classes defined in user applications. In order to resolve this issue, Jason EE introduces a new *Remote Object Factory* interface. The interface defines a set of methods that will be called whenever a user-defined object is to be created. Each user application needs to realize this interface in the form of a stateless session bean.

Agent architecture, Execution control and Environments are all deployed in the enterprise application server, and executed on top of a computer cluster. Since XJAF is used as the underlying infrastructure, the same approach of mapping agents to EJBs is used in Jason EE. This means that Jason EE agents exhibit the two important features described previously: automatic load-balancing, and state replication and failover.

[3] `https://github.com/gcvt/siebog`, retrieved on October 15, 2014.

[4] `http://www.jgroups.org/`, retrieved on October 15, 2014.

[5] All class diagrams in this paper have been generated from the source code of Jason 1.4.1 using *ObjectAid UML Explorer for Eclipse*: `http://www.objectaid.com/`, retrieved on September 19, 2014.

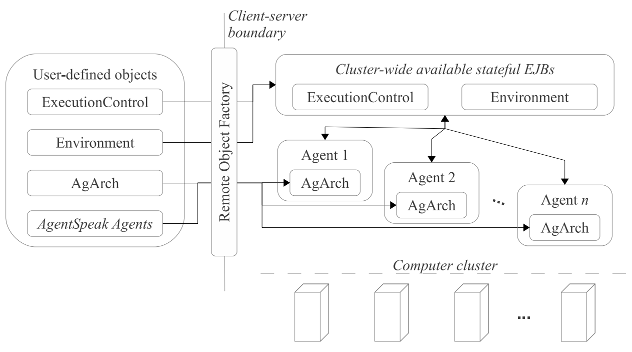


Figure 2: Organization and distribution of components in Jason EE-based applications.

## 3.2 Custom agent architectures

In Jason, users provide custom implementations of the Agent architecture by inheriting the base `AgArch` class (Figure 1). Some of its more important modifiable methods include:

- `perceive`: Perceives the environment, and returns the list of percepts;
- `act`: Performs the given action. The action does not have to be completed in the same method call, and the feedback (i.e. success or failure information) can be provided in one of the future reasoning cycles.
- `checkMail`, `sendMsg` and `broadcast`: Used for inter-agent communication.
- `sleep` and `wake`: Suspend and resume the agent's execution.

Jason EE provides a slightly different, intermediary class for the Agent architecture, in order to accommodate for the enterprise setting. The new class, `JasonEEAgArch`, modifies parts of the base `AgArch` class that are in charge of interaction with other components. For example, Execution control is an EJB in Jason EE, and the process of constructing and using EJBs is slightly different than constructing and using regular objects.

In order to connect the new Agent architecture class with the XJAF-based infrastructure, an additional component is provided. As shown Figure 2, `JasonEEAgArch` is embedded within an XJAF agent. The agent controls the architecture's life-cycle and also acts as a layer between the architecture and the remaining parts of the system. For example, it translates XJAF's FIPA ACL message format into Jason's KQML, but also controls the architecture's reasoning cycles, either directly in asynchronous, or indirectly in synchronous execution mode, as described later.

Jason EE agents are represented as EJB components, and are created through JNDI lookups. During the lookup phase, the enterprise application server will choose a node in the cluster to host the agent instance. From then on, the agent will have an affinity to that node, meaning that all invocations will be executed on it. However, whenever the agent's internal state changes, it will be copied to a predefined number of other nodes in the cluster. In case the current host fails, the agent will be restored on one of the remaining nodes, and continue its execution there.

Unfortunately, the base `AgArch` class in Jason, which is also used in Jason EE, is not fully serializable. This means that the agent's internal state, including, for example, the *transition system*, cannot be fully replicated across cluster nodes. If the agent's host fails, the agent will be transparently restored on one of the remaining nodes, but some of its parts will need to be re-initialized. The support for full state replication would require changes in the Jason interpreter itself. More details on this issue are given in Section 4.

## 3.3 Execution control

Jason supports two execution modes [5]: asynchronous and synchronous. In asynchronous mode, the agent executes its reasoning cycles continuously, regardless of the behavior of other agents. In synchronous mode, the agent can continue to the next reasoning cycle only after all other agents have completed the *current* reasoning cycles as well.

The execution control component is used in synchronous mode only. It maintains the list of active agents, and instructs them when to advance to the next reasoning cycle. This happens either when all agents complete the current reasoning cycle, or after a specified amount of time has passed (e.g. in case an agent has died unexpectedly). Users can provide their own, custom execution modes by inheriting the appropriate base class.

Having one central component that manages the execution of other components might not seem as a good design approach for distributed systems, such as Jason EE. However, the Execution control component can be thought of as a *synchronization barrier*. Barriers represent efficient synchronization approach when distributed or parallel processes need to operate in global computational steps [22]. In synchronous execution mode, Jason (EE) agent reaches the barrier after completing one reasoning cycle, and then waits until other agents have reached the barrier too.

The sequence diagram shown in Figure 3 outlines the execution of agent's reasoning cycles in Jason EE. Once the agent is created, it registers itself with the Execution control component. Subsequently, it will receive a signal to advance to the next reasoning cycle. When finished, the agents reports back to the Execution control. Once the condition is met (i.e. all agents have reported back, or the timeout has expired), the process is repeated.

The main technical difficulty in developing this Execution control for Jason EE is to properly design it for for clustered environments. That is, there should be a single instance of the component for the entire cluster, it should be easily accessible from any node, and should preferably exhibit state replication and failover. The singleton session

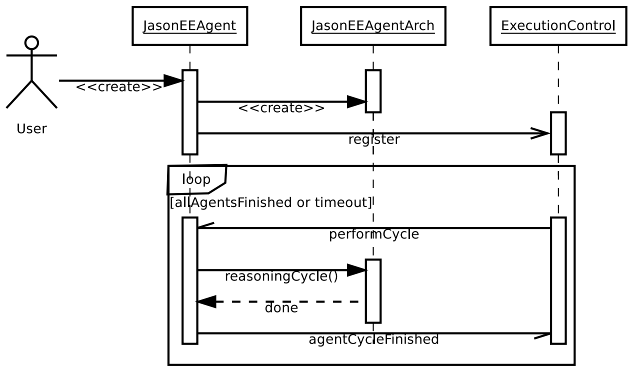


Figure 3: Synchronous execution of an agent managed by the Execution control.

EJB cannot be used, as there exists a single instance of this object per JVM.

The solution used in Jason EE is to create a single stateful EJB component and store it in a global, cluster-wide *Infinispan* cache [12] included in the *WildFly* enterprise application server[6]. As shown earlier in Figure 2, this Execution controls runs, figuratively speaking, on top of the entire cluster, supporting the aforementioned features.

End-users are offered an additional class, named `UserExecutionControl`, for customization. The Jason's standard class for this purpose, `ExecutionControl` (Figure 1), cannot be used in Jason EE, for two main reasons. First, the class is not serializable, and thus cannot be used in the state replication process. Secondly, the class creates and manages its own threads. Java EE applications are managed by the enterprise application server. The server needs the full control over the application's resources in order to secure scalability and high-availability. If an application creates its own threads, the server looses this control and the whole concept is undermined[7]. For these reasons, Jason EE provides the new base class, i.e. one that satisfies all the requirements and recommendations for Java EE applications.

## 3.4 Environment

The Environment component in Jason provides a model of a real-world or artificial environment [5]. It is strongly related to the Agent architecture, in the sense that the architecture can delegate perception and action execution to the Environment component. In addition, the Environment can exhibit "individualized perception" [5], and provide only a subset of percepts to an agent. This can be useful, for example, in evaluating how the agent performs under varying degrees of available information.

[6]`http://wildfly.org`, retrieved on October 15, 2014.

[7]The Java EE 7 specification defines so-called *managed executor services* which should be used by applications that need to spawn their own threads.

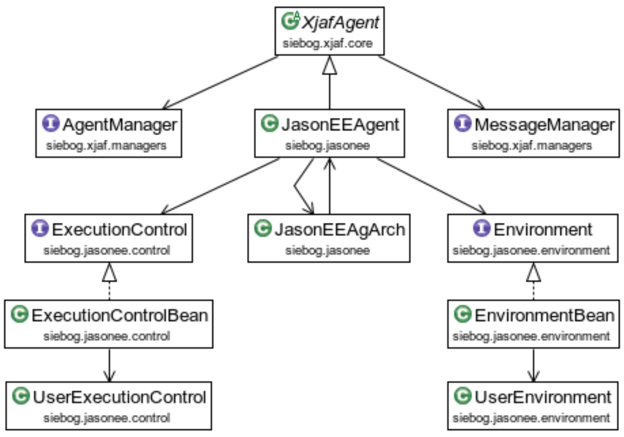


Figure 4: Diagram of the core classes in Jason EE. The connection to standard Jason is available through `JasonEEAgArch`, `ExecutionControl` and `Environment`, which are sub-types of, respectively, Jason's `AgArch`, `ExecutionControlInfraTier`, and `EnvironmentInfraTier` (not shown here for clarity).

Users provide custom environments by redefining methods of the base `Environment` class (Figure 1). All of its methods are dedicated to perception management (retrieval, removal, etc.), as well as action execution. For example:

– `getPercepts`: Returns the list of percepts for the given agent. The list will include only new percepts, i.e. percepts that have been obtained since the previous invocation.

– `scheduleExecution`: Performs an asynchronous execution of the provided action in the environment. The agent will be notified of the result later on, once the execution is completed, either successfully or unsuccessfully.

From the technical side, the Jason EE Environment is realized in a similar way as the Execution control component: in the form of a cluster-wide stateful EJB, with an additional `UserEnvironment` class for user customizations. The final class diagram of Jason EE is shown in Figure 4. Each deployed Jason EE application includes a single Environment instance, which is given a cluster-wide unique identifier and kept in a global cache. All agents of that particular application use the identifier in order to interact with the Environment. The same is true for Execution control.

The connection with XJAF is made through the `JasonEEAgent` component. Being a special XJAF agent, this component relies on the agent manager for creating and destroying other (Jason) agents, and on the message manager for sending and receiving messages.

The effectiveness of the proposed Jason EE architecture is demonstrated using two case-studies, described in details in the following section.

## 4 A case study

The case study presented in this section demonstrates state replication and failover in practice. Its purpose is to outline one of the technical advatages brought by Jason EE. The benefits of using AgentSpeak and Jason for complex agent development are presented in e.g. [23]. For XJAF agent load-balancing in computer clusters, see [16]. The full source of this case study is available at the XJAF homepage[8].

The case-study includes two highly-available agents. Each agent has a single belief: a list of strictly monotonically increasing numbers. The agent periodically prepends a number $n = h + 1$ to the list, where $h$ is the current head. Listing 1 shows the AgentSpeak source code of the agent.

Listing 1: AgentSpeak source code of the highly-available agent used in the case study.

```
numbers([0]). // inital belief
!addNextNum. // initial goal

+!addNextNum : true <-
 ?numbers([OldHead | Tail]);
 NewHead = OldHead + 1;
 NewList = [NewHead, OldHead | Tail];
 -+numbers(NewList);
 printList(NewList);
 !!addNextNum.
```

The case-study also includes a user-defined Agent architecture and a user-defined Execution control. The architecture is capable of executing the `printList` action shown in Listing 1, while the user-defined Execution control simply outputs the current reasoning cycle. These components were developed primarily in order to show how the state replication and failover work with other Jason EE components as well (and not just agents).

A cluster with two virtual nodes, a master and a slave, was setup, and the load-balancer was configured to put both agents and the Execution control component on the slave node. The project was then executed, yielding in the following possible output (filtered for clarity):

```
Cycle 0 on node slave@192.168.213.129
...
agent0 on slave@192.168.213.129: [1,0]
agent1 on slave@192.168.213.129: [1,0]
Cycle 7 on node slave@192.168.213.129
...
agent0 on slave@192.168.213.129: [2,1,0]
agent1 on slave@192.168.213.129: [2,1,0]
Cycle 15 on node slave@192.168.213.129
...
```

The slave node was then forcibly terminated. In response, all components from the slave have been automatically restored on the master node, and continued to operate as follows:

```
Cycle 63 on node master:xjaf-master
...
agent0 on master:xjaf-master: [1,0]
agent1 on master:xjaf-master: [1,0]
Cycle 68 on node master:xjaf-master
...
```

Here, it can be seen that both the agents and the Execution control have successfully continued their execution on the remaining node, confirming that the state replication and failover in Jason EE works as expected.

However, while the Execution control's internal state was successfully restored, the belief base of each agent has been reset. As noted earlier, some of the important Jason components, such as the Agent architecture and its transition system are not serializable. Since these are used in Jason EE as well, the agent's internal state cannot be fully replicated. Currently, Jason EE detects this issue and emits a warning, but the full support for the agent state replication requires changes in the Jason implementation.

## 5 Related work

In general, there are two main approaches for writing software agents. The first one is to use an existing programming language, such as Java. For example, in JADE the process of writing agents consists of inheriting the proper base classes [1]. More recently, source code annotations have been proposed as a convenient approach for developing BDI agents [18]. Extensions to the Java programming language have been proposed as well [26]. Main advantages of these approaches are a flatter learning curve and the availability of existing programming libraries and tools. The main disadvantage is that the agent source code is "cluttered" with object-oriented programming constructs.

The second approach is to use dedicated, agent-oriented programming languages. These languages offer programming abstractions that enable straightforward implementations of advanced multiagent concepts, and hide the overall complexity of writing intelligent agents. As a result, agent developers can focus on solving the problem in question, rather than dealing with class inheritance and method overriding.

Over time, a plethora of agent-oriented programming languages has been developed [6, 3]. Among the most recent is ASTRA[9], which combines AgentSpeak and *Teleo-Reactive* functions [9, 17]. Among the well-established languages, besides AgentSpeak, the two notable examples are *Goal-Oriented Agent Language* (GOAL)[10], and *A Practical Agent Programming Language* (2APL) [8].

In addition to achievement, GOAL adds support for *maintenance* goals [10]. An agent uses maintenance goals to refrain itself from acting, and to keep the current state of affairs. Agents generally follow the *blind commitment*

[8]`https://github.com/gcvt/siebog`, retrieved on October 15, 2014.

[9]`http://www.astralanguage.com/`, retrieved on October 15, 2014.

*strategy* [10, 20]: an active goal will not be dropped until it is fully completed. Actions are mostly user-provided, and are guarded by preconditions and postconditions. Action execution strategy is determined by *action rules*. An action rule defines a mental state that has to hold before the corresponding action can be considered as a candidate for execution.

2APL combines declarative and imperative programming styles [8]. It offers several important agent-oriented programming concepts, including beliefs, goals, events, actions, and plans. Belief and goals are declarative constructs that describe the agent's mental state. Events carry information about some change in the environment, and can trigger the plan execution. Actions describe agent's capabilities, and are divided into six categories, including *belief updates*, *goal updates*, and *abstract actions* which act as procedure calls. Finally, a plan consists of a sequence of actions, with the addition of conditional statements, loops, and non-interleaving operators for building atomic plans.

Each of these languages represents a powerful tool for developing intelligent agents. One of the main reasons AgentSpeak was selected for the work presented in this paper is its practical interpreter Jason. As shown, Jason is highly customizable and portable, allowing AgentSpeak agents to be executed on different multiagent platforms and environments. It is worth noting that Jason is not the only interpreter for AgentSpeak. For example, *AF-AgentSpeak* is an implementation of the language for the versatile *Agent Factory* framework[10].

In addition to Jason EE, two Jason infrastructures are available. The Centralised infrastructure is a lightweight and an efficient (performance-wise) solution, but designed for single-machine deployments only. The JADE infrastructure uses JADE as the underlying multiagent platform. Therefore, it provides all the features available in JADE, including distributed execution and platform fault-tolerance [1]. Jason EE implementation is to a certain degree based on these solutions. Its main advantages include state replication and failover demonstrated in Section 4, which is more advanced than the one offered by JADE, and automated agent clustering and load-balancing shown in [16].

## 6 Conclusions and future work

Jason EE represents an enterprise-scale agent development framework. It combines Java EE, one of the most widely-used software development platform, with AgentSpeak and Jason, a popular agent-oriented programming language and its interpreter, designed for writing complex, reasoning agents.

As discussed throughout the paper, Jason EE brings several important benefits over standard Jason, as well as other similar solutions. On the technical side, the underlying enterprise application server manages the application resources, and provides advanced programming features, such as agent load-balancing, scalability, and fault-tolerance. This enables agent developers to focus on solving the problem at hand, without having to bother with technical difficulties. The end-goal of the presented research, however, is to enable seamless integration of intelligent agents in modern enterprise applications, and to bridge the gap between the two approaches to distributed software development.

For the future, several research and development directions of Jason EE are planned. First of all, as discussed in Section 3, changes in the Jason implementation itself are required. These would allow not only the complete state replication and failover of agents, but also the full portability of Jason and Jason EE applications. In the longer run, the remaining two components of the *JaCaMo* project will be re-designed for enterprise environments: the *CArtAgO* artifacts modeling framework, and the *Moise* framework for virtual multiagent organizations [2].

[10] `http://www.agentfactory.com/`, retrieved on October 15, 2014.

## Acknowledgement

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, through project no. OI174023: "Intelligent techniques and their integration into wide-spectrum decision support."

## References

[1] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, 2007.

[2] O. Boissier, R. H. Bordini, J. F. Hubner, A. Ricci, and A. Santi. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761, 2013.

[3] R. H. Bordini, L. Braubach, M. Dastani, A. E. Fallah-Seghrouchni, J. J. Gomez-Sanz, J. Leite, G. M. P. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia)*, 30(1):33–44, 2006.

[4] R. H. Bordini and J. F. Hubner. BDI agent programming in AgentSpeak using Jason. In F. Toni and P. Torroni, editors, *Computational Logic in Multi-Agent Systems*, volume 3900 of *Lecture Notes in Computer Science*, pages 143–164. Springer Berlin Heidelberg, 2006.

[5] R. H. Bordini, J. F. Hubner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons Ltd, 2007.

[6] C. Bădică, Z. Budimac, H.-D. Burkhard, and M. Ivanović. Software agents: Languages, tools, platforms. *Computer Science and Information Systems*, 8(2):255–298, 2011.

[7] W. F. Clocksin and C. S. Mellish. *Programming in Prolog: using the ISO standard*. Springer, 5 edition, 2003.

[8] M. Dastani. 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.

[9] A. Dhaon and R. Collier. Multiple inheritance in AgentSpeak(L)-style programming languages. In *Proceedings of the 4th International Workshop on Programming based on Actors, Agents and Decentralized Control*. 2014.

[10] K. V. Hindriks. Programming rational agents in GOAL. In A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, editors, *Multi-Agent Programming: Languages, Tools and Applications*, pages 119–157. Springer US, 2009.

[11] Java EE at a glance. `http://www.oracle.com/technetwork/java/javaee/overview/index.html`. Retrieved on October 15, 2014.

[12] F. Marchioni and M. Surtani. *Infinispan data grid platform*. Packt Publishing Ltd., 2012.

[13] D. Mitrović, M. Ivanović, and C. Bădică. Jason agents in Java EE environments. In E. Petre and M. Brezovan, editors, *3rd Workshop on Applications of Software Agents (WASA 2013), held within 17th International Conference on System Theory, Control and Computing (ICSTCC 2013)*, pages 768–771, Sinaia, Romania, October 2013.

[14] D. Mitrović, M. Ivanović, Z. Budimac, and M. Vidaković. Supporting heterogeneous agent mobility with ALAS. *Computer Science and Information Systems*, 9(3):1203–1229, 2012.

[15] D. Mitrović, M. Ivanović, and Z. Geler. Agent-based distributed computing for dynamic networks. *Information Technology and Control*, 43(1):88–97, 2014.

[16] D. Mitrović, M. Ivanović, M. Vidaković, and Z. Budimac. Extensible Java EE-based agent framework in clustered environments. In J. Mueller, M. Weyrich, and A. L. C. Bazzan, editors, *12th German Conference on Multiagent System Technologies*, volume 8732 of *Lecture Notes in Computer Science*, pages 202–215. Springer International Publishing, 2014.

[17] N. J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158, 1994.

[18] A. Pokahr, L. Braubach, C. Haubeck, and J. Ladiges. Programming BDI agents with pure Java. In J. P. Müller, M. Weyrich, and A. L. C. Bazzan, editors, *Multiagent System Technologies*, volume 8732 of *Lecture Notes in Computer Science*, pages 216–233. Springer International Publishing, 2014.

[19] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. V. de Velde and J. Perrame, editors, *Agents Breaking Away: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world (MAAMAW '96)*, volume 1038 of *Lecture Notes in Artificial Intelligence*, pages 42–55. Springer-Verlag, 1996.

[20] A. S. Rao and M. P. Georgeff. Intentions and rational commitment. Technical Report 8, Australian Artificial Intelligence Institute, 1993.

[21] A. S. Rao and M. P. Georgeff. BDI agents: from theory to practice. In V. Lesser and L. Gasser, editors, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 312–319, 1995.

[22] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[23] S. Vester, N. S. Boss, A. S. Jensen, and J. Villadsen. Improving multi-agent systems using Jason. *Annals of Mathematics and Artificial Intelligence*, 61(4):297–307, April 2011.

[24] M. Vidaković, M. Ivanović, D. Mitrović, and Z. Budimac. Extensible Java EE-based agent framework – past, present, future. In M. Ganzha and L. C. Jain, editors, *Multiagent Systems and Applications*, volume 45 of *Intelligent Systems Reference Library*, pages 55–88. Springer Berlin Heidelberg, 2013.

[25] WildFly 8 high availability guide. `https://docs.jboss.org/author/display/WFLY8/High+Availability+Guide`. Retrieved on October 15, 2014.

[26] M. Winikoff. Jack intelligent agents: An industrial strength platform. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 175–193. Springer US, 2005.

# Expressing GMoDS Models into Object-Oriented Models Using the Event-B Language

Marius Brezovan, Liana Stanescu and Eugen Ganea
University of Craiova, Romania
E-mail: {mbrezovan, lstanescu, eganea}@software.ucv.ro

**Keywords:** multi-agent systems, goal-oriented specification, object-oriented specification, Event-B

**Received:** July 24, 2015

*Among the agent-oriented methodologies that use goals for specification of multi-agent systems, the Goal Model for Dynamic Systems (GMoDS) method allows to specify goals during requirements engineering process and then to use them throughout the system development and at runtime. Because the semantics of the GMoDS models involves the use of object-oriented concepts we choose to express a GMoDS model in an object-oriented specification. We use Event-B as a method for both specifying the GMoDS models and implementing the semantics of the runtime model of GMoDS. Because Event-B is not an object-oriented language, the goal of our research is to add support to Event-B for object-oriented modeling by using the modularization plug-in of the Rodin framework. This aim of paper is twofold: (a) to describe an object-oriented specification in Event-B, and (b) to express a GMoDS model into an object-oriented Event-B specification.*

*Povzetek: Razvit je agentni sistem z dodatnimi lastnostmi objektnih sistemom.*

## 1 Introduction

In recent years the domain of *multi-agent systems* (MAS) is perceived as generating a new paradigm in order to cope with the increasing need for dynamic applications that adapt to unpredictable situations. This new software engineering domain, *agent-oriented software engineering*, provides the tools and techniques to use in designing complex, adaptive systems.

Several frameworks for multi-agent system specification have been proposed to deal with the complexity of large software systems, such as Tropos [22], Gaia [5], MaSE [11], and ROADMAP [21]. To reduce the complexity of a correct and effective design for such systems, *Organization-based Multi-Agent Systems* (OMAS) have been introduced as an effective paradigm for addressing the design challenges of large and complex MAS [18]. In OMAS there is a clear separation between agents and system, allowing a reduction in the complexity of the system. To support the design of OMAS, several methodologies have been proposed [16].

Among these proposals, the *Organization-based Multi-agent Systems* (O-MaSE) methodology [12] seems to be the only framework which integrates a set of concrete technologies aimed at facilitating industrial acceptance through situational method engineering. In O-MaSE methodology, goals are specified using *Goal Model for Dynamic Systems* (GMoDS) [13], a methodology that provides a set of models for capturing system level goals, for using them during both the design and runtime phases, in order to allow the system to adapt to dynamic problems and environments.

The development of correct/safe complex MAS is difficult with traditional software development methods. Hence, formal methods are needed in order to ensure their correctness and structure their development from specification to implementation. To that end, formalization is needed, which has begun to receive a substantial amount of interest. Several approaches for formalizing MAS development are proposed. For instance, in [19] a general framework for modelling MAS based on Object-Z and statecharts is proposed, which focuses on organizational aspects in order to represent agents and their roles. Similarly, in [24] Z notations are combined with linear temporal logic to specify the internal part of agents and the specification of the communication protocols between agents. In [8], an approach based on capturing interaction protocols between requesters, providers and middle-agents as finite state processes represented using FSP process algebra is proposed, and the resulting specifications are formally verifiable using FLTL temporal logic.

However these approaches do not address the the problem of using formal methods within a well-defined MAS development methodology. This is the reason for our attempt to use the *Event-B* both as a method to specify the O-MaSE models, and as a tool to implement the semantics and the runtime model of O-MaSE. Event-B is a state-based formal method that supports a refinement process in which an abstract model is elaborated towards an implementation in a step-wise manner. In addition Event-B is proven to be applicable in a wide range of domains, including distributed algorithms and multi-agent system development. Its deployment is supported by the *Rodin* toolset, which includes proof obligation generation and verification through a collection of mechanical provers. *Rodin* was used in several academic and complex industrial size systems.

We started our research with the study of the GMoDS methodology, an important part of O-MaSE, by translating GMoDS models in object-oriented specifications in Event-B. GMoDS represents a framework for developing complex multi-agent systems using *goals* to capture requirements, the same set of goals being used for MAS design, and at runtime. In GMoDS, goals are organized in a *goal tree* such that each goal is decomposed into a set of sub-goals using *AND*/*OR* decomposition. Leaf goals are simple goals that must be achieved by agents. Within O-MaSE, each MAS contains a set of *roles* that it can use to achieve its goals. The roles for MAS can be derived from the goal tree, each leaf goal should have at least one role that can achieve it. For simplicity, we assume that is an one-to-one mapping between the set of goals and the set of roles. Each *agent* from a MAS is capable of playing at least one role, with the property that at every moment, an agent can have only one role. Thus, at every moment, an agent from MAS is related to a leaf goal from the goal tree of the GMoDS framework.

In GMoDS, there are two types of goals: *goal classes* and *goal instances*. Goal classes define templates from which goal instances are created. A goal class contains a set of goal attributes that are used to define the state od a goal instance. When a goal is instantiated, all its attributes must be given explicit values. While goal classes are used in the design process of MAS, the goal instances are used at runtime, or during a simulation process. Goal classes and goal instances are analogous to object classes and object instances from the object-orientation paradigm. This is the reason for using an object-oriented framework to specify the GMoDS models.

Event-B extended with several facilities, such as modularity, decomposition, the use of records and generic instantiation, shows a good potential for the use in the industrial practice. Unfortunately the Event-B language is not object-oriented because it does not support the main object-oriented concepts, such as inheritance, subtyping, class instantiation, calling of public methods of class instances, and polymorphism. Some approaches, such as records [17], modularisation [20], generic instantiation [26], and especially the *UML-B* method [27], bring closer Event-B to an object-oriented language. The UML-B graphical modelling notation provides four kind of diagrams: package, context, class and state machine diagrams. However, UML-B does not address some important object-oriented concepts, such as subtyping, polymorphism, and calling public methods of the class instances. Because GMoDS models involve the use of calling operations of some objects within the plans from the plan models, we use interfaces and modules from the modularisation plug-in of the Rodin framework, and the principles from the UML-B method for managing classes, class instances, class attributes and associations, in order to allow appropriate object-oriented specifications in Event-B. In addition we model in Event-B specifications other two main object-oriented concepts: *inheritance* and *polymorphism*. *Inheritance* is needed for creating dynamic trees of goal instances from the GMoDS runtime model, while *polymorphism* is needed for calling the appropriate operation, when a class hierarchy is used.

In conclusion, the aim of this paper is twofold: (a) to propose an extension of the Event-B method that allows the creation and destruction of class objects, as well the call of public methods of classes, inheritance, and polymorphism, as well as (b) to use this extension for translating GMoDS models into Event-B object-oriented specifications.

The rest of this paper is organized as follows. Section 2 presents the O-MaSE methodology framework, and its associated GMoDS methodology. Section 3 presents the main concepts of the Event-B method, and some of its extensions that will be used in the paper. Section 4 presents a proposal for constructing an object-oriented specification in Event-B that allows calling public methods of class instances. In Section 5 this proposal is used to express the main GMoDS models using object-oriented Event-B specifications. Finally, conclusions are given in Section 6.

## 2 O-MaSE and GMoDS methodologies

The Organization-Based Multiagent System Engineering [12] methodology extends the original MaSE [11] methodology to allow the design of organizational multi-agent systems. The definition of O-MaSE consists of three main components: the O-MaSE meta-model, method fragments, and guidelines.

The O-MaSE *Meta-Model* is based on an organizational approach, which extends the organization model for adaptive computational systems (OMACS) [10]. OMACS defines an organization as a set of *Goals* that the organization is attempting to accomplish, a set of *Roles* that must be played to achieve those goals, a set of *Capabilities* required to play those roles and a set of *Agents* who are assigned to roles in order to achieve organizational goals. The environment is modeled using the *Domain Model*, which defines the types of objects in the environment and the relations between them. In addition to OMACS, the O-MaSE meta-model adds new concepts, such as: *Plans* that capture algorithms that agents use to carry out specific tasks, *Actions* that allow agents to perceive or sense objects in the environment, *Organisational agents* that capture the notion of organizational hierarchy, and *Protocols* that define interactions between roles or between the organization and external actors. Figure 1 shows a simplified OMACS meta-model.

In a multi-agent organization (MAO), organizational goals are typically organized in a goal tree. In OMACS, and thus in O-MaSE, goals are specified using GMoDS

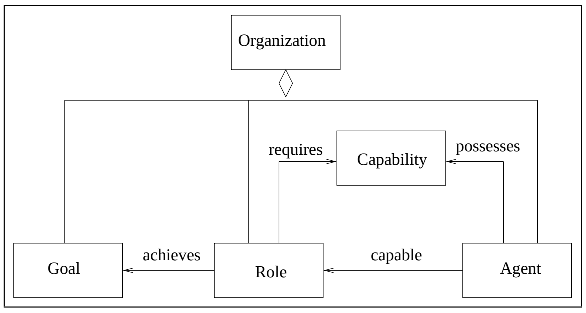


Figure 1: Simplified OMACS metamodel.

[13]. The GMoDS specification model includes the notions of goals, goal decomposition, events, precedence, and goal instantiation. The GMoDS instance model captures the dynamics of the system state while maintaining the structure of the specification model. The execution model implements these models in an efficient manner. The GMoDS specification model is used in the design process of a MAO, while the GMoDS instance and execution models are used in execution, or simulation processes of MAOs. Both in the design process, and in the execution process, the leaf goals are directly related to the agent plans. A GMoDS goal specification tree is presented in Fig. 2 (from [13]).

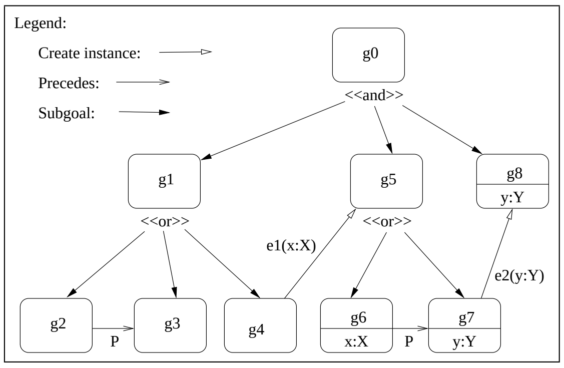


Figure 2: A GMoDS Goal specification tree.

A basic O-MaSE process is presented in Fig. 3.

A centralized *Organization-based agent architecture* is presented in Fig. 4 [12]. The *Control Component* contains *Goal Reasoning* and the *Reasoning Algorithm* that use specifications of the organisation goal, role, and agent models to perform reasoning about goals, and the assignment of agents to roles. The *Execution Component* contains the agents of the MAO specified by their roles and capabilities.

From the *Control Component*, *Goal Reasoning* is the module that implements the GMoDS framework. In this paper we describe a specification of the Goal Reasoning module using an object-oriented extension of the Event-B method.

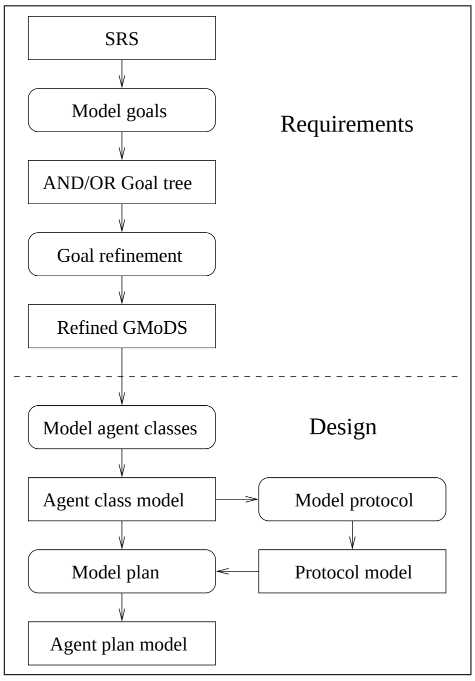


Figure 3: A basic O-MaSE Process.

## 3 Event-B method

*Event-B* [2] is a formal method for modelling concurrent systems by adopting a top-down development process. The Event-B method is influenced by the B Method [1] by using typed set theory as the mathematical language for defining state structures and events. However there is a conceptual difference between these two formal methods: while the B Method is aimed at software development, the Event-B is aimed at system development.

In order to support construction and verification of Event-B models, *RODIN*, an open toolset implemented on the top of the Eclipse platform, was constructed. The RODIN tool was initially developed as part of the European Union ICT Project RODIN (2004 to 2007) [25], and then continued by the EU ICT research projects DEPLOY (2008 to 2012) [14] and ADVANCE (2011 to 2014) [3]. The tool is implemented in Java and it uses several plug-ins that extend the basic functionality of the Event-B framework.

Event-B models are described in terms of two basic components: *contexts*, which contain the static part of a model, and *machines*, which contain the dynamic part. Contexts may contain *carrier sets*, *constants*, *axioms*, and *theorems*, where carrier sets are similar to types, while machines, which provide behavioral properties of Event-B models, may contain *variables*, *invariants*, *theorems*, and *events*. The state of a machine is described by its variables, which are constrained by invariants.

Each machine may contains a set of *events*, which describe possible state changes. Each event is a specialized B operation, and it is composed of a guard $G(t, v)$ and an action $A(t, v)$, where $t$ represents parameters the event may contain, and $v$ a subset of the variables of the machine. A

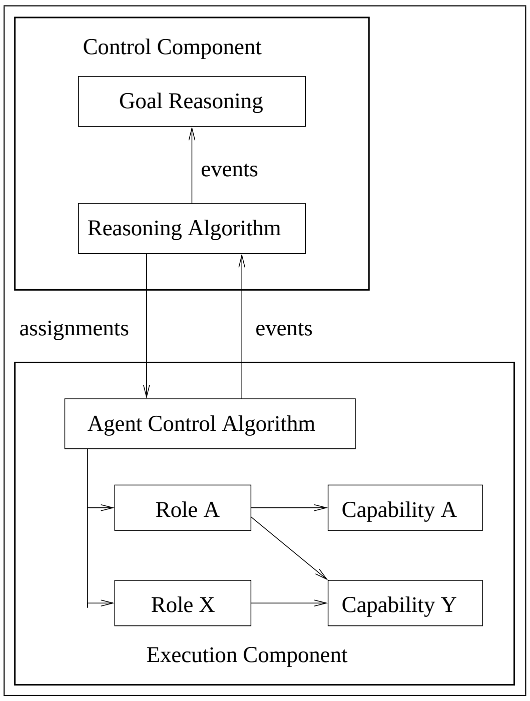


Figure 4: Organization-based agent architecture.

special event, *initialisation*, is used for describing the initial state of the machine. A machine can *see* multiple contexts. During the development, a context can *extend* one or more contexts by declaring additional carrier sets, constants, axioms or theorems.

The *refinement* is the only operation that can be applied to a machine. If a machine $N$ refines another machine $M$, then $M$ is called the *abstract machine*, while $N$ is a *concrete machine*. Event-B uses two principal types of refinement: superposition refinement [6] and data-refinement [7]. Superposition refinement corresponds to a spatial and temporal extension of a model, while data refinement is used in order to modify the state of the machine.

The Event-B language does not allow a modular development of a system. In order to manage this development method some plug-ins have been added to the RODIN platform. In the following we shortly present the *Modularisation* plug-ins that we use for constructing our proposal. The *Modularisation* plug-in allows a modular development of a specification by defining *modules* [20], a new type of Event-B components containing groups of callable operations. A module description consists of two parts, *module interface* and *module body*. A *module interface* is a separate Event-B component that consists of a set of external module variables ($v$), constants ($c$), and sets ($s$), the external module invariant, and a description of module operations, specified by their pre- and post-conditions. In addition, an interface can see its context. Denoting by $M$ a module, by $MI$ its interface, and by $MI\_ctx$ the context of $MI$, the interface $MI$ has a structure as follows:

```
**INTERFACE** $MI$
**SEES** $M\_ctx$
**VARIABLES** $v$
**INVARIANT** $M\_Inv(c, s, v)$
**INTIALISATION** $M\_Init(v)$
**OPERATIONS**
  $oper_1 \mathrel{\widehat{=}}$
    **ANY** $par_1$
    **PRE** $M\_Pre_1(c, s, par_1, v)$
    **RETURN** $res_1$
    **POST** $M\_Post_1(c, s, par_1, v, v', res'_1)$
    **END**
  ...
**END**,
```

where the primed variables of the interface and the variables representing the result of the operation, specified in the predicates representing the postcondition of the operation, stand for the variable values after operation execution.

A *module body* is an Event-B machine, where the operations specified in its interface are implemented. Each operation is implemented by a *group* of events, one group for each operation. Some events from a group play a special role of operation termination events and are called *final* events. A final event returns the control to a caller.

An *operation* defined into a module $M$ can be invoked into a Event-B machine, which can be another module, only if the module $M$ is *imported* into this machine. The inclusion of a module into a Event-B machine is specified by a clause **USES** in the importing machine. This clause specify the interface of the imported module, and a prefix that is used to emulate a dedicated namespace for the imported module. All the names of the imported module are modified by adding this prefix.

The syntax for an operation invocation is similar to a function call. The semantics of an operation invocation is also similar to the standard semantics of a function call as in the most programming languages. Because an operation invocation is atomic, the events from the group corresponding to the operation in the module body run until termination without interference from other groups.

## 4 Writing object-oriented specifications in event-B

Because B and Event-B methods are not object-oriented, there are several proposals in the last years to bring object-oriented concepts into these methods. First of all, both B and Event-B have only static structuring mechanisms: they allow to define abstract machines with a static architectural structure that do not change at run-time [15]. In [4] an extension of the syntax of B is proposed for supporting the management of dynamic populations of components. In this proposal a *population manager* is associated to a machine for managing its instances. Although this extension is not object-oriented, the population manager for a machine $M$ is a machine which represents a dynamic set of $M$ instances, including operations for the creation and deletion of machine instances.

A similar mechanism is used also in the UML-B method: for each machine representing a class hierarchy, an implicit context is generated, which defines the set of all instances, $A\_SET$, for each class $A$ from the class hierarchy. As opposed to the B method, where an abstract machine can represent a class, and a hierarchy of classes is constructed by several machines that use the clause **USES** to include other the classes from the hierarchy, the Event-B method does not have **USES** and **INCLUDE** clauses, thus a hierarchy of classes must be defined in a single machine. Another weakness of the UML-B method is the absence of the method calls of the class instances, because an Event-B machine has only events (or transitions in UML-B), not operations as in the case of the B abstract machines.

The aim of this Section is to bring some object-oriented concepts into Event-B modelling, without changing the syntax of Event-B, and thus allowing the Event-B specifications to be realized and verified with the *Rodin* tool. We do not use the UML-B method because of the weakness above mentioned, related to the absence of the method calls. In fact, we use the modularization approach [20] in order to allow this action, while preserving some object-oriented elements from the UML-B, such as management of class instances, attributes, associations, and inheritance.

We use *interfaces* for describing class hierarchies and the methods (operations) of the classes, and *modules* for implementing the class methods. For a hierarchy $H$ containing the classes, $A_1, \ldots, A_k$, the following Event-B components are defined:

- A context, $H\_Ctx$, which contain: the set $INST$ of all instances of all class from the $H$ hierarchy, the constant $Void$ representing the $null$ instance, and the set of all instances of the classes $A_1\_Inst, \ldots, A_k\_Inst$ respectively, with the property that:

$$INST = \bigcup_{i=1}^{k} A_i\_Inst \cup \{Void\},$$
$$A_i\_Inst \cap A_j\_Inst, \forall i \neq j.$$

- An interface, $H\_Intf$, having:
  - as variables, the sets $A_i \in \mathbb{P}(A_i\_Inst)$ representing the set of active objects of the class $A_i$, $i = 1, \ldots, k$, and relations and functions representing attributes of these classes and the associations between some classes,
  - as operations, the constructor and the destructor for each class, and other operations representing the methods of the classes $A_i$, $i = 1, \ldots, k$
- A module, $H\_Impl$, where the operations defined in $H\_Intf$ are implemented.

As an example, we consider two classes, $Node$ and $List$, where each list is an ordered sequence of nodes, and each node has as attribute with an integer value. The context related to classes $Node$ and $List$ is defined as follows:

```
CONTEXT $H\_Ctx$
SETS $INST$
CONSTANTS $Void, Node\_Inst, List\_Inst$
AXIOMS
  $Void \in INST$
  $Node\_Inst \subseteq INST$
  $List\_Inst \subseteq INST$
  $\mathrm{partition}(INST, \{Void\}, Node\_Inst, List\_Inst)$
END
```

From the interface $H\_Intf$, the variables, their invariants and initializations are defined as follows:

```
INTERFACE $H\_Intf$
SEES $H\_Ctx$
VARIABLES $Node, value, List, first, next$
INVARIANTS
  $Node \in \mathbb{P}(Node\_Inst \cup \{Void\})$
  $value \in Node\_Inst \rightarrow \mathbb{N}$
  $List \in \mathbb{P}(List\_Inst \cup \{Void\})$
  $first \in List\_Inst \rightarrow Node\_Inst \cup \{Void\}$
  $next \in List\_Inst \rightarrow (Node\_Inst \rightarrow (Node\_Inst \cup \{Void\}))$
INTIALISATION
  $Node := \varnothing, value := \varnothing$
  $List := \varnothing, first := \varnothing, next := \varnothing$
OPERATIONS
  ...
```

From the operations related to the $Node$ class we present only the constructor $newNode$, and the function $getValue$:

```
$newNode \mathrel{\widehat{=}}$
  ANY $self, v$
  PRE
    $self \in Node\_Inst \setminus Node$
    $v \in \mathbb{N}$
  RETURN $ret$
  POST
    $Node' = Node \cup \{self\}$
    $value' = value \cup \{self \mapsto v\}$
    $ret' = self$
  END
$getValue \mathrel{\widehat{=}}$
  ANY $self$
  PRE $self \in Node$
  RETURN $ret$
  POST $ret' = value(self)$
  END
```

From the operations related to the $List$ class we present only the the destructor $deleteList$ and the operation *insertFront*:

```
$deleteList \mathrel{\hat{=}}$
  ANY $self$
  PRE $self \in List$
  RETURN $ret$
  POST
    $first' = (\mathrm{dom}(first) \setminus \{self\}) \lhd first$
    $next'(self) = \varnothing$
    $List' = List \cup \{self\}$
    $\forall a, b \cdot a \in Node\_Inst \wedge a \in Node\_Inst \wedge$
        $a \mapsto b \in next(self) \Rightarrow a \notin Node'$
    $ret' = Void$
  END
$insertFront \mathrel{\hat{=}}$
  ANY $self,\ n,\ v$
  PRE
    $self \in List$
    $n \in Node\_Inst \setminus Node$
    $v \in \mathbb{N}$
  RETURN $ret$
  POST
    $value'(n) = v$
    $next'(self) = next(self) \cup \{n \mapsto first(self)\}$
    $first'(self) = n$
    $ret' = first'(self)$
  END
```

In the implementation module, *H_Impl*, each operation is implemented by a group containing one or more events. Other defined operations can be called in the action part of these events. For example, in the implementation of operation *insertFront*, the constructor *newNode* of the class *Node* is called:

```
$insertFront \mathrel{\hat{=}}$
  ANY $self, n, v$
  WHERE
    $self \in List$
    $n \in Node\_Inst \setminus Node$
    $v \in \mathbb{N}$
  THEN
    $n := newNode(v)$
    $next(self) := next(self) \cup \{n \mapsto first(self)\}$
    $first(self) := n$
    $insertFront\_ret := n$
  END
```

From all operations of the class *List*, only the operation *deleteList* has a group containing two events: *deleteListNonEmpty*, that occurs for each node in a non-empty list, and *deleteListEmpty* that occurs when the list is empty.

In order to describe the modeling of the inheritance and polymorphism concepts in Event-B, we use an example of a class hierarchy with three classes, $B$, $D1$ and $D2$, as in Fig. 5, where $D1$ and $D2$ inherit the class $B$. In addition, all three classes have the same operation, *op*.

Denoting with *INST* the set of all instances of the classes form the above hierarchy, with *B_Inst*, *B_Inst*, *D1_Inst*, and *D2_Inst* the set of all possible instances of the classes $B$, $D1$, and $D2$ respectively, the fact that $D1$ and $D2$ *inherit* the class $B$ can be described as follows:

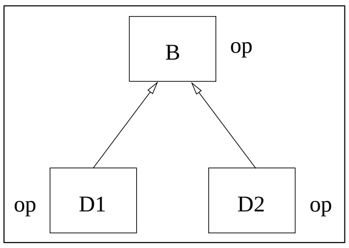


Figure 5: A class hierarchy with one class root.

```
CONTEXT $Ctx$
SETS $INST$
CONSTANTS $Void,\ B\_Inst,\ D1\_Inst,\ D2\_Inst$
AXIOMS
  $Void \in INST$
  $\mathrm{partition}(INST, \{Void\},\ B\_Inst,\ D1\_Inst,\ D2\_Inst)$
  $\mathrm{partition}(B\_Inst, D1\_Inst,\ D2\_Inst)$
END
```

The *polymorphism*, related to the operation *op* in this case, is modeled in the interface, $I$, which has only one operation, denoted by *op* in this case, and in its associated module, $M$, which contains a group with two different final events, denoted by $op1$ and $op2$ in this case, one event for each each operation from a inherited class.

```
INTERFACE $I$
SEES $Ctx$
VARIABLES $B,\ D1,\ D2$
INVARIANTS
  $B \in \mathbb{P}(B\_Inst \cup \{Void\})$
  $D1 \in \mathbb{P}(D1\_Inst \cup \{Void\})$
  $D2 \in \mathbb{P}(D2\_Inst \cup \{Void\})$
INTIALISATION
  $B := \varnothing,\ D1 := \varnothing,\ D2 := \varnothing$
OPERATIONS
$op \mathrel{\hat{=}}$
  ANY $self$
  PRE $self \in B\_Inst \setminus B$
  RETURN $ret$
  POST
    $B' = B \cup \{self\}$
    $self \in D1\_Inst \setminus D1 \Rightarrow D1' = D1 \cup \{self\}$
    $self \in D2\_Inst \setminus D2 \Rightarrow D2' = D2 \cup \{self\}$
    $ret' = self$
  END
```

The module $M$ can be described as follows:

```
MACHINE $M$
  IMPLEMENTS $I$
  SEES $Ctx$
  ...
  GROUP $op$ BEGIN
    FINAL $op1$ $\widehat{=}$
      ANY $self$
      WHERE
        $self \in D1\_Inst \setminus D1$
      THEN
        $D1 := D1 \cup \{self\}$
        $op1\_ret := self$
      END
    FINAL $op2$ $\widehat{=}$
      ANY $self$
      WHERE
        $self \in D2\_Inst \setminus D2$
      THEN
        $D2 := D2 \cup \{self\}$
        $op2\_ret := self$
      END
  END
END
```

## 5 Expressing GMoDS models in object-oriented specifications in event-B

As stated in Section 2, we describe a specification of the GMoDS framework using an object-oriented extension of the Event-B method, which represents the *Goal Reasoning* module from the *Control Component* of an *Organization-based agent architecture*.

The GMoDS definition contains three different models [13]: (i) a *Specification model* that contains a tree structure of *goal classes* and their associations, and (ii) a *Runtime model* that contains a tree structure of *goal instances* and the actions that are executed, each action being related to a association between classes, and (iii) an *Execution model* that implements GMoDS using and updating continuously a collection of sets of goal instances, according to the current state of each goal instance.

### 5.1 GMoDS Models

The *Specification model* of GMoDS contains the *goal specification tree*, $G_{Spec}$, which describes how the goal classes are related to one another, and where upper level goals (parents) are decomposed into lower level sub-goals (children) and each parent has either a conjunctive or disjunctive achievement condition as shown via the $\langle\langle and\rangle\rangle$ and $\langle\langle or\rangle\rangle$ decoration in Fig. 2. Goals without children are known as *leaf goals*.

In addition to goals, the specification model uses another concepts, such as, *relations*, *events*, and *parameters*. The main relation type used by this model is the *goal precedence*, specified by the $precedes$ relationship, that ensures that no agents work on a specific goal until all goals that precede that goal have been achieved. In Fig. 2 there are two $precedes$ relations: $precedes(g2, g3)$, and $precedes(g6, g7)$. Events in GMoDS are represented by *triggers*:

- a *positive trigger*, or simply $trigger$, which allows a new goal instance of a certain class $g_j$ to be created when and event $e_k$ occurs during the pursuit of a goal instance of a class $g_i$, eventually by passing some parameter values $p$. In Fig. 2 there are two triggers: $trigger(g1, e1, x) = \{g5\}$, and $trigger(g7, e2, y) = \{g8\}$.
- a *negative trigger*, or $\neg trigger$, which allows an active goal instance of a certain class $g_i$ to eliminate another active goal instance of a certain class $g_j$ from the set of *active goal instances* when an event $e_k$ occurs.

There is always an *initial trigger*, usually denoted by $e_0$, that is used when the system starts, which creates an instance of the root goal (and, recursively, it can create others goal instances).

The *Runtime model* is represented by a dynamic tree of goal instances, $G_{Inst}$ that retains the structure of $G_{Spec}$ while allowing dynamism by way of triggering and precedence. For each goal instance from $G_{Inst}$, four predicates are dynamically set:

- $achieved$, which determines whether a goal has been achieved by the system. For *leaf goals* $achieved$ becomes true when the agent pursing the goal notifies the system of its achievement, while for parent goals, the value of the $achieved$ is based on the achievement condition (conjunction or disjunction) and the state of its children.
- $obviated$, which states whether a goal is no longer needed by the system. A goal becomes obviated if it is a child of a disjunctive goal that has been achieved that does not precede any other system goal.
- $preceded$, which becomes $true$ if a goal preceding it has not been achieved, or if a new goal may still be instantiated that may precede it.
- $failed$, which becomes $true$ if the system has deemed that the goal can never be achieved by the system.

For example, after the *initial trigger*, the instance tree, $G_{Inst}$, has a structure as presented in Fig. 6. Instances of the goals $g5$ (and subsequently $g6$ and $g7$) and $g8$ are not created because they will be created (triggered) by $g4$ and $g7$ when the events $e1$ and $e2$ respectively will occur.

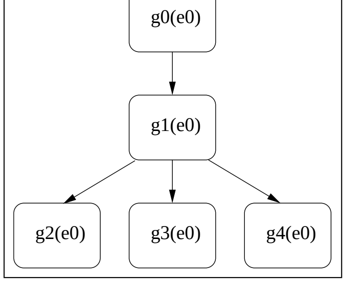


Figure 6: The tree $G_{Inst}$ associated to the tree $G_{Spec}$ from Fig. 2 after the initial trigger.

In the *Runtime model* there are maintained and updated six sets, $G_{I-Triggered}$, $G_{I-Active}$, $G_{I-Achieved}$, $G_{I-Removed}$, $G_{I-Failed}$ and $G_{I-Obviated}$ as shown in Fig. 7.

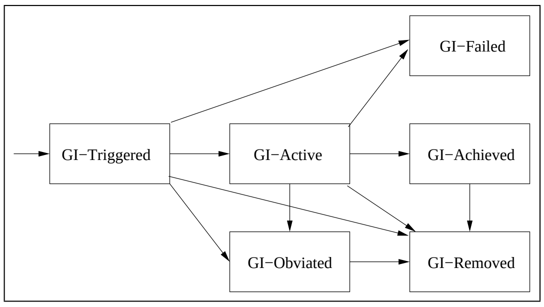


Figure 7: Goal execution model.

Each set contains current instance goals having the same state:

– $triggered$, for all instances created by a trigger event, or, recursively, by a parent goal,
– $active$, for all triggered instances that are not preceded,
– $obviated$, that is based on the $obviated$ predicate,
– $achieved$, that is based on the $achieved$ predicate,
– $failed$, that is based on the $failed$ predicate,
– $removed$, for all goal instances destroyed by a negative trigger.

When the state of a goal instance is one of the last three state, this goals remains in this state until the system stops.

## 5.2 Expressing GMoDS Models into an Object-Oriented Model

For specifying in Event-B the GMoDS framework (in fact the Goal Reasoning module), all the three GMoDS models must be specified. I3n the case of the Specification model, the goal tree $G_{Spec}$ is defined by using goal classes as nodes. All classes from a goal tree will form a hierarchy having an abstract class, denoted by $Goal$ as the root of this hierarchy. For the goal tree from the Fig. 2, the goal class hierarchy is presented in Fig. 8, where the classes $g0$, $g1$, . . ., $g8$ inherit the class $Goal$.

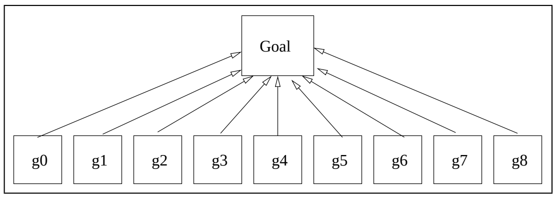


Figure 8: Goal class hierarchy.

The main two attributes of the class $Goal$ are $goal\_state \in Goal\_STATES$ and $goal\_type \in Gol\_TYPE$, where:

$$Goal\_STATES = \{triggered, active, achieved, failed, obviated, removed\},$$
$$Goal\_TYPE = \{AND, OR, LEAF\}.$$

In order to allow the specification of:

– the trigger events from the specification model,
– the predicates from the runtime model,
– the sets of goal instances, from the implementation model,

the following associations between goal classes are used:

– $down$ and $right$, which allows to specify the goal tree from $G_{Spec}$,
– $creates$, $created$ and $destroy$, which allow to specify the positive and negative triggers,
– $precedes$ and $preceded$, which allow to specify the precedence relation between goals,
– $up$, which allows to retrieve the parent of a goal.

These associations are presented in Fig. 9.

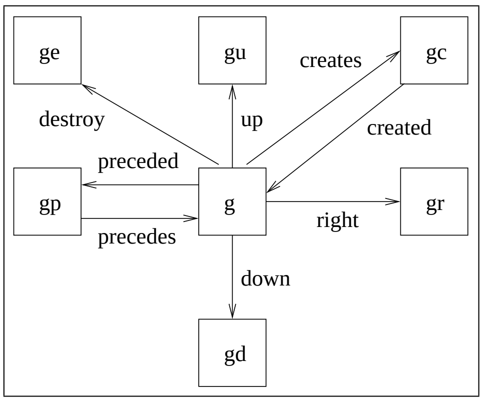


Figure 9: Goal class associations.

For the specification the GMoDS runtime model, a tree of goal instances must be specified. Because the type of goal instances does not need to be specified, nor the associations $precedes$, $creates$ and $destroy$, in this case only the tree structure of the instances is specified. Unfortunately the associations $up$, $down$, and $right$ from $G_{Spec}$ can not be used, because the tree structure of goal instances, $G_{Instances}$ is not always identical with the tree structure of $G_{Spec}$: a positive trigger event can create multiple instances of the same goal that are "sibling" nodes (having the same parent). For solving this problem we use different associations related to the sets of goal instances: $upInst$, $downInst$, and $rightInst$. The class used to specify the runtime model is $Tree$, a tree of goal instances, which is related to the set $G\_Instance$ from the runtime model.

There is no need to implement the six sets of goal instances from the implementation model, as presented in Fig. 7, because the current state of each goal instance specifies exactly the set to that instance belongs to.

For translating the GMoDS framework into an object-oriented model, we use the following classes:

- The class *G_Spec*, which is the main class of the translated model, because it contains both the static tree of goal classes, and the dynamic tree of the goal instances.
- The class $GName$, whose elements represent the nodes of the static static tree of goal classes.
- The class *Tree*, which represents the dynamic tree of the goal instances.
- The class *Goal*, whose elements represent the nodes of the dynamic tree of the goal instances.
- The class $Env$, that implements the rest of the Organization-based agent architecture: the Reasoning algorithm, and the Execution component.

In fact, $GName$ is not really a class, because it does not have constructors and destructors (the tree of the goal classes from *G_Spec* is static). We use instead the notion of *Records* for $GName$, an extension of the Event-B method. The main components of $GName$ are the following:

- $state$, which represents the current state of the corresponding goal class, from the set *Goal_*$STATES$.
- *curr_inst*, which represents the set of active goal instances of the corresponding goal class.
- $up$, $down$, $right$, $precedes$, and $preceded$ that represent the relations between goal classes, as presented in Fig. 9.

The class *Goal* represents the goal class hierarchy, as presented in Fig. 8. It contains only the attributes $upInst$, $downInst$, and *rightInst*, representing the relations between goal instances in a dynamic tree structure. the only operations allowed by *Goal* are the constructor $newGoal$ and the destructor $delGoal$.

The singleton class *Tree* contains only one attribute, *rootInst*, which represents the root of the dynamic tree of goal instances. *Tree* is a singleton class because there is a single object of *Tree*, which is an attribute of *G_Spec*. In addition, *Tree* has three operations:

- *deleteInst*, which deletes all the sub-tree having as parameter its root.
- *addChildInst*, which adds a new created instance as the first child of the parent specified as parameter.
- $addBrotherInst$, which adds a new created instance as the right of the goal instance specified as parameter.

The elements of *Tree* are instances of the class *Goal*. There is only one instance of the class $Tree$, which is a member of the class *G_Spec*.

The singleton class *G_Spec* contains only two attributes:

- *rootG*, an element of the $GName$ set, representing the root of the static tree of goal classes (e.g. $g0$ in our example).
- $tr$, the unique instance of the class *Tree*, representing the dynamic tree of goal instances.

In addition, *G_Spec* has several operations, according to the relations between the classes *G_Spec* and $Env$, as presented in Fig. 10:

- $start$, representing the event that starts the execution (or simulation) process of the MAO, and thus the Goal reasoning algorithm.
- *achivedInstGoal*, which informs *G_Spec* that a goal instance have been achieved.
- *createInstGoal*, which informs *G_Spec* that an instance of a goal class must be created.
- *deleteInstGoal*, which informs *G_Spec* that a goal instance must be deleted.
- *failedInstGoal*, which informs *G_Spec* that an active goal has failed.
- *createdInstGoal*, which informs $Env$ that a goal instance has been created.
- *deletedInstGoal*, which informs $Env$ that a goal instance has been deleted.

The unique instance of the class *G_Spec*, $gsp$, represents the *Goal reasoning* module, a part of the *Control component*, from the *Organization-based agent architecture*.

$Env$ represents the environment for the GMoDS framework that:

- Contains the *Reasoning algorithm* from the *Organization-based agent architecture* that performs the reorganisation structure of a MAO, based of information received from the Goal reasoning algorithm (e.g. from the GMoDS framework).
- Contains the *Execution components* from the *Organization-based agent architecture*, which contains the agents that achieve the roles related to the instances of the leaf goals in the goal tree, and send messages to those instances, when a goal has been achieved, or when it failed,
- Can send a message to the GMoDS system to start its execution (i.e. it sends the initial trigger to the parent goal of the goal hierarchy).

The relations between the classes $Env$ and the *G_Spec* are presented in Fig. 10, where:

- The relation $start$ exists between $Env$ and the root of the goal hierarchy (e.g. $g0$ in our example),
- The relations $achieved$ and $failed$ exist between $Env$ and the leaves of the goal hierarchy (e.g. $g2$, $g3$, $g4$, $g6$, and $g7$ in our example),
- The relation $create$ exists between $Env$ and some nodes from the goal hierarchy having a positive trigger (e.g. $g5$ and $g8$ in our example),
- The relation $delete$ exists between $Env$ and some nodes from the goal hierarchy having a negative trigger.
- Relations $created$ and $deleted$ exist between the goal classes from *G_Spec* and the $Env$, indicating to the Reasoning algorithm that some goal instances have been created, or deleted.

All these relation represent in fact operations of the class $Env$. Excepting the operations $start$, $achieved$, $failed$, $create$ and $delete$, the rest of the class $Env$ is not specified in this paper. This will be the subject of a future research.

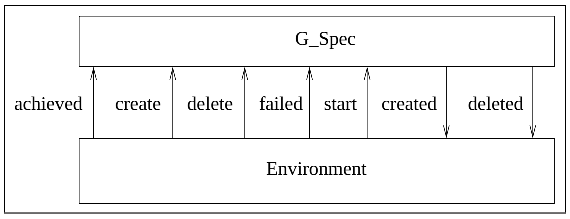


Figure 10: Environment and Goal classes associations.

## 5.3 An Example of Expressing GMoDS Models in Event-B

Using the patterns specified in Subsection 5.2 we can express the GMoDS system from Fig. 2 into an object-oriented model in Event-B. In fact, the model specified in Event-B encapsulates the GMoDS framework representing the Goal reasoning module into an object, $gsp$, instance of the class $G\_Spec$, while the rest of the Organization-based agent architecture is represented by the object $env$, instance of the class $Env$.

The context of the modeled system will contain the sets and the constants that follows the object-oriented patterns specified in Subsection 5.2.

```
CONTEXT $OBAA\_Ctx$
SETS
  $INST,\ GName,\ Goal\_STATES,\ Goal\_TYPE$
CONSTANTS
  $Void,\ Goal\_Inst,\ G\_Spec\_Inst,\ Tree\_Inst,\ Env\_Inst$
  $g0\_Inst,\ g1\_Inst,\ g2\_Inst,\ \ldots,\ g8\_Inst$
  $gsp,\ tr, env$
  $g0,\ g1,\ g2,\ \ldots,\ g8$
  $triggered,\ active,\ achieved,\ failed,\ obviated,$
      $removed,\ inactive$
  $AND,\ OR,\ LEAF,\ NONE$
AXIOMS
  $Void \in INST$
  partition$(INST, \{Void\}, Goal\_Inst, Tree\_Inst,$
      $Env\_Inst,\ G\_Spec\_Inst)$
  partition$(Goal\_Inst,\ g0\_Inst,\ \ldots,\ g8\_Inst)$
  partition$(G\_Spec\_Inst,\ \{gsp\})$
  partition$(Tree\_Inst,\ \{tr\})$
  partition$(Env\_Inst,\ \{env\})$
  partition$(GName,\ \{g0\},\ \{g1\},\ \ldots,\ \{g8\})$
  partition$(Goal\_TYPE, \{AND\}, \{OR\}, \{LEAF\})$
  partition$(Goal\_STATES, \{triggered\}, \{active\},$
      $\{achieved\}, \{failed\}, \{obviated\}, \{removed\})$
END
```

In the interface $OBAA\_Intf$, the variables and their invariants allow to specify the main object-oriented concepts, as defined in the Subsection 5.2:

```
INTERFACE $OBAA\_Intf$
SEES $OBAA\_Ctx$
VARIABLES
  $Env,\ G\_Spec,\ Tree,\ Goal$
  $type,\ state,\ curr\_inst$
  $creates,\ destroy,\ up,\ down,\ right,\ precedes,\ preceded$
  $rootG,\ rootInst,\ treeInst$
  $lastInst,\ lastGoalChild$
  $goalName$
INVARIANTS
  $Env \in \mathbb{P}(Env\_Inst \cup \{Void\})$
  $G\_Spec \in \mathbb{P}(G\_Spec\_Inst \cup \{Void\})$
  $Tree \in \mathbb{P}(Tree\_Inst \cup \{Void\})$
  $Goal \in \mathbb{P}(Goal\_Inst \cup \{Void\})$
  $rootG \in G\_Spec \rightarrow GName \cup \{Void\}$
  $treeInst \in G\_Spec \rightarrow Tree \cup \{Void\}$
  $rootInst \in Tree \rightarrow Goal\_Inst \cup \{Void\}$
  $lastInst \in GName \rightarrow Goal\_Inst \cup \{Void\}$
  $goalName \in Goal\_Inst \rightarrow GName$
  $type \in GName \rightarrow Goal\_TYPE$
  $state \in Goal\_Inst \rightarrow Goal\_STATES$
  $curr\_inst \in GName \rightarrow \mathbb{P}(Goal\_Inst \cup \{Void\})$
  $available\_inst \in GName \rightarrow \mathbb{P}(Goal\_Inst \cup \{Void\})$
  $up, down, right \in GName \rightarrow GName \cup \{Void\}$
  $creates, created \in GName \rightarrow GName \cup \{Void\}$
  $precedes, preceded \in GName \rightarrow GName \cup \{Void\}$
  $upInst, downInst, rightInst \in Goal\_Inst \rightarrow Goal\_Inst$
    $\cup \{Void\}$
INTIALISATION
  $\ldots$
```

The *initialization* event means in fact the creation of the static tree structure of $G_{Spec}$, as defined in Fig. 2, and some other initializations, such as (a) initialization of singleton classes, (b) defining the goal types, (c) managing the goal instances, (d) specifying the hierarchical structure of the goal tree, (e) specifying the positive and negative triggers, and (f) specifying the precedence relations between goals:

```
$Env := \{env\},\ G\_Spec := \{gsp\},\ Tree := \{tr\}$
$rootG(gsp) := g0,\ treeInst(gsp) := tr$
$rootInst(tr) := Void$
$\ldots$
$type(g0) := AND,\ type(g1) := OR$
$\ldots$
$curr\_inst(g0) := \varnothing,\ curr\_inst(g1) := \varnothing,$
$\ldots$
$available\_inst(g0) := g0\_Inst,\ available\_inst(g0) := g0\_Inst,$
$\ldots$
$lastInst(g0) := Void,\ lastInst(g1) := Void,$
$\ldots$
$up(g0) := Void,\ up(g1) := g0,$
$\ldots$
$down(g0) := q1,\ down(g1) := g2,\ down(g5) := g6,$
$\ldots$
$right(g1) := g5,\ right(g5) := g8,$
$\ldots$
$creates(g4) := g5,\ created(g5) := g5,$
$\ldots$
$precedes(g2) := g3,\ preceded(g3) := g3,$
```

In the following we present only the operations related to the operation $create$ of the environment. The other operations are similar. [1]

[1]The entire Event-B model is available at `http://software.ucv.ro/~mbrezovan/fm/gmods_model.zip`

Because some of the operations, such as $createGoalInstance$, of the class $G\_Spec$ creates recursively all the nodes that from a sub-tree having a root a goal instance, for correctly specifying the post-condition of this operation we need to define the transitive closure of the relation $down$. The same operation is needed for $createGoalInstance$, when the transitive closure of the relation $up$ is needed. Unfortunately, the Event-B language has a strict mathematical language, which is based on a set-theoretic model and corresponding proofs for modeling and refinement consistencies, and on the First Order Predicate Calculus for decomposition. For extending this mathematical language, the Theory plug-in was implemented, which is a Rodin extension that provides the facility to define mathematical extensions as well as prover extensions. There three kinds of extension, one of them is related to extensions of set-theoretic expressions or predicates. One example extensions of this kind consist of adding the transitive closure of relations or various ordered relations.

Butler [9] proposes propose a special case of an operator defined as the solution of some predicate, namely a fixed-point definition. For example, transitive closure of a relation $R$ may be defined as follows [9]:

```
**operator** $tcl$
    **prefix**
    **args** $r$
    **type parameters** $T$
    **condition** $down \in T \leftrightarrow T$
    **fixpoint** $y$ **where**
        $r \cup r\,;y$
        **order**$\{a \mapsto b \mid a \in T \leftrightarrow \leftrightarrow \wedge a \subseteq b\}$
**end**
```

We can define the transitive closure of the relation $down$ (as well as for the relation $up$) following the above example:

```
**operator** $tcl$
    **prefix**
    **args** $down$
    **type parameters** $GName$
    **condition** $r \in GName \leftrightarrow GName$
    **fixpoint** $y$ **where**
        $down \cup down\,;y$
        **order**$\{a \mapsto b \mid a \in GName \leftrightarrow GName \wedge a \subseteq b\}$
**end**
```

The transitive closure of the relation $down$ allow us to determine all the pairs $(g1 \mapsto g2)$ such that $up(g1) = g2$.

The operation $newGoal$ of the class $Goal$ class can be defined as follows:

```
$newGoal \mathrel{\widehat{=}}$
    **ANY** $self, g$
    **PRE**
        $self \in Goal\_Inst$
        $g \in Goal\_Inst \setminus Goal$
    **RETURN** $ret$
    **POST**
        $Goal' = Goal \cup \{g\}$
        $ret' = g$
    **END**
```

For allowing the polymorphism in this case, in the implementation module there will be eight final events related to the group $newGoal$: $newGoal\_g1$, $newGoal\_g2$, ..., $newGoal\_g8$.

The operation $addGoalInst$ of the class $Tree$ will add a single goal instance to the tree.

```
$addGoalInst \mathrel{\widehat{=}}$
    **ANY** $self, ge, gi$
    **PRE**
        $self \in Tree\_Inst$
        $ge, gi \in Goal\_inst$
    **RETURN** $ret$
    **POST**
        $ge = Void \Rightarrow rootInst'(self) = gi$
        $ge \neq Void \wedge downInst(ge) = Void \Rightarrow downInst'(ge) = gi$
        $ge \neq Void \wedge downInst(ge) \neq Void \Rightarrow lastInst'(ge) = gi$
        $ret' = self$
    **END**
```

In the implementation module there are three events in the group $addGoalInst$: $addRootInst$, $addChildInst$ and $addBrotherInst$, corresponding to the three above cases.

The main operations of create and destroy a goal instance of the interface $OBAA\_Intf$ are related to the class $G\_Spec$, which contains the tree of goal instances as attribute. The creation of an instance of a parent goals recursively creates children to the corresponding subtree that are non-triggered subgoals.

The operation $createGoalInstance$ of the $G\_Spec$ class can be defined as follows:

```
$createGoalInstance \mathrel{\widehat{=}}$
    **ANY** $self, gc, gi$
    **PRE**
        $self \in G\_Spec$
        $gi \in available\_inst(gc) \setminus curr\_inst(gc)$
        $gc \in GName$
        $created(gc) \neq Void$
    **RETURN** $ret$
    **POST**
        $curr\_inst'(gc) = curr\_inst(gc) \cup \{gi\}$
        $preceded(gc) = Void \Rightarrow state'(gi) = active$
        $preceded(gc) \neq Void \Rightarrow state'(gi) = triggered$
        $\ldots$
```

The following two predicates specify the recursive creation of the sub-tree having $gi$ as root for non-preceded goals:

```
$\forall gc \mapsto g \in tcl(down) \wedge created(g) \neq Void \wedge$
    $\exists i \in available\_inst(g) \setminus curr\_inst(g) \wedge$
    $preceded(g) = Void \wedge down(up(g)) = g$
    $\Rightarrow curr\_inst'(g) = curr\_inst(g) \cup \{i\} \wedge$
    $state'(i) = active \wedge$
    $downInst'(lastInst(up(g))) = i$
$\forall gc \mapsto g \in tcl(down) \wedge created(g) \neq Void \wedge$
    $\exists i \in available\_inst(g) \setminus curr\_inst(g) \wedge$
    $preceded(g) = Void \wedge down(up(g)) \neq g \wedge$
    $\exists gl \in GName \wedge right(gl) = g$
    $\Rightarrow curr\_inst'(g) = curr\_inst(g) \cup \{i\} \wedge$
    $state'(i) = active \wedge$
    $downInst'(lastInst(right(gl))) = i$
```

The case for preceded goals is similar to the non-preceded case. The last predicates specify the linking of the sub-tree root, $gi$, in the tree $treeInst$ of $G\_Spec$:

```
    $rootInst(tr) = Void \Rightarrow rootInst'(tr) = gi$
    $rootInst(tr) \neq Void \wedge down(up(gc)) = gc \wedge$
      $\text{card}(curr\_inst(gc)) = 1 \Rightarrow down(lastInst(up(gc))) = gi$
    $rootInst(tr) \neq Void \wedge down(up(gc)) = gc \wedge$
      $\text{card}(curr\_inst(gc)) > 1 \Rightarrow right(lastInst(gc) = gi$
    $rootInst(tr) \neq Void \wedge down(up(gc)) \neq gc \wedge$
      $\exists gl \in GName \wedge right(gl) = gc \wedge$
      $\text{card}(curr\_inst(gc)) = 1 \Rightarrow right(lastInst(gl))) = gi$
    $rootInst(tr) \neq Void \wedge down(up(gc)) \neq gc \wedge$
      $\exists gl \in GName \wedge right(lastInst(gl)) = gc \wedge$
      $\text{card}(curr\_inst(gc)) > 1 \Rightarrow right(lastInst(gc)) = gi$
    $ret' = gi$
  END
```

When implementing this operation in the implementation module, $OBAA\_Impl$, the function $createGoalInstance$ can be recursively applied, because the two associations, $down$ and $right$ can be viewed as the two links, $left$ and $right$ of a binary tree. There are four events in the group $createGoalInstance$ in the implementation module, two related to the leaf nodes, and two related to the non-leaf nodes:

– $createGoalInstanceLeafNotPrededed$,
– $createGoalInstanceLeafPrededed$,
– $createGoalInstanceNotPreceded$,
– $createGoalInstancePreceded$.

From the implementation module, $OBAA\_Impl$, we present only a single event for each described above operation.

For the operation $newGoal$ of the class hierarchy $Goa$ we present the event $newGoal\_g1$:

```
$newGoal\_g1 \;\widehat{=}$
  ANY $self, g$
  WHERE
    $self \in G\_Spec$
    $g \in g0\_Inst \setminus curr\_inst(g0)$
  THEN
    $curr\_inst(g0) := curr\_inst(g0) \cup \{g\}$
    $newGoal\_g1\_ret := g$
  END
```

For the operation $addGoalInst$ of the class $Tree$ we present the event $addChildInst$:

```
$addChildInst \;\widehat{=}$
  ANY $self, ge, gi$
  WHERE
    $self \in Tree\_Inst$
    $ge, gi \in Goal\_inst$
    $ge \neq Void$
  THEN
    $downInst(ge) = gi$
    $addChildInst\_ret := gi$
  END
```

When implementing the operation $addGoalInstance$ of the class $G\_Spec$ in the implementation module, the function $createGoalInstance$ can be recursively applied, because the two associations, $down$ and $right$ can be viewed as the two links, $left$ and $right$ of a binary tree. We present the event $acreateGoalInstanceNotPreceded$.

```
$createGoalInstanceNotPreceded \;\widehat{=}$
  ANY $self, gn, gi$
  WHERE
    $self \in G\_Spec$
    $gn \in GName$
    $gi \in available\_inst(gn)$
    $created(gn) = Void$
    $down(gn) \neq Void$
    $right(gn) \neq Void$
    $preceded(gn) = Void$
  THEN
    $curr\_inst(gn) = gi$
    $state(gi) := active$
    $addChild(treeInst(self), gi,$
        $createGoalInstance(down(gn)))$
    $addBrother(treeInst(self), gi,$
        $createGoalInstance(right(gn)))$
    $createGoalInstanceNotPreceded\_ret := gi$
  END
```

Finally, the environment class, $Env$, has five operations that simply call the operations of the class $G\_Spec$: $start$, $create$, $delete$, $achieved$, and $failed$. We present the operation $create$:

```
$start \;\widehat{=}$
  ANY $self$
  WHERE
    $self \in Env$
  THEN
    $createGoalInstance(gsp, g0)$
    $start\_ret := Void$
  END
```

We uses the *Pro-B* plug-in [23] for the *Rodin* platform [25] to verify the consistency of the modeled system. *ProB* is an animator and model checker for Event-B. It allows animation of Event-B specifications, and it can be used for model-checking, and for evaluating a variety of provers or tactics on a selection of proof obligations.

## 6 Conclusions

In this paper we presented an initial research related to express Organisation-based multi-agent software engineering (O-MaSE) to an object-oriented model in Event-B. We started to study the *Goal Model for Dynamic Systems* (GMoDS), a methodology that defines the operational semantics of a dynamically changing model of system goals, which has been used as the requirements modeling for the O-MaSE methodology.

Because the object-oriented model translated from the GMoDS models use some object-oriented concepts, such as inheritance, and calling of class methods, we used the modularisation plug-in of Rodin for implementing these concepts. We presented some pattern to translate GMoDS models to an object-oriented specification in Event-B, and we have illustrated these patterns for implementing an example from [13].

We planned to accomplish this work by testing the proposed patterns on real multi-agent systems, and to extend the research to the O-MaSE framework.

## References

[1] Jean-Raymond Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] Jean-Raymond Abrial. *Modeling in Event-B. System and Software Engineering*. Cambridge University Press, 2010.

[3] ADVANCE. `http://www.advance-ict.eu/`.

[4] Nazareno Aguirre, Juan Bicarregui, Theo Dimitrakos, and Tom Maibaum. Towards Dynamic Population Management of Abstract Machines in the B Method. In *ZB 2003: Formal Specification and Development in Z and B*, volume 2651 of *Lecture Notes in Computer Science*, pages 528–545. Springer-Verlag, 2003.

[5] Michael Wooldridge ans Nicholas R. Jennings and David Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[6] R.-J. Back and J. von Wright. Refinement Calculus, Part I: Sequential Nondeterministic Programs. In J.W. deBakker, W.-P. deRoever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 42–66. Springer, 1990.

[7] Ralph-Johan Back. Refinement Calculus, Part II: Parallel and Reactive Programs. In J.W. deBakker, W.-P. deRoever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 67–93. Springer, 1990.

[8] Amelia Badica and Costin Badica. FSP and FLTL framework for specification and verification of middle-agents. *International Journal of Applied Mathematics and Computer Science*, 21(1):9—25, 2011.

[9] Michael Butler and Issam Maamria. Mathematical Extension in Event-B Through the Rodin Theory Component. Technical Report 251, Electronics and Computer Science, University of Southampton, 2010.

[10] S.A. DeLoach, W. Oyenan, and E.T. Matson. A Capabilities Based Model for Artificial Organizations. *J. of Autonomous Agents and Multiagent Systems*, 16(1):13—56, 2008.

[11] S.A. DeLoach, M.F. Wood, and C.H. Sparkman. Multiagent Systems Engineering. *Intl. J. of Software Engineering and Knowledge Engineering*, 11(3):231–258, 2001.

[12] Scott A. DeLoach and Juan Carlos Garcia-Ojeda. O-mase: A Customizable Approach to Designing and Building Complex, Adaptive Multiagent Systems. *Intl. J. of Agent-Oriented Software Engineering*, 4(3):244–280, 2010.

[13] Scott A. DeLoach and Matthew Miller. A Goal Model for Adaptive Complex Systems. *Intl. J. of Computational Intelligence: Theory and Practice*, 5(2), 2010.

[14] DEPLOY. `http://www.deploy-project.eu/`.

[15] T. Dimitrakos, J. Bicarregui, B. Matthews, and T. Maibaum. Compositional Structuring in the B-Method: A Logical Viewpoint of the Static Context. In *ZB 2000: Formal Specification and Development in Z and B*, volume 1878 of *Lecture Notes in Computer Science*, pages 107–126. Springer-Verlag, 2000.

[16] A. Estefania, J. Vicente, and B. Vicente. Multi-Agent System Development Based on Organizations. *Electronic Notes in Theoretical Computer Science*, 150(3):55—71, 2006.

[17] Neil Evans and Michael Butler. A Proposal for Records in Event-B. In *FM 2006: Formal Methods*, volume 4085 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2006.

[18] J. Ferber, O. Gutknecht, and F. Michel. From Agents to Organizations: An organizational View of Multi-Agent Systems. In P. Giorgini, J.P. Müller, and J.J. Odell, editors, *Agent-Oriented Software Engineering*, volume 2935 of *Lecture Notes in Computer Science*, pages 214—230. Springer, 2004.

[19] V. Hilaire, P. Gruer, A. Koukam, and O. Simonin. Formal Specification Approach of Role Dynamics in Agent Organisations: Application to the Satisfaction-Altruism Model. *Intl. J. of Software Engineering and Knowledge Engineering*, 16(3), 2007.

[20] A. Iliasov, E. Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, D. Ilic, and T. Latvala. Supporting Reuse in Event-B Development: Modularisation Approach. In *Abstract State Machines, Alloy, B, and Z*, volume 5977 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2010.

[21] T. Juan, A. Pearce, and L. Sterling. ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In *Proc. 1st Intl. Joint Conf. on Autonomous Agents And Multiagent Systems*, pages 3–10, 2002.

[22] J. Mylopoulos, J. Castro, and M. Kolp. Tropos: A Framework for Requirements-Driven Software Development. In J. Brinkkemper and A. Solvberg, editors, *Information Systems Engineering: State of the Art and Research Themes*, pages 261–273. Springer-Verlag, 2000.

[23] ProB. `http://www.stups.uni-duesseldorf.de/ProB/index.php5/ProB_for_Rodin`.

[24] A. Regayeg, A. H. Kacem, and M. Jmaiel. Specification and Verification of Multi-Agent Applications Using Temporal Z. In *Proc. Intl. Conf. on Intelligent Agent Technology*, pages 260—266, 2004.

[25] RODIN. `http://www.event-b.org/`.

[26] Renato Silva and Michael Butler. Supporting Reuse of Event-B Developments through Generic Instantiation. In *Formal Methods and Software Engineering*, volume 5885 of *Lecture Notes in Computer Science*, pages 466–484. Springer, 2009.

[27] Colin Snook and Michael Butler. UML-B and Event-B: An Integration of Languages and Tools. In *Proc. IASTED Intl. Conference on Software Engineering*, pages 336–341, 2008.

# HTML5-based Mobile Agents for Web-of-Things

Jari-Pekka Voutilainen, Anna-Liisa Mattila, Kari Systä and Tommi Mikkonen
Tampere University of Technology, Korkeakoulunkatu 1, FI-33720 Tampere, Finland
E-mail: first.last@tut.fi

**Keywords:** web applications, mobile agents, Internet-of-Things, Web-of-Things, HTML5, JavaScript

**Received:** July 24, 2015

*Systems and services utilizing Internet-of-Things can benefit from dynamically updated software in a significant way. In this paper we show how the most advanced variant of moving code, mobile agents, can be used for operating and managing Internet-connected systems composed of gadgets, sensors and actuators. We believe that the use of mobile agents brings several benefits, for example, mobile agents help to reduce the network load, overcome network latency, and encapsulate protocols. In addition, they can perform autonomous tasks that would otherwise require extensive configuration. The need for moving agents is even more significant if the applications and other factors of the overall experience should follow the user to new contexts. When multiple agents are used to provide the user with services, some mechanisms to manage the agents are needed. In the context of Internet-of-Things such management should reflect the physical spaces and other relevant contexts. In this paper we describe the technical solutions used in the implementation of the mobile agents, describe two proof concepts and we also compare our solution to related work. We also describe our visions of the future work.*

*Povzetek: Razvit je sistem mobilnih agentov v HTML5 za splet stvari.*

## 1 Introduction

One of our drivers, the Internet of Things (IoT) refers to an approach where extensive amount of physical objects are inter-connected and also connected to the Internet. When implemented, IoT systems open possibilities for new applications and services for the users. At the moment much of the research has been invested in low-level issues related to addressing the different kinds of devices, bandwidth used in the communication, and latency in communications. However, since the main goal is to enable new applications and services higher-level protocols are also needed. Due to the diverse needs of different applications and services and vast number of different devices the protocols face extensive needs of adaptability. Design of such protocols upfront would assume extensive configurability and in extreme cases extra proxies and other workarounds. If all connected devices can dynamically accept new executable code, the risks are significantly reduced.

The other driver, from the human user point of view is the fact that people use an increasing number of Internet-connected devices to access services and applications from the Internet. This leads to a need to different multi-device experiences and eventually to concept of Liquid Software [23], where the user can effortlessly use multiple devices to access their applications and content from different devices in different contexts. Liquid Software, as described in [23] concentrates in systems where end-user devices with screens interact with Internet services. In this paper we show how the ideas of Liquid Software and Mobile Agents, as one building block to implement Liquid Software, can be applied in the world of Internet of Things.

Many researchers, for instance [10] separate Web of Things (WoT) from Internet of Things (IoT), because the former is based on resource-based APIs, resource-oriented architecture (ROA) and RESTful paradigm [6], and the latter is based on approaches that reflect the remote procedure call (RPC) paradigm. The main purpose of both approaches is the same: to connect devices around us to Internet and to use them in providing value to users. The difference is in the architectural approach, and because we share the architectural approach of WoT we use term WoT (Web of Things) in this paper.

In this paper we present our framework where HTML5 based mobile agents are used for programming WoT. The framework contains an agent framework that enables the usual operations associated with mobile applications, an application model for creating such agents, and a management system that is based on physical spaces and other real-world concepts.

This paper summarizes our earlier work on mobile agents [11], [12], [14] and [22], but also reports new work, for example, new way of separating user interface from the agent logic and for management of the agent system – including mobility of the agents.

The rest of this paper is structured as follows. After background and motivation in Section 2, we introduce our mobile agent framework and its implementation, and programming framework in Section 3. This description is based on older publication [22], but significantly reorganized and updated to reflect our latest design including new features like Management server and new declarative way to handle UI. Especially in Subsections 3.5 and 3.6 we discuss how a “thing” can host agents, what operations the agent can perform and how the system can be organized in Cloud Spaces. In Section 4 we present some experiments we have done with the system. In Section 5, we briefly address related work. In Section 6 we discuss current state of our work and our vision of future work. Finally in Section 7 we draw some final conclusions.

## 2 Motivation and background

Mobile Agents are executable entities that can move from one node to another together with the internal state of the application. This means that an executing agent can pause its execution in current location and then continue in a new location. In fact, mobile agents represent a special case of moving code combining remote evaluation with preservation of the internal state. The mobile agents discussed in this paper can preserve the internal state if the application needs that functionality. In some cases, we just need to send the code for remote evaluation.

Mobile agents have certain benefits that we see especially useful for Internet of Things. Among the benefits listed in [13], the following have special relevance in the scope of IoT:

- *Mobile agents reduce the network load.* Many "things" include sensors that monitor physical environments and thus potentially generate hidden data flows. If all that data is sent to application on another end of the network, the network may be flooded with data. A mobile agent running in the thing can reduce the network load by pre-processing the data generated by the sensors.
- *Mobile agents overcome network latency. The* latencies of networks, especially in wireless networks, can make real-time control impossible. Thus, everything cannot be done in the cloud and local execution is needed.
- *Mobile agents encapsulate protocols.* New protocols get invented frequently and objects in IoT should adapt to those. Agents are good tools for introducing new protocols or data formats.
- *Mobile agents execute asynchronously and autonomously.* This means that there is no need to generate network traffic for every execution. In case of wireless networks this also reduces power consumption.

As already pointed out, we propose using mobile agents in the context of WoT [12]. Our mobile agents are based on web technologies. An agent can move between different devices, and if necessary it is also possible to clone agents to create more instances. This enables the creation of increasingly complex configurations, where device- and context-specific decisions can also be taken in devices.

The Liquid Software dimension of our research is related to dynamically moving applications that enable use of several devices for accessing and controlling the WoT systems. The idea is that the execution should dynamically move to a location where it can be done more efficiently and where the required resources are. On the other hand, things that matter to the user, like user interfaces and user content should follow the user whenever possible and be accessible with device that user happens to have in her hand at that moment.

Third aspect of this paper is organization of the agent platform to "spaces" that relate to physical spaces and other real-world contexts. These Cloud Spaces define management structure for the WoT systems.

## 3 Architecture and concepts

Our whole system is based on mobile agents implemented with HTML5 technologies. This framework has originally been described in [22], but the design has evolved since then. Subsections 3.1, 3.2 and 3.3 report our current design, including new technique to separate UI from the logic, and framework for external control of the agents. The mobile agent framework is then in subsection 3.5 applied to Internet of Things by bringing agent servers close to various devices [12]. Finally, in subsection 3.6 we show how the systems are organized to managing contexts called Cloud Spaces [14].

### 3.1 Execution of HTML5 agents

In our design, an HTML5[1] agent is an HTML5 application that can run in two modes, with a user interface inside a browser and in a headless mode, that is, without a user interface, in an environment called Agent Server [22]. For executing the agent headlessly in the Agent Server, only a JavaScript virtual machine with a simple runtime environment is required. No full browser is needed. The state of the agent is saved during the migration between server and browser and the agent continues its execution as if there wasn't any change in the mode.

During its life cycle the agent may visit several browsers and several Agent Servers. An example life cycle is presented in Figure 1. The instance of an agent is created when it is downloaded from the Origin Server. This server is similar to an ordinary web server, and its task is simply to host applications. After the download, the executing agent can move to an Agent Server to continue its execution and back to a browser again.

The Origin Server maintains and serves all the files, and when the agent moves between Agent Servers and Browsers we usually deliver only the URL that point to a resource in the Origin Server. The receiving entity then fetches the static content from Origin Server.

The dashed box "Mgmt. server" and the dashed arrows in Figure 1 depict an optional management functionality that allows external entities to control agents.

Our all protocols are Web-friendly and rely on standard HTTP. Both Origin Server and Agent Server are

[1] For the purposes of this paper, the overall goal of HTML5 to support rich applications is important; we do not refer to any specific new technology introduced by HTML5.

HTTP servers that can be accessed with HTTP requests. Agents are fetched for execution with GET and pushed to server with POST. This means that an agent can also move from one server to another. In addition, the Agent Server can provide a list of running agents. Concretely speaking, the most important parts of the HTTP interface of the Agent Server are:

- */list* (HTTP GET) gets a list of active agents as an HTML file that can be shown in a browser.
- */upload* (HTTP POST) sends URLs to agent code and user interface together with serialized state. After receiving the Agent Server instantiates and starts the agent.
- */<id>* (HTTP GET) pauses the agent in server, serializes the state and sends it to the requesting browser

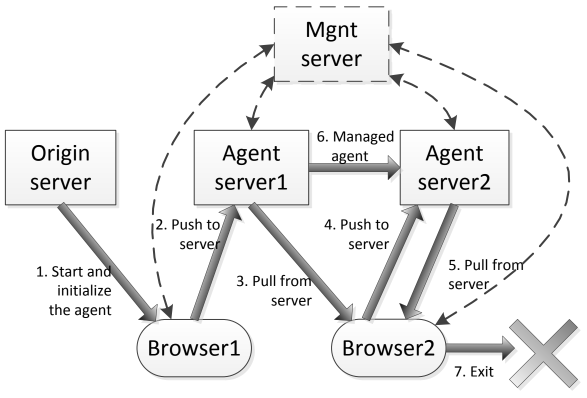


Figure 1. Life cycle of an HTML5 agent in the framework.

As usually in today's web applications, the HTML file of the agent includes references to Cascading Style Sheets (CSS), to other HTML files, images and other resources, and JavaScript files. Also, similarly to standard web applications the agent is first started by downloading the HTML file from the origin server.

Agents are serialized whenever they are moved between servers or between a server and a browser. The implementation of the framework provides mechanism for serialization of the relevant parts of the state. When an agent is about to move to a new location its state is serialized into JavaScript Object Notation (JSON) based on state variables defined by developer. An example of serialized agent description is shown below, where state of this agent includes four variables *low*, *high*, *count* and *history*:

```
{"auri":
  "http://xx.xx.xx.fi:pppp/gmonitor.js",
 "huri":
  http://xx.xx.xx.fi:pppp/gmonitor.html
 "id"    : "526636" ,
 "memory": {
   "high"   : 0.0253 ,
   "low"    : 0.0214 ,
   "count"  : 3,
   "history": [0.0253, 0.0234 ,0.0214]}}
```

This serialization contains URIs for the agent functionality (JavaScript file) and HTML based UI. In addition it has a unique identity variable (id) and set relevant variables in application state encoded in JSON dictionary “memory”.

When an Agent server receives the serialized agent description, it fetches the JavaScript code from the address in *auri*, in addition, the Agent downloads the other JavaScript files implementing the framework, it initializes the agent using the serialized state, and finally it starts the execution of the Agent.

When a browser requests the agent from the Agent Server some special arrangements are needed due to security and other limitations of the browsers. As response to a request from browser to the Agent Server, the Agent Server sends the content of HTML-file identified by ‘huri’ field of agent description. To that HTML file the Agent Server injects JavaScript to restore the transferred local state of the agent,

The execution model of the agent also needs to be suitable for the execution environment. First of all it needs to be suitable for running in the browser. For instance it should not block the event loop of the browser run-time. On the other hand it needs to proceed without user interface events delivered by the browser. Furthermore, the agent needs to have safe points in execution so that a consistent state can be serialized. In practice this means that all the application logic is embedded in specific event handlers that are triggered by timer events. This event-based execution model fits well to Agent Servers that have been implemented with Node.js [19].

## 3.2 Management API

The management protocol is also made compatible with the overall design. The Management Server implements a REST interface for both the moving agents and a control application. The control application may be operated by a human user, or be an autonomously running application. The most important part of the API for agents consists of two kinds of REST calls: “ImHere” when the agent has arrived to a new location, and “Status” call is sent to the Management Server on regular intervals. The response to these REST calls may contain an instruction to the agent to move to a new location (see arrow 6 in Figure 1). Our current implementation includes also instructions for the application to exit and to change values of variables. Control applications can browse the agents and their histories. Control applications can also send instructions to the Agent.

In the following we give a short example. When control application makes a GET request to */Agents* it gets a list of agent IDs as a response.

```
GET http://host/Agents => [211, 311]
```

In this case the Management Server knows about two agents. Detailed information about a specific agent can be retrieved with

```
GET http://host/Agents/211 => {…}
```

The response includes information about the location and status of the agent. The control application can request an agent to move to a new location by sending payload `[{"goto": http://server2}]` by using a PUT request

```
PUT http://host/Agents/211/todo
```

This request is now waiting in the Management Server until Agent 211 contacts the Management Server. When the agent 211 updates its status by sending

```
{"id":"211", "Status":"I'm fine"}
```

with request

```
PUT http://host/Management/Status
```

to the Management Server, the request to move to new location is delivered to agent 211 in the response, and the Agent framework initiates the move to the requested location.

This is a lightweight management framework that assumes the agents co-operate and does not affect agents that do not participate. The REST API of the Management Server has been designed both for automated control and for management user interface described in Subsection 4.2. On the other hand the framework relies on basic HTTP protocol and thus does not require the infrastructure to support any other protocol. In the current implementation only the agents that are in server obey all received instructions and agents that are in browser ignore the requests to move.

## 3.3 Programming agents

Core parts of the Agent framework have been encapsulated in a reusable JavaScript class Agent, and the developer should specialize her own version from that class. So far we have used the functional inheritance pattern presented in [3], but the more traditional prototype inheritance could be used, too. The application-specific sub-class of the Agent can override the following methods:

- Method *getRunningStatus()* – should return a string that the management interface of the agent server context can show.
- Method *preupload()* is called by upload() just before serialization as the first the uploading. By overriding this method the agent can implement application specific preparations for the uploading.
- Function *continueWork()* – re-initializes the execution when the agent has arrived and de-serialized in a new location and the execution should be resumed. This function initializes the state of the agent by recreating the variables.

In addition, the agent has to provide a function that creates and initializes the agent object.

The framework provides also a set of utility methods that the above methods and functions can call. The most important utility methods are:

- *registerVar(name*) – with this function the application can state that a variable is part of relevant local state and will be automatically serialized.
- *setWork(function, interval)* – sets the work function that is periodically executed with the given interval. The framework assumes that the work function returns reasonable quickly.
- upload(url) – uploads the agent to an Agent Server specified by parameter url. This function first stops execution, then serializes the state and finally sends the serialized agent to the Agent Server.

As discussed earlier, the agents run in the headless mode in Agent Server and with the HTML and CSS files in browser. This means that the JavaScript code of the agents has to be written to be executable without presence of the complete Document Object Model (DOM) tree. Separation of the application logic from the UI part is not always easy since many Web application frameworks rely on existence of the DOM-tree. In our first implementation we assumed that agents are written for the framework so that user interface is nicely separated from the application logic. In addition, we provided a very simple DOM emulation to help writing of portable applications. We have later experimented with a different approach to help application developers in implementation of Agents that can run with and without DOM. This approach is based of declaration of the binding between application logic and user interface with primitives like:

```
BindModeltoView(['var','elem']);
BindModeltoView(['var.func', 'elem']);
```

The first declaration states that if element with id 'elem' exists in DOM-tree its value (innerHTML) is updated with the value of variable 'var' whenever value of var changes. In the latter version function 'func' is used instead of simple assignment to innerHTML of 'elem'. If the above binding mechanism is used, the application code does not need include UI-specific code and thus there is no need to deal with differences between server and client since the framework includes conditional code.

## 3.4 Agent communication

A simple agent-to-agent communication framework has also been implemented [11]. This framework allows agent to send messages and to receive their messages regardless of their current location. Because the web infrastructure does not support communication between browsers, the all communication is routed through an agent server.

With the current API, the sending agent initializes the communication as follows:

```
c = new CommComponent(function(msg) {
  …
});
c.setNameSpace("myChannel");
c.initIO();
```

and a message is sent with:

```
c.sendMessage(obj)
```

Like in other parts of our framework the content is sent over the network as a JSON string. The namespace “myChannel” is kind of channel and the receiving end can listen the channel with the following code:

```
c = new CommComponent(function(msg){
    // process incoming msg
});
c.setNameSpace("myChannel");
c.initIO();
```

We have not used this framework much in our applications still, because most of our example applications have assumed that the moving agents bring the data with them. We have just validated that our implementation that is based on WebSockets [25] works.

## 3.5 Agent servers in “things”

As described earlier, the core components of our Agent Server are the HTTP server and a virtual machine executing JavaScript. These can be implemented, for example, with Node.js [19] technology. The agent server has two main functions: 1) implement execution environment that is compatible enough with the browser and 2) simple management function for agents.

As the implementation of the Agent Server only requires Node.js and a few hundred lines of JavaScript code and because our Agent Server does not require lot of computational resources, and it can be included in many small devices – or “things” – that are connected to the Internet. In our experiments we have used a low-cost single board computer Raspberry PI [20], which typically runs Linux. The infrastructure requirements are equal to those of WoT, because the devices are accessed with standard HTTP requests such as GET and POST. With these requirements the Raspberry PI device goes beyond the bar with a clear margin.

We assume that most devices that can be nodes in SOA based IoT or REST-based WoT can also host our Agent Server. This would bring benefit of mobile agents described above, but also enable new ways for remote management and extending the functionality by adding new code in a form of mobile agents. The possible application areas include the following:

- *Home automation that goes beyond remote control*. An intelligent agent can work on behalf of the user and implement even complex strategies to optimize energy consumption and user comfort.
- *Support for new communication protocols or applications*. In most cases the applications are in the Internet, but the “things” need to be accessible. Sometimes new application will need new functionality from the devices.
- *Compatible extensions to already existing systems*. Interoperability with new devices may be achieved by adding new intelligence to existing devices.

The proposed approach has obvious benefits over the solutions that have been more conventionally used. From the perspective of the “thing” executing the agent, the agent framework based on managed runtime effectively creates a sandbox that separates the agent from the rest of the system. Therefore it is, for instance, possible to run real-time critical code in the same system, and only execute the agent when there is leftover execution time, a partitioning which is supported by many real-time operating systems. A further benefit over other agent frameworks is that we are solely relying on web protocols and technologies. The ecosystem that builds web applications has presently advanced to a level where the web is increasingly a platform for all applications. Allowing this ecosystem to build mobile agents for WoT creates low-hanging opportunities, because there is no need to invest in familiarizing yet another platform.

## 3.6 Structuring of the framework

In [14], we have presented and demonstrated a structuring concept that incorporates HTML5 agents and their servers with a data solution to store user’s content. Each Cloud Space is essentially a private cloud which hosts user’s data and applications. In the context of WoT, Cloud Spaces can be seen as ecosystem where each “thing” provides small functionality for the Cloud Space as a whole. Data streams between the nodes can be implemented using agents and when adding a new node to the system, agents can provide architecture configuration automatically to the new node. Figure 2 depicts on possible Cloud Space configuration with Web of Things.

Each “thing” in WoT implements minimum of the Agent Server architecture and when new device is added to WoT, agent is sent to the new device to configure it. For example agent creates new public interfaces to the new device, which can then be used for data streams. Or the agent can implement application logic for the device, which is then executed even without the agent. Even if the agent does not modify the programming of the new device, it can provide information about WoT, for example, location of other servers and devices.

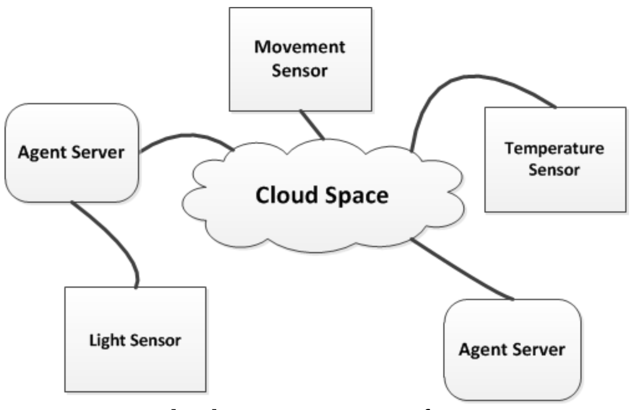


Figure 2. Cloud Space in context of WoT.

# 4 Proof-of-concept experiments

During our research we have implemented a few proofs of concept and demonstrators. Here we describe two demonstrators that relate to devices in WoT and to management of the agents and Cloud Spaces.

## 4.1 Agents for embedded device

To verify and demonstrate our idea we implemented a simple agent that collected information from different sensors hosted by different devices (Figure 3).

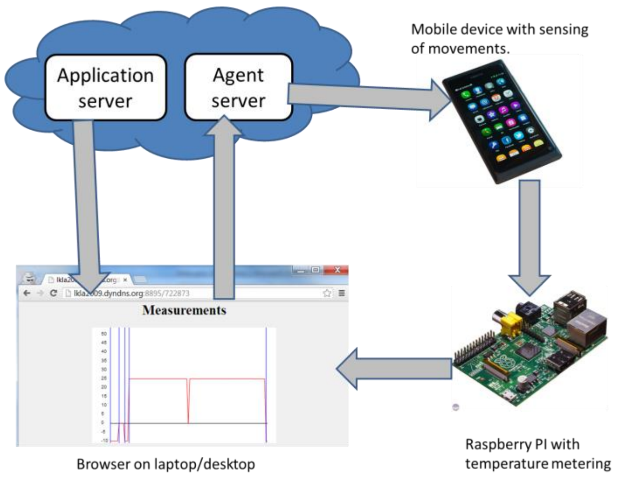

Application server
Agent server
Mobile device with sensing of movements.
Measurements
Raspberry PI with temperature metering
Browser on laptop/desktop

Figure 3. A traveling agent in different devices.

The implementation is based on the following components.

1. Two Agent Servers, one running in Raspberry PI, and another in a standard Linux server running on a virtual machine in a cloud. Both servers are based on Node.js [19] technology, and the implementation of the agent framework is the same in both servers. It is also possible to connect external sensors and actuators to Raspberry PI. In our case, we have connected a temperature sensor DS18B20 [15] for our experiments.
2. An agent that travels between servers and browsing devices. The agent is written in a manner that it can measure temperature when it is in the Raspberry-hosted server and if temperature sensor is available. When in browser the agent receives DOM device orientation events [24] and measures the orientation of the device. Based on the orientation events, the agent also calculates a "restless index" – i.e. how much the device is rocked or shaken lately. In other words, the agent collects different data in different devices but remembers and aggregates all the collected data to a pre-processed form.
3. Visualization of the collected data when the agent is in browser. In this visualization we show graphs of the temperature and restless index over time.

In the scenario depicted in Figure 3 the agent is first downloaded to a browser running on a Windows laptop. From there it is pushed to an Agent Server in the cloud. The graphical display disappears but collected statistics are preserved and the agent continues its execution. Unfortunately no sensors were available, so no real data was collected. Next the agent is downloaded to a browser running on a smart phone, there the graphical visualization is generated again and the user can see from the graph what has been measured and collected. From mobile browser the agent is uploaded into an Agent Server in Raspberry PI. From there the agent is finally downloaded to a desktop browser.

The purpose of this experiment was to validate that the agent runs in all needed hosts and also to demonstrate the idea. An example – a different execution from the one shown in Figure 3 – of the visualization has been given in Figure 4. The X-axis in Figure 4 represent time (concrete values not shown in Figure 4) and Y-axis show the sensor values.

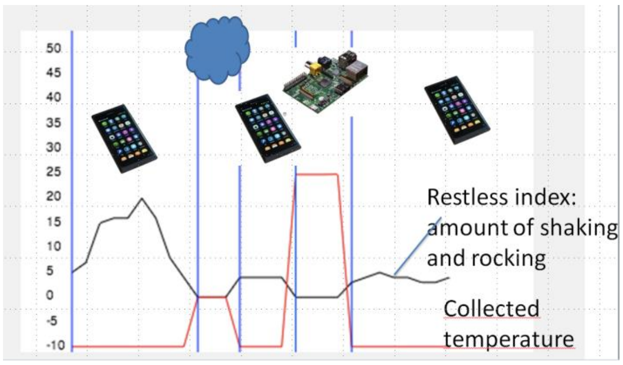


Figure 4. Visualization of the collected sensor data.

Additional text and images have been added to the picture to improve the presentation in this paper. The blue vertical lines indicate moves from one location to another. The history of events in this example run is the following: the agent was first downloaded to browser in a smart phone. Since accelerator sensors were available the restless index gets calculated and recorded, but because temperature sensor is not accessible, temperature defaults to -10 degrees C. The agent is next pushed to an Agent Server in the cloud where neither sensor is available and both readings default to 0. Then the agent is downloaded back to a mobile browser and further pushed to a server in Raspberry PI. In Raspberry PI the agent reads and collects temperature data until it gets downloaded back to mobile browser.

## 4.2 Managing agents in cloud space

For combining concept of agents and Cloud Space we have also implemented a proof of concept manager for agents running in Cloud Space [14]. The manager is a web application with a 3D interface for managing agents in Cloud Space contexts. By using the manager the user can access to her Cloud Space context and agent servers inside the Cloud Space. User can fetch agents from servers to her web browser and move agents from a server to another server even between contexts.

As Cloud Space context can represent a physical place, panoramic photo spheres are used to visualize the context in the 3D management UI. A real-world image helps the user in mapping of the concepts of Cloud Space to the physical space. Agent Servers in a context are represented as 3D grids and agents running in a server are shown as cuboids are placed to the grid. User performs all management actions, e.g. navigating in contexts, moving agents and fetching agents, via direct manipulation using mouse and keyboard.

Figure 5 presents the management UI in action. In the top section of Figure 5 (marked with 1) the user has dragged the agent on top of the context which she wants to move the agent. When she releases the agent the Management View changes to the context she chose. This is visualized in the middle section of Figure 5. Finally the user can drag the agent to the server in the context and the agent is moved there (bottom section in Figure 5).

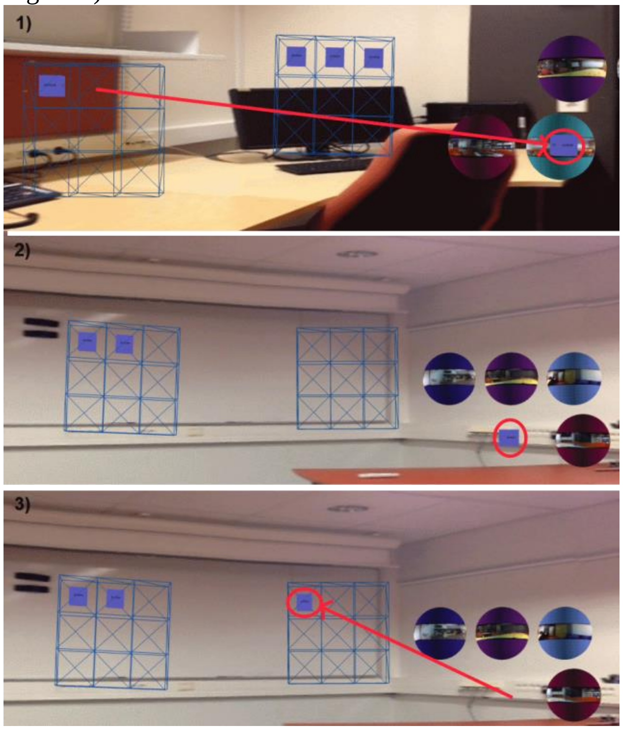


Figure 5. Examples of management views [14].

## 5 Related work

Use of Web and HTML as an agent platform is not very common. The Radigost system [16] [17] uses Web and JavaScript as an implementation platform for multi-agent systems. It has many interesting features like support for standard agent communication mechanisms and yellow-pages service for agent directories. However, it does not support dynamically moving agents or running agents outside browser. Radigost has later been merged with JavaEE-based agent framework that allows execution of agents also in the server side. In contrast to many benefits of JavaEE-based agent platform, it cannot be hosted on small devices as required by WoT applications. Another Web-based agent-framework has been described in [5]. In that concept the agent platform is based on concepts of Pneuna that is relatively close to our agent description and Soma that is the execution environment. In this approach Soma hides the differences of browser and server environment and creates a completely new application platform for mobile agents. In our approach standard and well-known HTML5 is the agent platform. In addition, the approach presented in [5] has not been designed for pushing agents to agent server when user or browser is not active or when the agent should find a new browser to run on.

As discussed earlier, Web of Things (WoT) and Internet of Things (IoT) approaches lead to a bit different architectures. Because the former leads to resource-oriented architecture (ROA) and the latter is based on approaches that reflect the remote procedure call (RPC) paradigm. While our work could be connected to both approaches, we propose a third approach that is based in sending code for execution in or close to a “thing”. In the categorization of moving code proposed in [1], this is called Remote Evaluation. The code sent to remote host can expose new interfaces either in WoT or IoT style. As our system allows executing code to move with its internal state and because the code and state can further move to a yet another location, our system fulfills the criteria of mobile agents. For many WoT and IoT applications, the core subset of mobile agent behavior – remote evaluation – is enough, but moving with state and ability to move even further are available for those applications that benefit from it.

There are a few approaches that support uploading and remote evaluation of code in a “thing”. For example, MoteLab [27] is a test bed for sensor networks. The developers using MoteLab can upload executable Java code with a job description towards a “thing”. The Web interface is a separate system based on PHP. Somewhat similar system is Kansei [4] – later refactored to KanseiGenie – where developers can also create jobs to execute sensor applications. Our system can also be used in a similar way and from similar motivations. However, in our system the uploaded code is Web content and we can upload an executing agent with its internal state.

Use of web technologies to for IoT or WoT applications is not new. For example, WebIoT [1] is based on similarities to our work. Similarities lie especially in the aim to bring IoT to Web 2.0 and allowing users to develop, deploy and execute their own applications. However, WebIoT does not support agent model.

Maybe the most similar approach to us is the mobile agent framework proposed in [6]. It provides nodes in heterogeneous device networks with a way to communicate and co-operate. Furthermore, it provides means to proactively search for required resources. The system is based on Java-based AgentSpace [21] mobile agent platform. At the moment we do not have similar automatic searching – this is left for future work. On the other hand we have the unique benefits of using the Web, which enables leveraging the power of the web development ecosystem in application development [22].

## 6 Discussion and future work

The ability to send code for remote evaluation and especially mobile agents is useful when implementing new types of IoT applications. This approach increases flexibility of the system design and evolution of IoT since the new code can add new functionality and adapt the device to new requirements. Moving code and especially agents can also be used to add autonomous intelligence to systems.

Our example agent collected data that was available at the particular location of execution and different data was collected in different locations. So, the agent adapts to its execution environment. One benefit of mobile agent is reduction of communication. In our case the sensor data, like temperature measurement, was continuously collected but sent over network only when agent moves. Furthermore, calculation of the restless index is an example of agent that reduces communicated data by pre-processing the raw data.

We see that use of web technologies as a basis for our agent framework gives us several benefits. First of all we gain ecosystem benefits in terms of competencies, training material and tools. Secondly, any device with a reasonable recent browser can be used to run and control the agents. Thirdly, web-based agents can be run both in “things” and servers in the cloud, and integrate well with the infrastructure of the Internet.

In the future we would like to study the opportunities when combining our mobile agents to RESTful or SOA paradigms more closely. For instance, and an agent that is located in a “thing” with a temperature sensor can expose a REST API for applications to ask current temperature, list of recent measurements, or some other information, depending on the application needs.

Mapping our framework to agent-related standards is also a potential future topic. Since our agent-to-agent communication solution is very generic, we assume that it can be used as a transport layer to FIPA Agent Communication language [7] in a similar manner as in Radigost [17], but the mapping between our management system and corresponding FIPA standard [8] requires some analysis.

So far we have tried the framework only in reasonable small cases, and one of the most important topics for future work is testing it in a larger context. One potential case is experience roaming with Liquid Software [23]. Mobile agents would allow users to bring their preferences and on-going work to the physical smart spaces they enter. For instance the user can bring his lightning, heating and other preferences to hotel rooms while they travel and they can also use their favourite user interface in favourite mobile device to monitor and control devices in the visited environment.

Another possible experimentation would involve a system that moves agents autonomously. Especially in sensor network systems, automatic crawling of agents could allow them autonomously search and collect the needed data. The recently added management API (see Subsection 3.2) and its underlying mechanisms provide a basis for this such experiment. One of the design goals of the management API was to support such autonomy. Since this API is still a reasonable new feature our framework and we need experiment with it by implementing some example application. Moreover, some obvious things that require attention are related to non-functional properties of our agent system, including scalability and security in particular.

## 7 Conclusions

In summary, we believe that by the end of this decade multi-device usage will become so seamless and ubiquitous that “it will weave itself into the fabric of everyday life until it is indistinguishable from it” [26]. In contrast to numerous platform and vendor-specific systems, our work on HTML5 agents and related infrastructure demonstrates that such future can be created with technologies that reflect Open Web principles laid out in the *Mozilla Manifesto* [18]. Built with technologies that are open, accessible and as interoperable as possible, and run in standards compatible web browser without plugins, extensions or additional runtimes, they require no installation or manual upgrades, and they can be deployed instantly worldwide, and allow application development and instant worldwide deployment without middlemen, distributors, or platform-specific app stores.

We believe that these properties will be key characteristics for the IoT and WoT devices of the future as well. When these properties are extended to devices, the devices can be part of the new and unified computing infrastructure defined by the Internet.

In particular, we believe that mobile agents can play a special role of connecting devices to the Internet and in allowing the most efficient use of them in the world where everything becomes Internet-connected.

## 8 References

[1] Carzaniga, A., Picco, G., P., Vigna, G., 1997. Designing distributed applications with mobile code paradigms. In Proceeding of the 19th international conference on Software engineering (ICSE’97), May 17-23, 1997, Boston, Massachusetts, USA. Pages 22-32.

[2] Castellani, A.P., Dissegna, M., Bui, N., Zorzi, M., WebIoT: A Web Application Framework for the Internet of Things, Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE, Paris France, 2012, pages 202 – 207.

[3] Crockford, D.: JavaScript: the Good Parts, O'Reilly Media, Inc. May 8, 2008.

[4] Ertin E., Arora A., Ramnath R., Naik V., Bapat S., Kulathumani V., Sridharan M., Zhang H., Cao H., and Nesterenko M., “Kansei: a testbedfor sensing at scale,” in ACM/IEEE IPSN, Nashville, Tennessee, USA,2006, pp. Pages 399–406.

[5] Feldman, M.: An approach for using the Web as a Mobile Agent infrastructure, In pro-ceedings of the International Multiconference on Computer Science and Information Technology, pp. 39 – 45, 2007.

[6] Fielding R.. Architectural styles and the design of network-based software architectures. Doctoral Dissertation. University of California, 2000.

[7] FIPA Agent Communication Language Specifications, http://www.fipa.org/repository/aclspecs.html, last visited 2.2.2015.

[8] FIPA, Agent Management Specifications, http://www.fipa.org/repository/managementspecs..html, last visited 2.2.2015.

[9] Godfrey W. W., Jha S. S., B. Nair S. B., On A Mobile Agent Framework for an Internet of Things,

In proceeding of: International Conference on Communication System and Technologies, CSNT 2013, 05-08 April 2013, Gwalior, India, At Gwalior, India. Pages 345 – 350.

[10] Hong Y, A Resource-Oriented Middleware Framework for Heterogeneous Internet of Things, International conference on Cloud and Service Computing (CSC), 2012, Shanghai, China, Pages 12-16.

[11] Järvenpää, L., Development and evaluation of HTML5 agent framework. Master of Science Thesis, Tampere University Technology, 2013.

[12] Järvenpää, L., Lintinen, M., Mattila, A-L., Mikkonen, T., Systä, K, Voutilainen, J-P. Mobile Agents for the Internet of Things, In WASA2013, 3rd Workshop on Applications of Software Agents, Sinaia, Romania, October 11-13, 2013.

[13] Lange, D., B., Oshima, M., 1999. Seven good reasons for mobile agents, In Communications of the ACM, Volume 42 Issue 3, March 1999, Pages 88 – 89.

[14] Mattila, A.L., Systä, K., Mikkonen, T. and Voutilainen, J.-P., Cloud Space – Web-based Smart Space with Management UI, A short paper in 10th International Conference on Web Information Systems and Technologies (WEBIST), Barcelona 3-5, April, 2014.

[15] Maxim Integrated, DS18B20, Programmable Resolution 1-Wire Digital Thermometer, Datasheet. http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[16] Mitrović, D., Ivanović, M., Budimac, Z., Vidaković M., Radigost: Interoperable web-based multi-agent platform, The Journal of Systems and Software 90, 2014. Pages 167–178.

[17] Mitrović, D., Ivanović, M., and Bădică, C., Delivering the multiagent technology to end-users through the web. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)* (WIMS '14). ACM, New York, NY, USA, 2014.

[18] MOZILLA-MF Mozilla, Inc., The Mozilla Manifesto. URL: http://www.mozilla.org/en-US/about/manifesto/.

[19] Node.js. Web page for document and download of nodejs technology, http://nodejs.org/. Last viewed 03.02.2013.

[20] Raspberry PI web page, http://www.raspberrypi.org, Last visited 28.6.2013

[21] Silva A., Da Silva M., An Overview of AgentSpace: A Next-Generation Mobile Agent System, In Proceedings of the Mobile Agents'98, Springer, 1998. Pages 148 – 159.

[22] Systä, K., Mikkonen, T., Järvenpää, L.,. HTML5 Agents – Mobile Agents for the Web, In the Proceedings of 9th International Conference on Web Information Systems and Technologies (WEBIST), Aachen, Germany, 8-10.5.2013. Pages 37-44.

[23] Taivalsaari, A., Mikkonen, T., Systä, K., Liquid Software Manifesto: The Era of Multiple Device Ownership and Its Implications for Software Architecture. A short paper to appear in 38th Annual IEEE International Computers, Software, and Applications Conference (COMPSAC) in Västerås, Sweden 21–25 July, 2014.

[24] W3C, DeviceOrientation Event Specification, W3C Working Draft 1 December 2011 http://www.w3.org/TR/orientation-event/

[25] W3C, The WebSocket API, W3C Working Draft, 19 April 2011, http://www.w3.org/TR/2011/WD-websockets-20110419/

[26] Weiser, M.,, The Computer for the 21st Century. Scientific American, September 1991, pp. 94-104.

[27] Werner-Allen G., Swieskowski P., and Welsh M., “MoteLab: a wireless sensor network testbed,” in ACM/IEEE IPSN, Apr. 2005, Pages 483–488.

# Secured Storage for Dynamic Data in Cloud

Veeralakshmi Ponnuramu, Department of Computer Science and Engineering
B.S.Abdur Rahman University, Chennai, India
E-mail: veerphd1@gmail.com

Dr. Latha Tamilselvan, Department of Information Technology
B.S.Abdur Rahman University, Chennai, India
E-mail: latha_tamilselvan@yahoo.com

**Keywords:** confidentiality, remote integrity verification, dynamic data operations, public verifiability, symmetric encryption, data storage security, privacy preserving auditing

**Received:** December 21, 2015

*Cloud is a growing computing paradigm in which the services and resources are provisioned dynamically through internet. In cloud, the users' data can be stored in the remotely located data servers that are maintained by cloud service providers to relieve the users from the local storage and maintenance. The major security challenge with cloud computing is that the users cannot have direct control over the remotely stored data. The imperative security concerns in cloud are the integrity and confidentiality of data. Many existing remote integrity checking methods fail to serve for the data that can be dynamically updated. To preserve the privacy of dynamically changing data, an efficient approach maintaining the confidentiality and assuring the integrity of data is proposed. In this scheme, public auditability is enabled by introducing a Third Party Auditor (TPA) for verifying the data integrity. It is ensured that the data stored in the untrusted cloud server is confidential and consistent by using a data encryption algorithm called 2-Keys Symmetric Encryption. Unlike other encryption algorithms, this encryption algorithm needs lesser computation overhead. Encryption and decryption algorithms are developed in java and Remote Method Invocation (RMI) concepts are used for communication between client and server. Simulation environment is set up with the eucalyptus tool. The performance analysis and simulation results prove that our proposed scheme is secure and proficient and it reduces the computation cost of server and verifier.*

*Povzetek: Ta članek predlaga postopek za zagotavljanje varnosti za dinamično spreminjanje podatkov, ki so shranjeni v oblaku.*

## 1 Introduction

Cloud is an on-demand, pay-by-use model for sharing a pool of computing resources like servers, CPU cycles, memory, applications, storage and services that is managed by cloud service providers. The services can be easily provisioned from the cloud providers and released with minimum endeavor by the cloud users. The users can access data and applications from remote servers with the fixed or mobile devices. Cloud storage becomes an increasing attraction in cloud computing paradigm. Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Services (S3) are the well-known examples of cloud storage services. With the cloud, the small organizations can hire the resources from the cloud rather than purchasing them and avoid the capital costs for software and hardware. With cloud, IT infrastructure can be easily adjusted to accommodate the changes in demand [5].

There are various [6], [10] issues like security, scalability, availability, resource scheduling, data migration, memory management, data privacy, data management, reliability, load balancing, access control in cloud because it uses many technologies including virtualization, networks, transaction management, databases and operating systems. Cloud moves the applications, software and databases to the large data centers that are located anywhere in the world, where the servers can't be trustworthy. This unique feature of cloud imparts many new security challenges like confidentiality and integrity. At the same time, the cloud offers many benefits like enhanced collaboration, limitless flexibility, portability and simpler devices. To enjoy the benefits of cloud, the users have to store their data in the encrypted format. Encryption of data can handle the confidentiality issue. But verification of integrity without having a local copy of data is a difficult task in cloud. So the existing methods like SHA, MD5 [16] can't be directly used. The simple way to check the integrity of data is to download the full data stored in the cloud to ensure its integrity. It incurs excessive I/O cost and heavy communication overhead across the network. So some effective methods are required for assuring the confidentiality and integrity of data stored in the cloud with the minimum overhead.

Recently many remote integrity checking methods [13], [8], [14], [1], [6], [15] were proposed to check the integrity of data stored at the remote server. In these some of the methods are not dealing with confidentiality and are not supporting the dynamic data operations. So, a new cryptographic mechanism for protecting confidentiality and integrity of stored data in cloud is needed.

- Confidentiality: It ensures that computer information are used and gained access by only authenticated and authorized individuals.
- Integrity: It denotes that the data in the cloud can be updated only by authorized persons in authorized ways. Updates of data file include writing, appending to the existing data, changing, deleting the data and creation of new information.

There are two kinds of encryption algorithms. That is symmetric and asymmetric encryptions. In symmetric algorithms, the same key can be used for both encryption and decryption. Symmetric algorithms are highly secured and can be executed in high speed. In case of asymmetric algorithms, different keys are used for encryption and decryption [16]. In this, data can be encrypted using a public key, and decrypted with a private key. Asymmetric encryption algorithms (called as public-key algorithms) need a key of 3,000 bits to produce the same level of security as that of a symmetric algorithm with a 128-bit key. Since the asymmetric encryption algorithms are slow, they cannot be used for encrypting bulk of data. So, in this paper, a novel symmetric encryption algorithm has been proposed for maintaining confidentiality and assuring data integrity.

## 2 Related work

### 2.1 Security issues in cloud

There are numerous security issues in cloud as the customers are not having direct control over the stored data in cloud. Jensen et al., [11], [12] discussed the security issues arising from the usage of cloud services and by the technologies used to build the internet-connected and cross-domain collaborations. It emphases browser security, WS-security, cloud integrity, transport layer security, and binding issues in the field of cloud.

### 2.2 Merkle hash tree (MHT)

Wang et al., [3] verified the correctness of data stored in server by allowing the Third Party Auditor. With the aid of Merkle hash tree it is possible for the clients to perform block-level operations on the data files by preserving the level of data correctness assurance. In this scenario, chances are there for the third party auditor to misuse the data while doing the verification operation. Lifei Wei et al., [9] established a new mechanism to verify the correctness of computations (addition, subtraction, multiplication, division, etc.) done by the cloud provider. For that, they have used the Merkle hash tree for checking the computation correctness. The only criteria are the number of computations submitted to the server must be in the power of 2, since the Merkle hash tree has the 2n number of leaves.

### 2.3 Advance computation of tokens

A storage correctness model for verifying the correctness of stored data by calculating a few numbers of tokens was proposed by Wang et al., [3]. This insists the user to pre-compute a number of verification precomputed tokens, each of them covering a random subset of data blocks. It allows the cloud user to challenge the cloud server with a set of pre-computed tokens. Once accepting the challenge token, the cloud server computes a signature over the specified data blocks and returns the signature to the cloud user. The signatures returned by the provider should match the relevant tokens pre-computed by the user. The main challenge of this system is that the cloud user can able to test the cloud server only for a definite number of times.

### 2.4 Proof of retrievability scheme (POR)

For verifying the data integrity some sentinel characters were embedded in the data file by A.Juels and B. S. Kaliski [1]. These sentinels were hidden in the data blocks. In the verification phase, the user can challenge the server by mentioning the positions of sentinels and request the provider to return the relevant sentinel values. This procedure allows the user to challenge the server for a limited number of times by knowing the positions of the sentinel values in advance. G.Ateniese et al., [6] proposed a new model called “Provable Data Possession” to ensure the possession of files stored on the untrusted server. They had used RSA- based homomorphic tags for assessing outsourced data. Here also the user needs to pre-compute the tags and store all the tags in advance. The computation of tags requires a lot of computation overhead and storage space. The homomorphic properties were also used to check the integrity of data [7]. For ensuring the remote integrity, the MAC and reedsolomon code were used [5].

### 2.5 Dynamic data operations

Many of the existing remote checking methods support only static data [13], [1], [6], [7], [5]. These are not featured with the methods for handling dynamically changing data. Several methods have been proposed for provisioning dynamic data in cloud [4], [15], [17], [8]. Among these, some of the papers are not offering the support for block insertion operations and are detecting the data corruption with a lesser probability [8]. For the high probability of detection of data corruption, it is needed for the increased number of challenges to the server from the client or TPA. These methods are not considering the issue of confidentiality. In this paper, a new method for assuring confidentiality and integrity of dynamically changing data is proposed.

Our scheme uses a stream cipher encryption algorithm called 2-Keys Symmetric Encryption [18] for protecting the confidentiality of data. This method generates the metadata for all the data blocks stored in the server for ensuring the integrity of data.

## 2.6 Secure storage and secure computation

To ensure the integrity of stored data in cloud, a scheme considering the positions of data has been suggested in [13]. And to ensure secure computation this method uses the Merkle hash tree for checking the correctness of computations done by the cloud service provider. This method is not featured for the dynamically changing data stored in cloud.

## 3 Problem definition

The major security issues in cloud computing are integrity and confidentiality of data stored at the servers. In cloud, the service providers and consumers should ensure that the data stored at the server is secure. In this paper a method for ensuring data integrity and confidentiality for dynamically changing data has been proposed. Dynamic data operations like insertion, deletion and appending are conceivable without retrieving the entire data from server by using the linked list data structure. Here public auditability is enabled by introducing a Third Party Auditor (TPA) without disclosing original data to the TPA.

### 3.1 System model

Our storage model consists of Data Owners (DO), n cloud storage servers (s1, s2., sn) under the direct control of Cloud Service Providers (CSP) and Third Party Auditors (TPA). Storage servers provide storage services.

**Data Owners**: Users having their data to be stored in cloud and depend on the CSPs for data computation. The client can be the individual user or an organization.

**Cloud Service Providers**: It can be the organization comprising many storage servers and offering significant storage space and computing resources.

**Third Party Auditors**: The Data Owner may call the Third Party Auditor to verify the integrity of data. The verifier's role falls into two categories.

**1. Private Auditability**: It permits the Data Owner to verify the integrity of data file stored in the server.

**2. Public Auditability**: It allows anyone including TPA to verify the integrity of data stored in the server.

### 3.2 Threat model

It is presumed that the TPA is honest. It executes honestly in the whole auditing process of checking the integrity of data. The data will not be leaked out to the third party auditor during the auditing process. But the server may conduct the following attacks:

**Replace Attack**: If the server discarded a challenged block or its metadata, it may choose another valid uncorrupted pair of data block and replace the original data block and metadata.

**Replay Attack**: The server generates the proof from the previous proof without retrieving the challenged data.
**External Attacks**: The external hackers who are capable of compromising the cloud servers can access the data stored in the server. He may delete or modify the data and may the leak the sensitive information.

### 3.3 System overview

To ensure the confidentiality and integrity of data stored in cloud, an efficient scheme has been proposed. The overall architecture of our system is described in the Figure 1. This system consists of four phases namely Setup phase, Verification phase, Dynamic data operations and Decryption phase.

#### 3.3.1 Setup phase

The Data Owner (DO) preprocesses the file F before storing it in cloud server. The setup phase consists of four steps.
(1). Key generation
(2). Binary sequence generation
(3). Encryption and
(4). Metadata generation.

**(1). Key generation**
The DO generates two symmetric keys by using the algorithm1. Two keywords (keyword1 and keyword2) of variable length are given as the input from the user for generating the keys. From the keywords two keys (key1 and key2) are generated by making use of the ASCII values of characters in the keywords and their position in the keyword. The two keywords are kept secret by the DO by storing it in the key store of data owner.

**(2). Binary Sequence Generation**
The DO generates the binary sequence consisting of 0s and 1s. This binary sequence is generated using the recurrence relation of the form as in the equation 1.

$$Xn+m = (C0Xn+C1Xn+1+C2Xn+2+\ldots Cm-1Xn+m-1) \bmod 2 \quad (1)$$

For generating the recurrence relation, the user needs an initial seed value m, the initial vector values like (X1, X2, X3, X4., Xm) and the coefficient values like (C0,C1,C2 C3.,Cm-1).For example, for the initial seed m=5, initial vector (0,1,0,0,0) i.e. X1=0; X2=1; X3=0; X4=0; X5=0 and for the coefficient (1,0,1,0,0) i.e. C1=1; C2=0; C3=1; C4=0; C5=0, m=5 the recurrence relation is like the equation 2.

$$Xn+5=Xn+Xn+2 \quad (2)$$

The binary sequence generated for the initial vector (0,1,0,0,0) and the coefficient (1,0,1,0,0) is 0100001001011001111100011011101010000 .

**(3). Encryption**
To ensure the confidentiality of data, the DO encrypts each data block using the 2-keys Symmetric Encryption algorithm [18]. It takes as input the data file F, binary sequence, key1,key2 and a secret random number for encryption and produces the cipher text in file F'. This

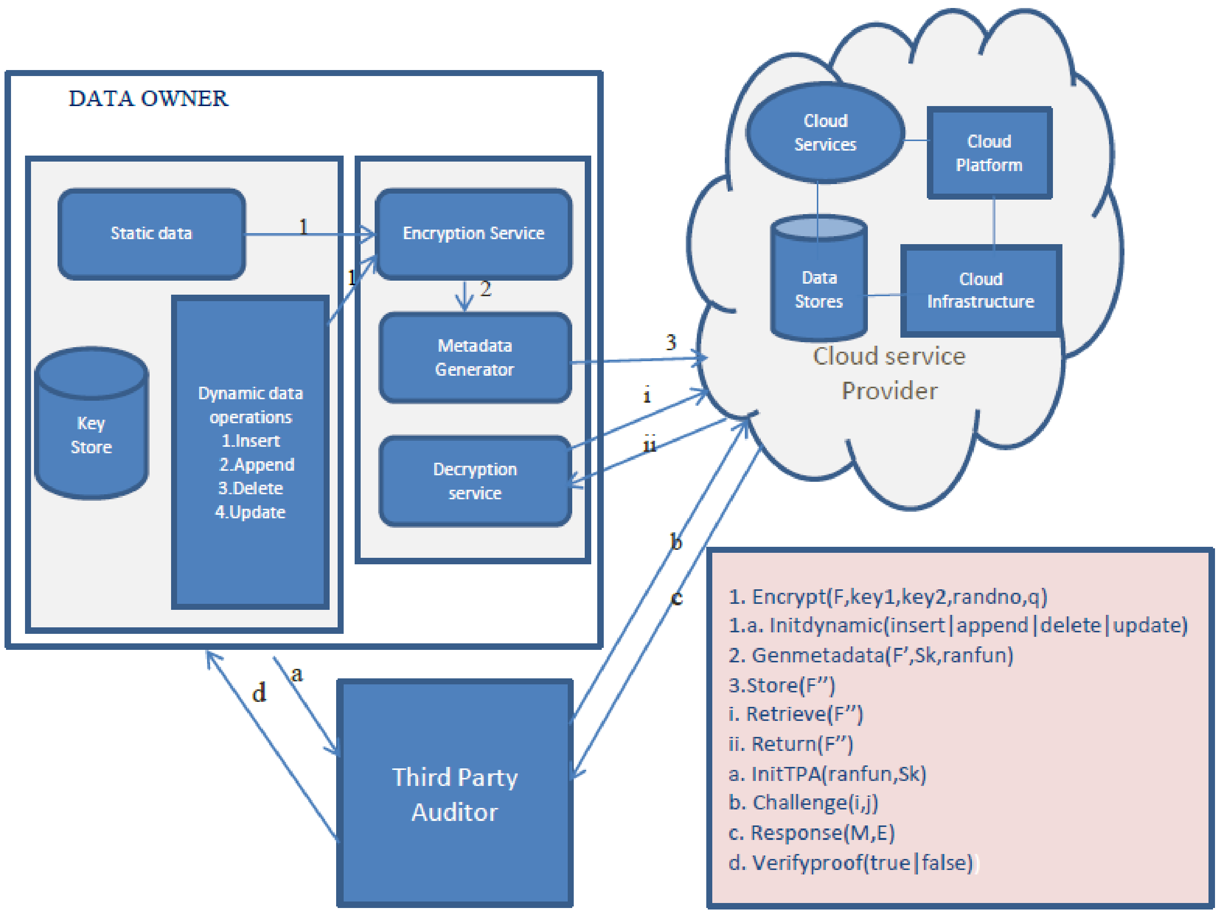


Figure 1: System Architecture Diagram.

encryption technique splits the data file **F** into **n** data blocks **db1, db2, db3., dbn** considering that each of the n data blocks contains s bytes.

**(4). Metadata generation**

After encrypting the data, the DO computes the metadata of the encrypted data blocks to ensure the integrity of data stored in the server. To generate the metadata the algorithm proposed in [13] can be used.

It is using some random functions involving the position of characters in the file **F'** and a secret key (**Sk**) chosen by the DO. The DO has to keep the random function as a secret one. The DO can issue the random function and the secret key (**Sk**) to the TPA for verifying the integrity of data.

An example for such random function f (i, j) to generate the Metadata M is

$$f(i,j)=M[i,j]=ASCII(F[i,j])*i*j*Sk. \quad (3)$$

The generated metadata is concatenated with the encrypted data of each block and is stored in the F''.

### 3.3.2 Verification phase

After storing data in the server, to ensure the integrity of data, our system depends on this phase. The Data Owner assigns this job to the Third Party Auditor (TPA). The DO submits {f, Sk} consisting of the random function, secret key used to generate metadata to the TPA. After receiving the key and the random function, the TPA creates a challenge message and sends it to the untrusted server. Upon receiving the challenge message from the TPA, the server generates response message and send it to the TPA. Using this, the TPA checks the integrity of the message as discussed with [13].

### 3.3.3 Dynamic data operations

The proposed scheme provisions dynamic data operations at the block level such as Modification, Insertion and Deletion operations. To achieve the dynamic data operations, we can use indexing [8], Merkle Hash Tree [14]. Here the simple data structure called the linked list is used. When generating metadata for the data file F, create a singly linked list such that each node in the list consists of the starting position of the new block .The linked list for the file will be stored with the data owner along with the keys for encryption. The DO can use the linked list to retrieve the original data in the correct order. The TPA and Cloud Server (CS) do not know about the linked list maintained with the DO.

**Construction of linked list**

For a file **F** containing **n** blocks with **BS** as the block size, linked list can be constructed as the Figure 2. Each node contains the starting position of the new block. This linked list is maintained with the DO for retrieving the data in order.

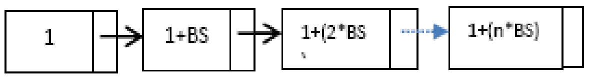


Figure 2: Linked List Construction.

### 3.3.3.1 Block insertion

In this, the Data Owner can insert a new block **db*** after the position **k** in the file **F'' = {db1, db2, db3., dbn}.** Usually the insertion operation changes the logical structure of the file **F''**. So, instead of inserting the block at the middle, append the block at the end and insert a node at the position **k** in the linked list maintained with the DO. This scheme performs the insertion operation without re-computing the metadata and encrypted data for all blocks that have been shifted after inserting a block. It calculates the encrypted metadata only for the block which we are going to insert. Block insertion can be done using the block insertion algorithm.

**Block Insertion Algorithm**

**Input**: Block to be inserted {**db***}, Position **k**

**Output**: Appended data at the server, Inserted Linked list at the DO

1. Get the new block to be inserted. (i.e.) **db*** and the position **k** after which it is to be inserted.
2. Perform the encryption for the new data block **db*** using the 2-keys symmetric encryption algorithm as explained in [18] considering the position **i**, the end of data file **F''**.
3. Generate the metadata of the encrypted block.
4. Append the metadata and encrypted data the end of the file **F''** in the server.
5. Insert a node in the linked list at the position k. The data at the node should be the position at which the data was appended in the server

The server contains the data as {db1, db2, db3., dbk-1, dbk, dbk+1,. db*,}.
The DO updates the linked list as in the Figure 3.

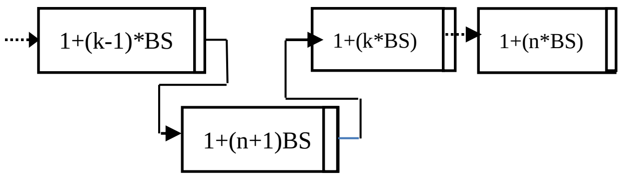


Figure 3: Linked list after Block insertion.

### 3.3.3.2 Block modification

Block modification can be simply done using our scheme. If the user wants to modify a block db2 at the position k with dbm, for the individual block dbm the encrypted data as well as the metadata can be calculated without affecting the other blocks in the server. After calculating the encrypted metadata, make an update request like **Update (F'', dbm, k)** to the server to modify the block db2 with dbm.

Upon receiving the update request **Update (F'', dbm, k)**, the server update the data as **{db1, dbm, db3., dbn}**. It is not required to update the linked list in the user side for block modification operation.

### 3.3.3.3 Block appending

Appending a block is also very simple in our method. To append a block dba at the end, the data owner computes the encrypted data and metadata without disturbing the other blocks. Then the user makes an append request **Append (F'', dba)** to the cloud server to append the data block dba at the end of the File **F''**.

The server appends the data block **dba** at the end as **{db1, db2, db3., dbn, dba}. T**he client updates the linked list as in the Figure 4.

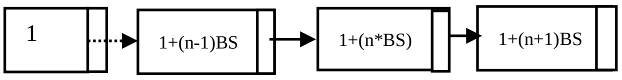


Figure 4: Linked list after Block Appending.

### 3.3.3.4 Block deletion

To delete a block at the position **k**, the user makes a delete request **Delete (F'', k)** to the server and the server deletes the encrypted data and the metadata at the position **k** and replaces the block with **null (Φ)**.For example to delete a block at the 2nd position, the server update the file as **{db1, Φ, db3., dbn**}. The client updates the node at the 2nd position with **Φ**.

### 3.3.4 Decryption phase

The user uses the secret key (Sk) and the reverse of the random function used to generate the metadata and recovers the encrypted data. From the encrypted data, the DO gets the original data using the decryption algorithm [18]. The DO generates the binary sequence from the recurrence relation, initial vector and the coefficients and the keys key1, and key2 from the keywords keyword1, keyword2 respectively for decryption. The DO divides the encrypted data file encdata.txt into **n** data blocks **db1, db2, db3., dbn** considering each n data blocks contains **s** bytes **like b1, b2, b3., bs.**

## 4 Security analysis

In this section, we present the security analysis for the integrity and confidentiality of the stored data in cloud.

### 4.1 Integrity

To assure the integrity, we need the following properties.

**Public Verifiability**: It permits anyone not just the DO to check the integrity of data. In our system, there is a Third Party Auditor (TPA), for integrity verification. So, this supports the public verifiability and also the private verifiability.

**Privacy of Data**: Because the verification is done at the TPA, we should assure the privacy of data such that no information should be leaked to the Third Party Auditor (TPA). In our scheme, the TPA does the auditing only in the encrypted data whose keys are maintained with the DO. So it is guaranteed that, no information is leaked to the TPA.

**Block less Verification**: No challenged file blocks should be fully retrieved by the TPA during the verification phase for security concerns. In our system, the TPA get only two characters (metadata and encrypted data) at the positions specified by the DO. No full block is retrieved from the server. Our System ensures block less verification.

**Low Computation**: Only a lesser computation should be done at the TPA to verify the integrity of the file.

**Low Storage Overhead at TPA**: The amount of storage in the TPA should be as small as possible to check the integrity. In our scheme, the TPA has to store only the random function and the secret key (Sk). So the storage at the TPA is less.

**Support Dynamic Operations**: After storing the data in server, the user can update the data dynamically. It should dynamic operations like block insertion, block modification, block deletion and block appending. Our system supports efficiently supports all the dynamic operations that is not discussed with [13].

**Probability of Detection of Data Corruption (Pd)**: The TPA should check the data corruption with high probability. We investigated the probability Pd based on the type of data corruptions done at the untrusted server. The data corruption can be classified into data deletion, data modification, data insertion and data appending. For data deletion, data insertion and data appending corruptions, our system detects the corruption with high probability of 1 with minimum number of challenges, because these operations change the position of characters in the file **F**.

To find the probability (Pd) of data replacement corruption, consider the following assumptions.

- The file **F** contains the **n** blocks and the probability to pick any block is **1/n**.
- The attacker modifies **m** blocks and the probability of modified block is m/n.
- The TPA makes **t** number of challenges to the server to detect the corruption.
- **s**-> the number of bytes in a block.

Based on these assumptions, probability detection of data replacement corruption is calculated by using the equation 4.

$$Pd=1-(1-m/n)\ ts \tag{4}$$

The Figure 5 illustrates the probability detection of data corruptions like data replacements, data appending, data insertion and data deletion for the file containing n=1000 blocks, s=20 sectors in a block and 1 corrupted block. This figure infers that the data appending, data insertion and data deletion corruptions are identified by the highest probability of 1 requiring minimum number of challenges. The data replacement corruptions are identified by somewhat high probability. In this the number of challenges is proportional to the number of corrupted blocks.

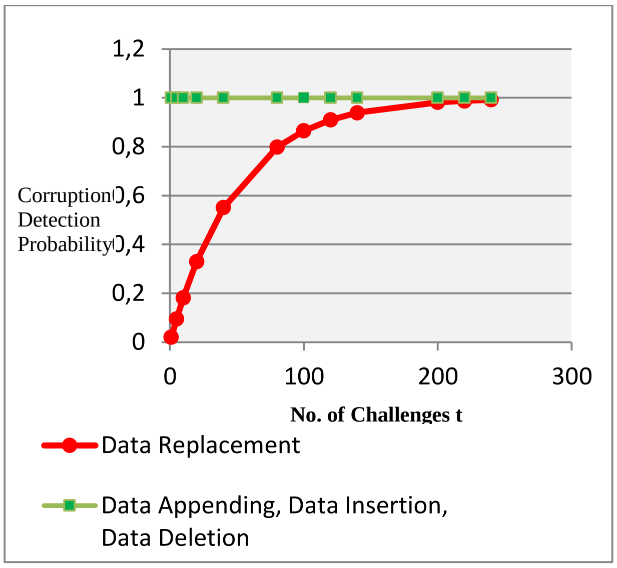


Figure 5: Probability of detection Pd of data corruptions for 1000 blocks, 20 sectors in a block and, 1corrupted block.

## 4.2 Confidentiality

We analyzed the confidentiality of our scheme. Here we have designed a symmetric stream cipher encryption algorithm. Stream ciphers are suitable for the larger data than the block cipher methods. In block cipher, the data is split into blocks. The same encryption algorithm and key are used for encrypting all the blocks in the data. If a same block is repeated again in the plaintext, in electronic code book (ECB) mode, we get the same cipher text for the repeated blocks. The modes of operations like Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB) are not suitable for dynamic data. So, it is decided that block ciphers are not suitable for larger dynamically changing data.

The existing stream ciphers are vulnerable to frequency analysis attack, brute force attack, correlation attack, algebraic attack known plain text attack, cipher text only attack, etc. One time pad encryption algorithm is getting relaxed from these attacks. But to generate the key, it is required to use the LFSR sequence, blub-blub generator, or recurrence relations for generating binary sequence. This binary sequence is prone to correlation attack. So we have used the 2-keys symmetric encryption algorithm which is freed from all the attacks.

## 4.3 Analysis of 2-Keys symmetric algorithm

**Brute Force attack:**

In this method, two keywords are used for encryption and also the keywords are of variable sizes. Along with the keywords, initial vector values. (X1, X2, X3, X4., Xm) initial coefficients (C0, C1, C2, C3., Cm-1) random number randno, number of characters in the block –q are all the secret information maintained by the cloud user.

So the brute force attack is not conceivable in this algorithm.

**Frequency Analysis Attack**

This cryptanalysis method computes the number of occurrences of characters in the cipher text and plain text. And it also compares their frequencies. Figure 6 shows the frequency of characters of the encrypted data.

**Correlation attacks:**

The attacks try to extract some information about the initial state from the output stream. Here, since the recurrence relation has been used to generate binary sequence, the attacker can try to get the initial seed (initial vector and initial coefficients) from the binary sequence. If the length of the initial seed is smaller, then the binary sequence will be getting repeated. Then it is possible for a hacker to get initial seed. In order to avoid this, the user must choose a larger initial seed.

## 4.4 Complexity of encryption function

The encryption function, we have proposed is

$$\text{Encrypt}[i,j] = (\text{key} + \text{ASCII}(F[i,j]) + \text{randno}(i+j)) \bmod 256 \qquad (5)$$

Key→the key generated from the keyword using the algorithm1.
ASCII ((F (i, j))→the ASCII value for the character at the **j**th position in the **i**th block.
Randno→any random number chosen by the DO.
Mod256→the result of the bigger value from the equation 5 is reduced to 256 so that the size of the cipher text never increases.
Randno ((i+j)) mod256→it is very difficult to find the values of **i** and **j** for a given plaintext-cipher text pair. It is based on the concept of discrete logarithm problem [16].
Let x, α, y and β are non-zero integers.
Suppose

$$\beta \equiv \alpha x \bmod y \qquad (6)$$

It is very difficult to find the value of x given α, y and β. Even though there is a method called Pohlig-Hellman algorithm [16] to compute the discrete log it is not possible in this algorithm to find the values of x, since it is changing for every character in the text.

## 5 Performance analysis

The performance in terms of storage, communication and computation complexity is analyzed.

### 5.1 Storage cost:

The storage cost of DO, TPA and CS (Cloud Server) are as follows.

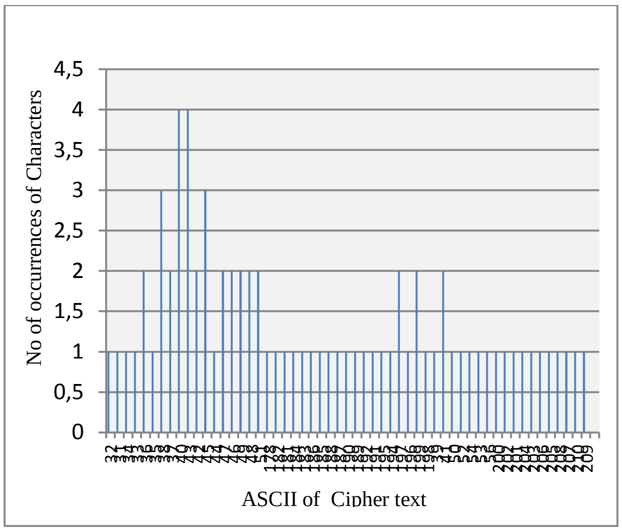


Figure 6: Frequency of characters of encrypted data.

**Data Owner**: client needs to store only the security parameters like keyword1,keyword2, initial seed, randno-q (the number of characters in the block), the random function and secret key (Sk) constantly. So the storage cost of the Data Owner is O(1).

**Server Side**: The server has to store the complete file containing encrypted data (n bits) along with the meta data (n bits) [13]. So the storage cost at the server side is O(2n). It is illustrated in the Figure 7.

The metadata is generated for all the characters in the plaintext depending on the position of characters in the file and the secret key (Sk). Then the metadata is appended to the data file. So the size of the data file becomes doubled. Even though the data size is increased, the client needs to store only the random function and the secret key (Sk). The data file along with the metadata is stored in the cloud server.

**Third Party Auditor**: The TPA or the verifier has to store only the random function and secret key (Sk) constantly. So the storage cost of the TPA O(1).

### 5.2 Computation cost

We analyzed the computation cost of the DO, TPA and the cloud server.

**Client:** The client generates the key1, key2 and the binary sequence of length n blocks. The cost of this computation is O(n). The cost of encryption function is O(n). The cost of metadata generation is O(n). So total computation cost at the user is O(n).

**Server:** The computation done at the server is very less. It has to send the response message consisting of encrypted data and metadata for the challenge message. The computation cost is O(1).

**TPA:** The computation done at the verifier is lesser. It has to generate the challenge message, and verify the integrity by doing the inverse of random function. The computation cost of TPA is O(1).

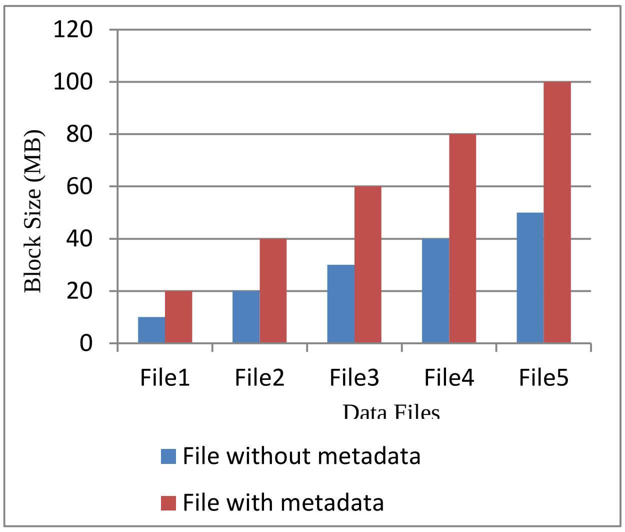


Figure 7: Comparison of file sizes with and without metadata.

### 5.3 Communication cost

The communication cost between the client and server is O(n), between the verifier and the server is O(1) and between the client and TPA O(1). The storage and computation costs are summarized in the Table 1.

Table 1: Storage and computation cost of our proposed scheme.

| Storage Cost | | | Computation Cost | | |
|---|---|---|---|---|---|
| DO | TPA | Server | DO | TPA | Server |
| O(1) | O(1) | O(2n) | O(n) | O(1) | O(1) |

## 6 Results

.A private cloud environment has been established with the open source eucalyptus cloud simulator. To install and configure an Ubuntu enterprise cloud, two Servers (Server1 and Server2) that run with 32-bits, 1GHz server version and two machines that run as a Desktop 32-bit version (Client and TPA) are required. The Ubuntu desktop version is installed on Client and TPA so that the browsers can be used to access the web interface of UEC. The experiment was conducted using a setup consisting of two servers and one desktop.

Encryption and decryption algorithms are implemented in java and communication between client and server is implemented with the java Remote Method Invocation (RMI) concepts. We compared our scheme with the existing remote integrity checking methods. The comparison analysis of our proposed method with the existing methods is illustrated in the Table 2. It shows that our scheme is the one which offers the highest probability of corruption detection. And also if there are corruptions like appending, deletion, and insertion of malicious data in the data stored in cloud, our scheme detects the corruptions with the highest probability of 1.

Probability of detection of data replacement corruption is illustrated in Figure 8. From this it is inferred that, the probability of detection of data replacement corruptions is higher in our proposed system and in the papers [8], [13], [5], [2]. But for the other data corruptions like data deletion, data insertion and data appending, our proposed system has the highest probability detection of 1 that was not achieved through the previous integrity checking methods.

## 7 Conclusion

In this paper, various security challenges in cloud environment have been analyzed and an appropriate solution for providing confidentiality and ensuring integrity of dynamically changing data has been proposed by using an efficient block cipher, 2-Keys Symmetric Encryption technique with the linked list data structure. This scheme can also be applied for the secure storage of bulk data. After the detailed study, it is analyzed that it is the first method that detects the data corruptions with the probability of 1. Since this system requires less computation and communication cost, it can be used for large-scale cloud storage systems. This can be further extended by finding a method that applies this mechanism for the security of unstructured data.

Table 2: Comparison of our proposed scheme with other integrity checking protocols.

| **Issues** | **[13]** | **[14]** | **[8]** | **[13]** | **[6]** | **[7]** | **[2]** | **Proposed** |
|---|---|---|---|---|---|---|---|---|
| **Confidentiality** | No | Yes | Yes | No | No | No | No | Yes |
| **Public Verifiability** | Yes | Yes | Yes | No | No | No | Yes | Yes |
| **Data Dynamics** | Yes | Yes | Yes | No | No | No | Yes | Yes |
| **Server Computation cost** | O(log n) | O(log n) | O(ts) | O(1) | O(t) | O(t+s) | O(t logn) | O(1) |
| **Verifier computation cost** | O(log n) | O(log n) | O(ts) | O(1) | O(t) | O(t+s) | O(t logn) | O(1) |
| **Probability of corruption detection (insertion, deletion, appending)** | $1-(1-p)^t$ | $1-(1-p)^t$ | $1-(1-p)^{ts}$ | 1 | $1-(1-p)^t$ | $1-(1-p)^{ts}$ | $1-(1-p)^{ts}$ | 1 |
| **Probability of corruption detection (data replacement)** | $1-(1-p)^t$ | $1-(1-p)^t$ | $1-(1-p)^{ts}$ | $1-(1-p)^{ts}$ | $1-(1-p)^t$ | $1-(1-p)^{ts}$ | $1-(1-p)^{ts}$ | $1-(1-p)^{ts}$ |

n- the number of blocks in the file, t- number of challenge requests to server, s- number of sectors in a block
p-probability of corrupted blocks

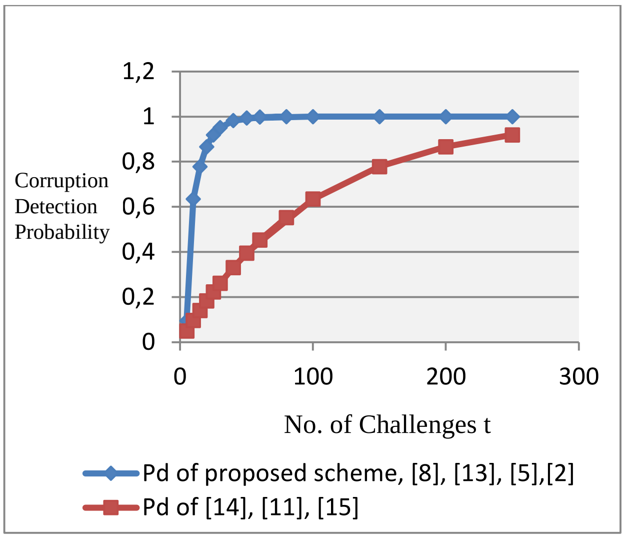


Figure 8: Probability of detection Pd of data replacement corruptions for 100 blocks, 10 sectors in a block and 1corrupted block.

## References

[1] A. Juels and B.S.Kaliski, (2007), “Pors: proofs of retrievability for large files,” in Proceedings of the 14th ACM conference on Computer and communications security, New York, NY, USA: ACM, pp. 584–597.

[2] C. Wang., Q. Wang., S.S.M.Chow., K. Ren., and W.Lou, (2013), “Privacy-Preserving Public Auditing for Secure Cloud Storage”, IEEE Transactions on Computers, Vol 62, No.2.

[3] C. Wang., Q. Wang., K. Ren., and W. Lou, (2009), “Ensuring Data Storage Security in Cloud Computing,” in Proc. Of IWQoS’ 09.

[4] C. Wang., Q. Wang., K. Ren., and W. Lo, (2010) “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing” in Proc. IEEE INFOCOM, pp. 525-533.

[5] E.-C.Chang., and J. Xu, (2008), “Remote integrity check with dishonest storage server” in Proc. Of ESORICS’08.Berlin, Heidelberg: Springer-verlag, pp. 223– 237.

[6] G. Ateniese., R. Burns., R. Curtmola., J. Herring., L. Kissner., Z. Peterson., and D. Song, (2007), “Provable data possession at untrusted stores,” in Proceedings of the 14th ACM conference on computer and communications security. New York, NY, USA, pp. 598– 609.

[7] H. Shacham., and B. Waters., (2008), “Compact Proofs of Retrievability” in Proc. 14th Int’l Conf. Theory and Application of Cryptology and Information Security:Advances in Cryptology, pp. 90-107.

[8] Kan Yang, and Xiaohua, (2013), ” An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing”, IEEE Transactions On Parallel and Distributed systems, VOL 24, NO. 9, pp.1717-1726.

[9] Lifei Wei., Haojin Zhu., Zhenfu Cao., and Weiwei Jia, (2010), “SecCloud: Bridging secure storage and computation in cloud”, in ICDCS’10.

[10] M. Armbrust et al., (2009), "Above the clouds: A berkeley view of cloud computing", ECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28,.

[11] M. Jensen et al., (2009), "On Technical Security Issues in Cloud Computing," in IEEE International Conference on Cloud Computing, Bangalore, India, pp. 109-116.

[12] Pearson (2009), “Taking Account of Privacy when Designing Cloud Computing Services”, in Proceedings of ICSE-Cloud’09, Vancouver.

[13] Ponnuramu Veeralakshmi, and Latha Tamilselvan, (2012), “Data Integrity Proof and Secure Computation in Cloud Computing’. J. Comput.Sci., 8: 1987-1995.

[14] Q. Wang., C. Wang., J. Li, K. Ren., and W. Lou, (2012), “Toward Secure and Dependable Storage Services in Cloud Computing,” IEEE Transactions On Services Computing, VOL. 5, NO. 2, pp.220-232.

[15] Q. Wang., C. Wang., J. Li, K. Ren., and W. Lou, (2011), “Enabling public auditability and data dynamics for storage security in cloud computing,” IEEE Transactions On Parallel and Distributed systems,VOL. 22, NO. 5, pp.847-858.

[16] W. Stallings, (2007), Cryptography and network security principles and practice, Fourth edition, Prentice hall

[17] Y. Zhu., H. Hu., G. Ahn., and M. Yu, (2012) “Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage”, IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 12, pp. 2231-2244.

[18] Veeralakshmi Ponnuramu., and Latha Tamilselvan, (2014), ”Encryption for Massive Storage in Cloud” in Computational intelligence in Data Mining, Volume 2,pp 27-38, Smart Innovation, Systems, and Technologies(Springer), volume 32.

# A Novel Video Steganography Algorithm Based on Trailing Coefficients for H.264/AVC

Yingnan Zhang, Minqing Zhang, Xu An Wang, Ke Niu and Jia Liu
Key Laboratory of Network and Information Security of CAPF, Electronic Department
Engineering University of the CAPF, Xi'an, Shaanxi, 710086, P. R. China
E-mail: zyn583@163.com, wangxazjd@163.com

**Keywords:** information security, video, steganography, trailing coefficients

**Received:** December 22, 2015

*With the development of high-speed networks, life is more convenient than ever. However, the information security issue of high-speed networks is still a big problem. As an important branch of information security, steganography is a useful method to protect confidential information. In this paper, by combining the trailing coefficient produced in the process of the quantization ofthe H.264 encoding standard with the current video steganographic algorithms, we implement a new kind of algorithm based on trailing coefficients. The algorithm firstly conducts a DCT transform on the frame, and then it obtains the trailing coefficient for each quantized DCT block;lastly, the secret information bit is embedded into the video. The experimental result indicates that this algorithm has little influence on video quality and has a large capacity of steganography;also, it has a strong anti-steganalysis capability and high robustness.*

*Povzetek: Opisan je nov algoritem za video steganografijo.*

## 1 Introduction

Currently, people are enjoying a high standard life due to the continuing growth of network bandwidth. In the past, people were unable to easily upload or share large digital content because of the bandwidth restrictions on networks. With the advancement of various technologies such as 4G/5G, the network speed has substantially increased, allowing people to share almost whatever they want. In this paper, we explore the concept of high bandwidth to design new video steganographic algorithms. As an important branch of information security, information hiding, also called steganography, has provided an efficient method to protect the security of sensitive information.

Steganography is a technique that can be used to transmit secret information publicly via digital media, while achieving covert communication [1]. There has been much advancement made in image steganography algorithms. However, because of the limited capacity of digital images, the capacity of secret information that can be embedded in the image is also restricted. Compared with a digital image, video has more advantages regarding steganography, such as it can support a larger capacity and there is more redundancy, a high communication quality, robustness, etc. As a new standard, H.264 has more advantages over the previous ones. There is a better compression of digital TV broadcasting, video real-time communication, network video streaming and multimedia messaging. Some of its biggest features are high reliability and high compression efficiency [2]. As a result, the study of steganographic methods based on video for H.264/AVC has become a popular research topic[3-8].

In the past, when video was transmitted on a network, it was transmitted by frames, like pictures, and an attacker would be able to observe the network's package transmitting. When an attacker would find many video frames in the network, he/she would pay attention to them; this situation,shown in Figure 1, is not an ideal situation for video steganography. Now, since there are very high-speed network and supported techniques [9-13], people can transmit video content easily. This does not gain the attacker's attention, as shown in Figure 2. It also satisfied the original intention of steganography-to hide the existence of secret information-so our algorithm can achieve a high security standard in networks when protecting sensitive information.

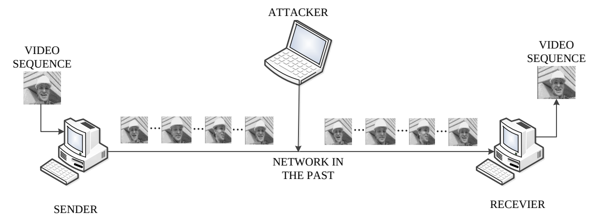


Figure 1: Transmitting video sequence in a network with low bandwidth.

When embedding secrets in the spatial domain, we found that it is easy to be detected by many steganalysis algorithms, so we chose to embed secrets in DCT coefficients. Furthermore, we believe that using the characteristic of the H.264 coding standard is a better choice because we can

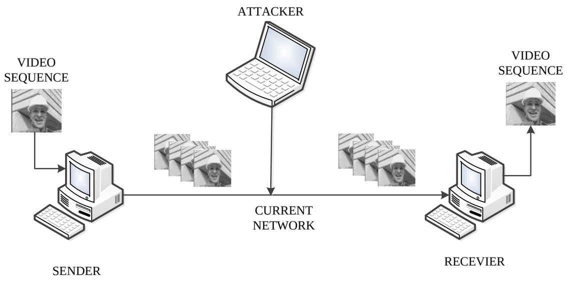


Figure 2: Transmitting video sequence in a current network.

embed secrets during the compression process. We save the time that being cost in embedding procession, thus reducing the complexity of our algorithm.

Hua et al. proposed a video steganography algorithm based on H.264/AVC [14]. The algorithm can be implemented to achieve good embedding and extracting, but the algorithm is weak when it is under attack. Langelaar proposed a mechanism for the compressed video stream [15]. The advantage of this algorithm is that it only needs part of the coding stream. As a result, this algorithm can achieve a higher data embedding capacity, but the algorithm is weak regarding anti-steganalysis detection. Ma proposed a novel algorithm based on H.264 [16],and it improves the visual quality, but the embedding efficiency and embedding capacity needs improvement. He also proposed a new method based on a motion vector [17], but the capacity of embedding is also limited. Zhang proposed a robust video watermarking algorithm for H.264/AVC based on texture features [18], it has little impact on the video quality and bit rate, but it can only achieve little capacity for embedding. Liu proposed a method based on macro-block segmentation [19], but the bit rate increase is very low, and it has weak anti-steganalysis detection. Ultimately, almost all of the steganography algorithms based on video for H.264/AVC exist some problems such as negative impacts on video quality, high complexity in embedding, and less capacity of embedding.

In this paper, we present a new video steganography algorithm by modifying trailing coefficients through certain rules to embed secret information. Experimental results showed that this algorithm has better visual invisibility, while improving the steganographic capacity, strong anti-steganalysis ability, and high robustness. Our algorithm's sketch is shown in Figure 3.

## 2 Trailing coefficients

Trailing coefficients are produced from CAVLC. By simply denoting, the trailing coefficient can be a number in the range of 0-3 and the amplitude can be 1. When the number of coefficients satisfying this condition is more than 3,

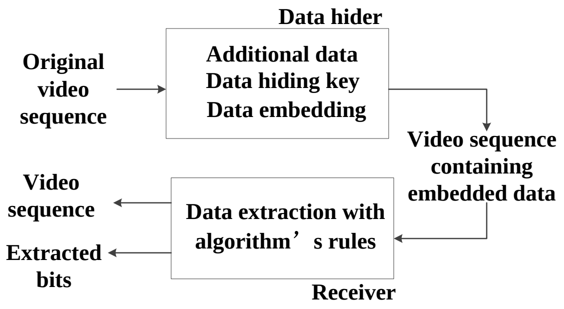


Figure 3: Sketch of proposed scheme.

we only choose the last three trailing coefficients; the other coefficients are considered normal non-zero coefficients.

Here is an example below: 3, 2, 1, 0, 2, -1, 1, -1, 0, 1, -1, 0, 0, 0, 0, 0, where the number of the trailing coefficients is 3, the value is -1, 1, -1 (8th, 10th, and 11ththat were zigzag scanned), and the other coefficients (3rd, 6th, and 7th that were zigzag scanned) are normal non-zero coefficients.

## 3 Proposed scheme

### 3.1 Trailing coefficients' application in steganography

Generally speaking, there are 15 kinds of trailing coefficients in DCT blocks, as shown in Table 1. As can be seen

Table 1: 15 kinds of trailing coefficients.

| | | | | | |
|---|---|---|---|---|---|
| 1 | none | 6 | -11 | 11 | -111 |
| 2 | 1 | 7 | -1-1 | 12 | 1-1-1 |
| 3 | -1 | 8 | 111 | 13 | -11-1 |
| 4 | 11 | 9 | 11-1 | 14 | -1-11 |
| 5 | 1-1 | 10 | 1-11 | 15 | -1-1-1 |

from the table, the trailing coefficients' distribution follows certain regularity. As a result of entropy coding, trailing coefficients will not be affected by the compression. Therefore, there is an advantage of selecting trailing coefficients as points in information hiding.

### 3.2 Embedding algorithm

– **Step 1.** Pretreatment: use $K$ to generate a pseudo-random sequence, $p$, and preprocess it as the following formula:

$$f_i = m_i \oplus p_i \tag{1}$$

where $m_i$ is the original secret information and $f_i$ is the sequence of post-processing. The purpose is to reduce the correlation of secret information and improve safety. This method can achieve real-time processing;

– **Step 2.** Perform a DCT transform on frames, then traverse all the DCT trailing coefficients and arrange the blocks in order;

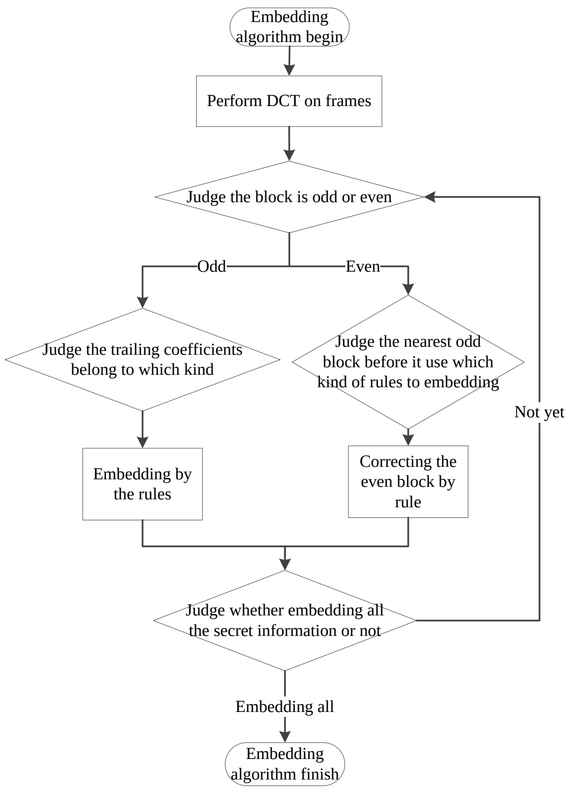

Embedding algorithm begin
Perform DCT on frames
Judge the block is odd or even
Odd
Even
Judge the trailing coefficients belong to which kind
Judge the nearest odd block before it use which kind of rules to embedding
Not yet
Embedding by the rules
Correcting the even block by rule
Judge whether embedding all the secret information or not
Embedding all
Embedding algorithm finish

Figure 4: Flow of embedding algorithm.

- **Step 3.** Use the odd-numbered blocks as embedding blocks and the even-numbered blocks as correcting blocks. Simply speaking, if the current block is embedding secret information, the following block is a correcting block;

- **Step 4.** Embedding process. Determining whether the summary value of trailing coefficients is positive or negative is the key point to our algorithm. We want the value to be negative when the secret information bit is 0, and the value should be positive when the secret information bit is 1. The embedding rules are as follows:

  1. *The embedding rules of the odd blocks.* When the secret information bit is 0, the rule is as shown in Table 2. When secret information bit is 1, the rule is shown in Table 3.

Table 2: The embedding rules when the secret information bit is 0.

| | | |
|---|---|---|
| 1→ -1 | -1-1→ -1-1 | 1-1-1→ 1-1-1 |
| -1→ -1 | 111→ 1-1-1 | -11-1→ -11-1 |
| 11→ -1-1 | 11-1→ 1-1-1 | $-1-11 \rightarrow$ -1-11 |
| 1-1→ -1-1 | 1-11→ 1-1-1 | $-1-1-1 \rightarrow$ -1-1-1 |
| -11→ -1-1 | -111→ -11-1 | $0 \rightarrow$ -1 |

As can be seen from the above tables, when modifying the DCT coefficients, the max number that can be modified is 2. When modifying 0, we choose the first 0 scanned by zigzag after the last non-zero numbers; the rest situations in Table 2 and Table 3 are only modified a number or without modification, so the algorithm can achieve better security.

Table 3: The embedding rules when the secret information bit is 0.

| | | |
|---|---|---|
| 1→ 1 | -1-1→ 11 | $1-1-1 \rightarrow$ 1-11 |
| -1→ 1 | 111→ 111 | $-11-1 \rightarrow$ -111 |
| 11→ 11 | 11-1→ 111 | $-1-11 \rightarrow$ -111 |
| 1-1→ 11 | 1-11→ 111 | $-1-1-1 \rightarrow$ -111 |
| -11→ 11 | -111→ -111 | $0 \rightarrow 1$ |

  2. *The rules of even blocks for correcting.* When an odd block has been modified, we will use the next even block to correct. If we change -1 to 1 to embed the secret, then we change the last 1 of a correction block to -1. If there is no existing1 in the correcting block, we change the first 0 bit which after the last non-zero numbers to -1. In the odd block, if we change 1 to -1, then we change the last -1 in the correcting block to 1; if there is no existing -1 in the correction block, change the first 0 bit which after the last non-zero numbers to 1.If we change 0 to 1 in odd block to hide the secret information, then the last 1 in the corrected block is changed to 0; if there is no existing 1, nothing is modified; if we change 0 to -1 in the modified block to hide the secret information, the last -1 in corrected block is changed to 0; if there is no existing -1, nothing is modified; the purpose of this correction is to hold the histogram between stego-DCT coefficients' and cover-DCT coefficients';

- **Step 5.** Repeating Step 4, until all the secret information is embedded. The flow of the embedding algorithm is show in Figure 4.

## 3.3 Extraction algorithm

- **Step 1.** Perform a DCT transformation on frames, and traverse all the DCT trailing coefficients, then arrange the blocks in order;

- **Step 2.** Follow the rules of extracting, based on the following Formula:

$$m_i = \begin{cases} 0, & if \;\; S(j) < 0 \; and \; j \bmod 2 = 1 \\ 1, & if \;\; S(j) > 0 \; and \; j \bmod 2 = 1 \end{cases} \quad (2)$$

Where $m_i$ is the secret information bit, $j$ is the order of the DCT block based on Step 1, and $S(j)$ is the sum-value ofthe trailing coefficients;

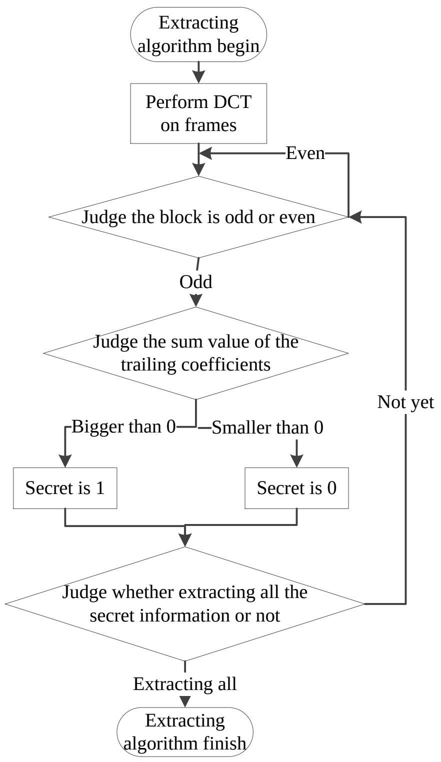

Extracting algorithm begin
Perform DCT on frames
Even
Judge the block is odd or even
Odd
Judge the sum value of the trailing coefficients
Not yet
Bigger than 0
Smaller than 0
Secret is 1
Secret is 0
Judge whether extracting all the secret information or not
Extracting all
Extracting algorithm finish

Figure 5: Flow of the extracting algorithm.

- **Step 3.** Repeat Step 2 until all the secret information is extracted;

- **Step 4.** Inverse pretreatment; the rules are shown in the following Formula:

$$m_i = f_i \oplus p_i \tag{3}$$

Where $m_i$ is the original secret information and $f_i$ is the sequence of extracting information. The flow of the extracting algorithm is shown in Figure 5.

## 4 Experimental results and analysis

Our experimental environment is based on X.264, VC++ 2008, and Matlab2008. The video sequences are downloaded from the website, "media.xiph.org". Each sequence is 15 frames/s, the bit rate is 396kbit/s, and the format is QCIF (News, Mobile), CIF (Container, Carphone). The secret information is the image "lena.bmp."

### 4.1 Theoretical analysis

The probability that changes the coefficient -1 to 1 can be expressed as follows:

$$\begin{aligned}
&P((-1)\to(1)) = p(m(i)=1)\{p((-1)\to(1))\\
&+p((1,-1)\to(1,1))+p((-1,1)\to(1,1))\\
&+2p((-1,-1)\to(1,1))\\
&+p((1,-1,-1)\to(1,-1,1))\\
&+p((-1,1,-1)\to(-1,1,1))\\
&+p((1,-1,1)\to(-1,1,1))\\
&+2p((-1,-1,-1)\to(-1,1,1))+p((0)\to(1))\}\\
&+p(m(i)=0)\{0\}
\end{aligned} \tag{4}$$

The probability that changes the coefficient 1 to -1 can be expressed as follows:

$$\begin{aligned}
&P((1)\to(-1)) = p(m(i)=0)\{p((1)\to(-1))\\
&+p((1,-1)\to(-1,-1))+p((-1,1)\to(-1,-1))\\
&+2p((1,1)\to(-1,-1))\\
&+2p((1,1,1)\to(1,-1,-1))\\
&+p((1,1,-1)\to(1,-1,-1))\\
&+p((1,-1,1)\to(1,-1,-1))\\
&+p((-1,1,1)\to(-1,1,-1))\}\\
&+p(m(i)=1)\{0\}
\end{aligned} \tag{5}$$

Since the secret information is encrypted, the probability of 0 and 1 remains the same, as shown below:

$$p(m(i)=1) = p(m(i)=0) \tag{6}$$

We can conclude from formula (4),formula (5), and formula (6), that we can obtain the probability $p((1) \to (-1)) = p((-1) \to (1))$. Therefore, this method can achieve high security.

### 4.2 Analysis of invisibility

#### 4.2.1 Subjective analysis of visibility

The original and embedded frames of test sequences are shown in Figure 6. As can be seen, there is no significant difference between them. So, we can conclude that our algorithm is better in terms of visibility.

#### 4.2.2 Objective analysis of visibility

The PSNR value is the key to judging the visibility. According to the HVS,when the PSNR value is above 30dB, the sequence is clear and fluent. The PSNR value of the test sequence has been shown in Table 4, and the results show that the decrease is low after embedding, and the average decline of the PSNR value is about 1.156dB.

Since the Carphone sequence is rich in texture blocks, and there are also many smoothing blocks, we calculate out pre-30 frames' PSNR value of it. Figure 7 shows the PSNR

(a) News original image (b) News image after being embedded

(c) Mobile original image (d) Mobile image after being embedded

(e) Container original image (f) Container image after being embedded

(g) Carphone original image (h) Carphone image after being embedded

Figure 6: Comparison on the original and embedded frames of test sequences.

Table 4: Comparison of PSNR values before and after embedding.

| Test sequence | Original PSNR value/dB | PSNR value after embedding/dB | Decrease /dB |
|---|---|---|---|
| News | 36.743 | 35.945 | 0.798 |
| Mobile | 35.376 | 34.347 | 1.029 |
| Container | 36.232 | 35.089 | 1.143 |
| Carphone | 35.798 | 34.141 | 1.657 |

value of the pre-30 frames of Carphone after embedding secret information. As can be seen, the impact of the first frame is about 2.27dB, and the impact of the No.16 frame is about 1.89dB; the other frames are not affected much.

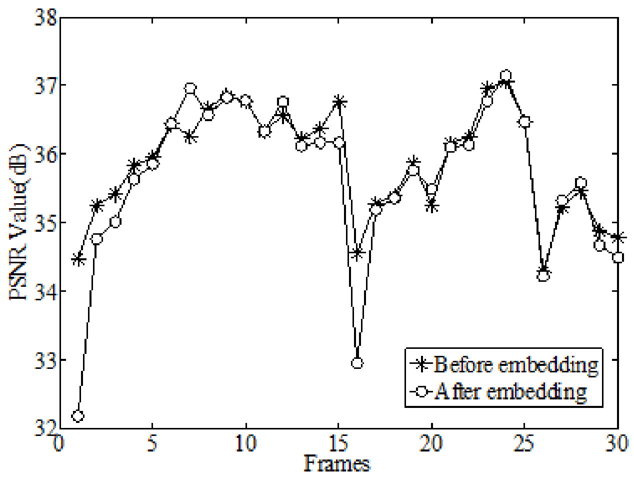


Figure 7: Video image and extraction of the secret information image after attack.

## 4.3 Steganographic capacity

Reference [16] proposed a steganographic algorithm based on modifying the DCT coefficients, and reference [19] also proposed an algorithm based on modifying the DCT coefficients for hiding, so we use these two references to be compared with our algorithm.

As can be seen from Figure 8, our algorithm's capacity has been improved.Because of our algorithm embedding secret information in half of the DCT blocks, it can achieve an improved capacity.

## 4.4 Robustness testing

The main goal of the robustness test of the steganographic algorithm is to detect anti-attack capability. For this test, we use salt and pepper noise and Gaussian low-pass filtering.

In the experiment, we add salt and pepper noise to the video sequence, and the intensity is 0.05; Figure 9 shows

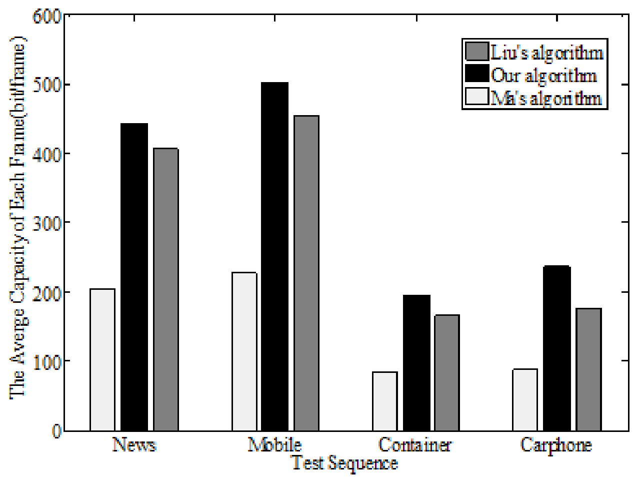


Figure 8: Video image and extraction of the secret information after attack.

the effect it has on News and the secret information image we extract.

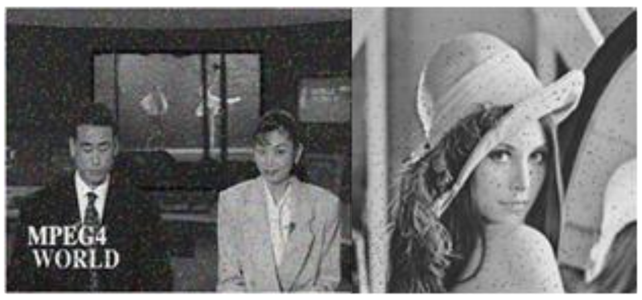


Figure 9: Video image and extraction of the secret information image after attack.

Figure 10 represents the experiment after a 3 × 3 Gaussian low-pass filter, the effect on Mobile, and the secret information image we extract.

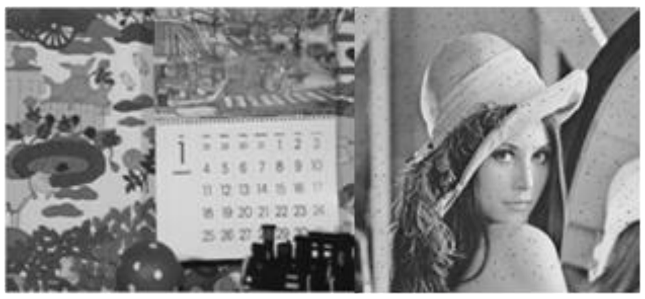


Figure 10: Video image and extraction of the secret information after attack.

Table 5 is the SIM value between the secret information after attack and the original one. The formula of SIM is as follows:

$$SIM(X,Y) = \frac{\sum_i X(i)Y(i)}{\sqrt{\sum_i X(i)^2}\sqrt{\sum_i Y(i)^2}} \tag{7}$$

In the formula, $X(i)$ and $Y(i)$ represent the original image and the stego-image to be evaluated in the one-dimensional sequence of pixel values respectively; $SIM \in (0,1]$ takes the value of 1 only when the images we compared are exactly the same.

Table 5: The SIM value of extracting secret information after attack.

| Video sequence | Add salt and pepper noise | Add Gaussian low-pass filtering |
|---|---|---|
| News | 0.827 | 0.899 |
| Mobile | 0.846 | 0.943 |
| Container | 0.813 | 0.921 |
| Carphone | 0.790 | 0.906 |

Figure 9, Figure 10, and Table 5 indicate that the algorithm has high robustness and anti-attack capability.

## 4.5 Steganalysis detection

Heidari has proposed a steganalysis algorithm [20], and it has a high detection rate for the steganography algorithm based on modifying the DCT coefficients. Therefore, our video sequence would be extracted for each frame, and we then we test the detection rate of every frame.

A false positive (FP) represents classifying the non-stego frames as stego frames. A false negative (FN) indicates classifying the stego framesas non-stego frames [21]. The results can be shown in Table 6:

Table 6: The results of using Heidari's algorithm.

| Video | FP (%) | FN (%) | Error rate(%) |
|---|---|---|---|
| News | 53.45 | 48.12 | 50.79 |
| Mobile | 46.51 | 45.08 | 45.80 |
| Container | 52.67 | 49.59 | 51.13 |
| Carphone | 47.75 | 40.64 | 44.20 |

Generally, the higher the FN and FP is, the better the steganographic algorithm is. As can be seen from Table 6, when detecting our algorithm,FP and FN are high.

## 5 Conclusion and discussion

In the past, transmitting a huge video sequence would take a great amount of time, so the video sequence is not common in steganography.Usually,the image is used as the cover carrier, but it also limits the information bit that is to be hidden. However, a high speed network offers a platform to transmit large multimedia, so we can use it to hide more information than before.

This paper proposed a video steganography algorithm based on trailing coefficients in a high speed network, and we modified the value of trailing coefficients to make sure that when the secret information bit is 0, the sum value is negative, and when the secret information bit is 1, the sum value is positive. In order to ensure that the DCT coefficients of the cover video after embedding have been changed, the algorithm used the method that modified the odd-numbered blocks to hide and modified the even-numbered blocks to correct. The experimental results indicated that our algorithm has little impact on the video's visual invisibility, the capacity of steganography is improved, and it has high robustness. Therefore, our research has better performance compared to previous algorithms.

Although this paper has proposed a scheme to protect the security of a high-speed network, and our algorithm can achieve some good features as mentioned before, some problems should also be discussed.For example, if we hide secret information in images or videos, no method to completely recover the hidden media has been found. In future studies, there will be a focus on revising the process of hiding information.

The next inadequacy of our method is that we did not use the protocol of a high-speed network; if we hide information in it, we will raise the capacity greatly and increase the cost of bandwidth. In future studies, the goal will be to find a method of hiding information in the protocol of networks, but without greatly increasing the cost of bandwidth.

## Acknowledgment

The authors would like to thank the anonymous reviewers for the constructive comments. This work was supported by the National Science Foundation of China: Theory research on extracting and optimizing feature of image steganalysis (Grant No.61379152). And it was also supported by the National Science Foundation of China: Cross-Domain Human Action Recognition by Knowledge Transfer (Grant No.61403417).

## References

[1] H. G. Zhang, R. Y. Du, J. M. Fu, B. Zhao, L. N. Wang. (2014) *Information security: an independent discipline a new subject.* Information and Communication Security, no.5 (in Chinese).

[2] H. J. Bi (2005). *A New Generation of Video Compression Standard—H.264/AVC.* Beijing: Posts & Telecom Press, pp. 110-111 (in Chinese).

[3] R. J. Mstafa, K. M.Elleithy (2014). *A Highly Secure Video Steganography using Hamming Code (7, 4).* 2014 IEEE Long Island Systems, Applications and Technology Conference (LISAT), pp.1 - 6.

[4] M. M. Sadek, A. S.Khalifa, M. G. M.Mostafa (2014). *Video steganography: a comprehensive review.* Multimedia Tools & Applications, vol.74, pp.1-32.

[5] J. Ridgway, M. Stannett (2014). *Developing a Video Steganography Toolkit.* Eprint Arxiv.

[6] K. P. Divya, K. Mahesh (2014). *Random Image Embedded in Videos using LSB Insertion Algorithm.* International Journal of Engineering Trends & Technology, vol. 13, no.8, pp.381-385.

[7] K. Churin, J. Preechasuk, C. Chantrapornchai (2013). *Exploring Video Steganography for Hiding Images Based on Similar Lifting Wavelet Coefficients.* Advances in Information Technology. Springer International Publishing, vol. 409, pp.35-46.

[8] H. Gupta, S. Chaturvedi (2014). *Video Steganography through LSB Based Hybrid Approach.* International Journal of Computer Science and Network Security (IJCSNS), vol.14, no.3, pp. 99-106.

[9] Y. Shen, Q. Pei, Q. Xu, Z, Zhang (2012). *The multimedia service session handoff method in heterogeneous wireless networks.* International Journal of Grid and Utility Computing, vol. 3, no. 1, pp. 68-77.

[10] D. V. Bernardo, D. B. Hoang (2012). *Multi-layer security analysis and experimentation of high speed protocol data transfer for GRID.* International Journal of Grid and Utility Computing, vol. 3, no. 2/3, pp. 81-88.

[11] Y. Wang; J. Du; X. Cheng; Z. Liu; K. Lin (2016). *Degradation and encryption for outsourced PNG images in cloud storage.* International Journal of Grid and Utility Computing, vol. 7, no. 1, pp. 22-28.

[12] J. Kolodziej, F. Xhafa (2011). *Supporting situated computing with intelligent multi-agent systems.* International Journal of Space-Based and Situated Computing, vol. 1, no. 1, pp. 30-42.

[13] R. Pereira, E. G. Pereira (2016). *Future internet: trends and challenges.* International Journal of Space-Based and Situated Computing, vol. 5, no. 3, pp. 159-167.

[14] G. L. Hua, Z. B. Li, B. Feng (2013). *Low frequency steganography algorithm for H.264/AVC.* Journal on Communications, vol. 34, no. Z2, pp. 47-50.

[15] G. C. Langelaar, R. R. Lagendijk (2001). *Optimal differential energy watermarking of DCT encode images and video.* IEEE Transactions on Image Processing, vol. 10, no.1, pp. 148-158.

[16] X. J. Ma (2010). *The Research on Video Data Hiding Algorithms Based on H.264/AVC.* Huazhong Univercity of Science and Technology (in Chinese).

[17] X. S. He, Z. Luo (2008). *A Novel Steganographic Algorithm Based on the Motion Vector Phase.* International Conference on Computer Science and Software Engineering CSSE, Wuhan, China, pp. 822-825.

[18] W. W. Zhang, R. Zhang, Y. J. Liu , et al (2012). *Robust Video Watermarking Algorithm for H.264/AVC Based on Texture Feature.* . Journal on Communications, vol. 33, no.3, pp. 82-89 (in Chinese).

[19] C. H. Liu, O. T. Chen (2008). *Data Hiding in Intra Prediction Modes of H.264/AVC.* IEEE International Symposium on Circuits and Systems, pp.3025-3028.

[20] Heidari, Mortaza, G.Shahrokh (2013). *Universal image steganalysisusing singular values of DCT coefficients.* 10th International ISC Conference on Information Security and Cryptology (ISCISC), Yazd, Iran, pp.1-5.

[21] T. Filler, J. Judas, J. Fridrich (2011). *Minimizing additive distortion in steganography using syndrome-trellis codes.* IEEE Transactions on Information Forensics and Security, vol. 6, no.3, pp. 920-935.

# An Ontology-Based Context Model to Manage Users Preferences And Conflicts

Salima Bourougaa Tria
Tebessa university, Tebessa, Algéria
E-mail: bourougaasalima@gmail.com

Hassina Seridi-Bouchelaghem
Annaba University, Algéria
E-mail: h_seridi@yahoo.com

Farid Mokhati
Oum-el-bouaghi university
E-mail: mokhati@yahoo.fr

**Keywords:** context of use, user profile, web-based information system, nomadic environment, ontology, user preferences, conflicts, ubiquitous computing, web service

**Received:** April 22, 2015

*In the last decade, ubiquitous computing (UC) has become an aspiration of the computing community. Nowadays, it is so profound that it is increasingly indistinguishable from the overall agenda of computing research. In UC, the main objective is to provide users the ability to access services and resources anytime, anywhere, in particular using Mobile Devices (MD). Applications in this domain are sensitive to the context. They have to be able to perceive this context and to adapt their behaviours by considering data that deals with the context of use and user preferences. Actually, ensuring access by nomadic users to information Systems through various devices and the adaptation of responses to nomadic users profile and context of use are two bound problems. In this paper, we attempt to answer to these problems and we propose a novel approach allowing essentially: (1) representing the context and preferences of nomadic users through ontology, to support context representation and reasoning (2) resolving conflicts that may arise between user preferences and, (3) adapting such applications to the context of use and user's profile. The approach is supported by a visual tool we developed. A case study is presented to give more illustration.*

*Povzetek: Opisana je uporaba konteksta s pomočjo ontologije za preference in profile mobilnih uporabnikov.*

## 1 Introduction

Currently, Web users access to a large mass of various data situations through distinct devices, to have answers to their requests that are usually very numerous, from multiple sources of information (heterogeneous and remote). Such answers are not all equally interesting and relevant, and they do not answer all the user's wishes, which may decrease the user satisfaction. This complexity is increased if the user is nomadic (user who frequently changes localization) and appealed the SIW (System Information on the Web), anywhere and anytime via mobile devices (PDAs, phones, laptops) because the change of localization, for example, causes a change in working conditions and consequently a change in the general context of use. Consequently, developers are incited to integrate these mobiles devices into their applications, giving rise to new information systems called pervasive or ubiquitous [1]. In this case, these applications must considering the user's situation called contextual situation. This latter includes the context of use as well as information on its profile. Adapting all application's behaviors, in order to return to users relevant responses (i.e. while considering content and time), is the subjacent idea of ubiquitous computing, where applications are sensitive to the context (context-aware applications) [2].

Actually, ensuring access by nomadic users to information Systems through various devices and the adaptation of responses to nomadic users profile and context of use are two bound problems. Dealing with these problems requires answers to the following questions:

- How to perceive the user's context?
- How to model the context of use and the nomadic user profile?
- How to resolve conflicts that may arise between user's preferences?
- How to adapt the context-aware application behavior to satisfy the needs of these mobile users?

In order to answer these questions, we propose, in this paper, a novel approach which essentially allows: (1) to model the context of use and the user's preferences using a developed ontology "Contology", basing on a new definition of the context which separates application data from contextual data. The ontology is useful to support context representation and reasoning, and the Dynamic requirements can be defined as context constraints and need to be supported by context reasoning features of the ontology, and they are most expressive and most promising for context description in an environment sensitive to the context. (2) To resolve conflicts that may occur when managing user's preferences, we propose to model conflicts and their solution in the ontology as rules by using the semantic web rule language (SWRL). Finally, to ensure the dynamic functional adaptation of context-aware applications, Web Service based architecture is proposed to show the effectiveness of our proposal in the context model.

The remainder of this paper is organized as follows: In Section 2, we give a brief overview of major related works. Section 3 outlines the motivation for using ontology, while section 4 presents the context model and the conflict management. We explain the ontology process building in Section 5. Section 6 details the context rules description and the ontology implementation is given in section 7. Section 8 details the adaptation process (ontology exploitation). We present a case study in section 9. Finally, we discuss our actual research, draw some conclusions and give some future work directions.

## 2 Related works

We distinguish four categories of context modeling approaches. The first category consists in storing the context by using key-value pairs (attribute, value) or by using a set of triplets. Famous examples of this category are: Context Toolkit of [3] and approaches used by [4] .The second category of the model-oriented approaches includes: (1) Markup Scheme Models: represent the context by using RDF. For example: CC/PP [5], [6] and ConteXtML [7], (2) Graphical Models: use UML (Unified Modeling Language) to model the context. For example: ContextUML [8] and CML [9], (3) Object Oriented Models use principal advantages of the modeling object. For example: Active Object Model [10] and the TEA project CUES [11]. The third category represents the context by a logic-based model. The context is defined like facts, expressions and rules. An early representative of this approach type is: ' Extended Situation Theory' [12], [13] and [14]. (4) The last category models the context by using ontologies. The most referred modeling are: CoOL [15] , SOUPA [16] , [17] a formal context model based on ontology using OWL to address issues including semantic context representation, context reasoning and knowledge sharing, context classification, context dependency and quality of context , [18] and [19] COBRA-ONT an ontology to support pervasive context-aware systems. COBRA-ONT, expressed in the Web Ontology Language OWL, is a collection of ontologies to describe places, agents and events and their associated properties in an intelligent meeting-room domain. [20] an intelligent web portal to serve as a service provider in the airlines travelling tasks , [21] a metadata model encoding semantic tourism destination information in an RDF-based P2P network architecture. The model combines ontological structures with information for tourism destinations and peers, [22] an approach based on ontologies provide the elements and guidelines to define and create a user profile in any multimedia domain. In order to describe the multimedia context and ontologies of PUMAS a framework based on the agents [23], [14] and [1].

In [24] and [6], we find a synthesis on the characteristics of the context modeling approaches and this let us deduce that in spite of the principal disadvantage of the ontology approaches which is the ontology's complexity execution and the reasoning weight on their facts and their entities. They are most expressive and most promising for context description in an environment sensitive to the context. This is our motivation to choose ontology in context modeling in this work. Those works have considerably forwarded the domain by proposing novel strategies to context modeling. However, they omit some important aspects which can be summarized as follows: firstly, none of existing ontologies of context modeling separate between the context data and the applications data. According to [25] and [6], this separation is very necessary to a reliable modeling of context. Also, the user's preferences management was only considered by PUMAS [23], [14] and [1]. Although, it represents a very important point to satisfy the user and to return him answer adapted to its context. Finally, the conflict's resolution is considered only by PUMAS [23]. It defined some conflicts and presents their corresponding solutions. But this approach does not solve this problem, because it has not considered all conflicts which can arise during the user's preferences checking.

## 3 Motivations for using ontology

The main goal of the proposed approach is to model the context of the user by use of a semantic representation and resolve conflicts that may arise during these preferences verification. This proposed context modeling objective is to adapt the initial request of use to this context, to have a contextual query, used to give to user a response adapted to his context. We opted, in the context of this work, for the use of ontologies for the advantages they procure. They provide the means to describe semantically information, share described data, easily to be used by other applications and to extend the initial description when new needs arise. Ontology languages can create expressive, scalable, reusable, sharable models, and on which we can reason using inference engine. OWL [26] for example, is a W3C recommended language to describe ontologies. It provides a simple and effective means based on an XML description model to share described data, reasoning about these data and adding axioms to describe specific relationships between information. Finally, ontologies are most expressive and most promising to context description in an environment sensitive to the context [24], [1].

In existing context-aware systems, notations like XML, XMbased CC/PP [27], UML [28], Topic Maps [29] and OWL [30], [31] are used in context modeling. We use the OWL to formalize context relationships based on the underlying DL representation. The choice of OWL is motivated by its reasoning support. It provides a logical language support to reasoning (OWL-DL) and supports Semantic Web Rule Language (SWRL) to enable rule-based reasoning [1]. The logical language (DL) supports context composition and context constraints enhancements. OWL facilitates the sharing of conceptualizations. The core elements of the DL used as an underlying abstract language shall be introduced. The Attributive Language with Complements (ALC) is the basis of many DL languages. The OWL-DL, the DL variant of OWL corresponds to SHOIN(D) [32], a DL language based on ALC with transitive roles, role hierarchies, nominals (enumerated classes of object value restrictions), inverse properties, cardinality restrictions and concrete data types[1]. In order to encode context aspects in SHOIN(D), and eventually in OWL-DL, an introduction of the constructors for SHOIN(D) is necessary. Their semantics is based on the usual interpretations of first-order logic. C denotes concepts, and R denotes property relationships. A DL specification can be constructed as a set of axioms. The basic constructors of SHOIN(D) can be used with either the subsumption or equivalence ≡ symbols to create DL statements. Axioms can be terminological axioms (TBox) or assertional axioms (ABox). Terminological axioms (statements about entities such as concepts and roles, but not individuals) can be subsumption or equivalence axioms. Assertional axioms (pertain only to individuals) can be concept assertions or role assertions axioms. A Subsumption axiom gives necessary conditions for some a concept tobe included (Subclassed) in another, e.g. A $\sqsubseteq$B where A, B are concepts. An equality axiom has the form A≡ B. A concept assertion is of the form C(i ) where C is a concept from a TBox and i is an individual. A role assertion is of the form R(a, b), where R is some role from a TBox and a and b are individuals.

# 4 The Context model representation

We will describe how we can define the context concepts. For the development of our Context Ontology "Contology", we used "METHONTOLOGY" [33]. According to [33], it is important to bear in mind that knowledge acquisition is an independent activity in the ontology development process. However, it is coincident with other activities. Most of the acquisition is done simultaneously with the requirements specification phase, and decreases as the ontology development process moves forward. Experts, books, handbooks, figures, tables and even other ontologies are sources of context from which the context can be elucidated using in conjunction techniques such as: brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools. In our approach the knowledge is the context of the user. The used techniques in the Context acquisition are: (1) Non-structured interviews with experts, to build a preliminary draft of the requirements specification document. (2) Informal text analysis, to study the main concepts given in books and handbooks. This study enables to fill in the set of intermediate representations of the conceptualization. (3) Formal text analysis. The first thing to do is to identify the structures to be detected (definition, affirmation, etc.) and the kind of knowledge contributed by each one (concepts, attributes, values, and relationships). (4) Structured interviews with experts to get specific and detailed knowledge about concepts, their properties and their relationships, to evaluate the conceptual model once the conceptualization activity has been finished, and to evaluate implementation. (5) All given definitions of context given by researchers and experts of context-awareness domain.

## 4.1 The context definition

Researches in the context-awareness domain have not yet led to a generic and pragmatic definition of context. Several definitions for the context were advanced [34], [35], [36], [25], [6]and [1].The definitions issued so far are very abstract or very specific to a particular domain, making the formalization of the context very difficult. The [3] definition is widely accepted as a "good" definition. According to [25], this definition does not help in separating the contextual data from the application data, and the core of the application should be designed in a context in dependent way. This separation separating the contextual data from the application data, and the core of the application should be designed in a context in dependent way. This separation according to [25] is very important, before beginning the design of an application sensitive to the context. A data defined as contextual in a field can be a data application in another field. For example, GPS localization is part of application data in a traffic regulation system, but is part of context data in a telemedicine application. Separation between the contextual data and the application data is also important in modeling context. [25] define the context as: ' the set of the external parameters that can influence the behavior of the application by defining new views on its data and its available services". Consequently, in the determination of the most descriptive concepts of information which constitutes the context, we chose the separation of the contextual data of the application data according to the definition of [25] of the context, because it seems to us relevant and generic. According to this definition, we can divide the concepts of context into two parts: the concepts which represent the context of use of a user and the concepts which represent the user profile. The context of use in our approach presents the set of data which allows indicating the situation of the user when it connects to the ubiquitous application. For example, it is represented by the following concepts: The user; the session; the used mobile device (MD) and location of the user. The user profile is presented by a set of preferences of user. We detailed these concepts in the following sections.

## 4.2 The Context representation: preferences, conflicts

Among the concepts of the user's context, we find the preferences. In this part we will define the concept of preference of the user and we detail a classification of different types of preferences. We will detail the concept of conflict and we will present its causes and solutions.

### 4.2.1 Preferences

By the concept of user preference, we refer to a set of descriptions covering what the user likes to receive as services, also the display of results choice. We define two types of preferences: Requested Service Preferences and Display Preferences and five conflicts.

#### a. Requested Service Preferences

Describe how the user chooses its services in the system. We define this type of preferences as follows: During his first contact with our system, the user can define the contents of each of his preferred services. The user can define from the beginning when he asks the service "S" what implies automatically the contents: C1, C2 ....Etc.

**Service (S) $\rightarrow$contents (C1, C2,……. etc.)**

As example to illustrate our proposition let us consider a user in travel who wants to have the list of restaurants in his entourage. He prefers that this list is displayed as a map. His user profile can, for example, specify that when it executes the service "consultation list of restaurants. » this user is only interested in restaurants offering dishes which respect his diet, because he has health problems. Thus, the preference says that user wants to execute the service "S" = "consultation list of restaurants" whose content is C = "restaurants that offer adequate food.", and preferably in the form of display image. Therefore, the preference of requested service is represented as follows:

**Requested_Service_Preferences(S, {content},**
**{** associated_ Requested_Services**}).**

***S***: is the service which the user wishes to carry out in the system. ***{Content}****:* is a list of the contents defined by the user from his first contact with our system. ***{Associated_Requested_Service}*** is a list of the associated services which the user wants to execute if he asks the service S. As example, we consider that a user wishes to execute service "S" which consists of one or several *contents* and possess one or several *associated_ RequestedServices*. Every time a teacher consults "the list of the planned meetings ", he wishes to know the meetings of the current week. Also, he executes associated_RequestedServices "possibility meeting", to see the possibilities of fixing a meeting between teachers by specifying the day, the hour and the list of the concerned teachers, and the associated_ Requesed_Services "the other possible dates " to know all the possible dates of meeting of one or several teachers (days and hours free).We can represent the data: Requested_Service_Preferences as follows:

S1 = Possibility meeting (list of teaching concerned, day, hour). S2 = the other possible dates (free day, free hours, list teachers). C1= meetings of the current week. Then, the Requested_Service_Preferences is presented as follows:

***Requested_Service_ Preference (S: "the list of the planned meetings ", {S1, S2}, {C1})***

In the following, we present the display preferences.

#### b. Display Preferences

Display Preferences describe how the user wants the information to be displayed on his MD (for example, the user only wants information in text format). At every service is associated a Display preference. It is represented as follows:

**Display _Preference (format, characteristics)**

Format which can take the value: "video", "text", "image", "sound". Each format is based on a set of characteristics. Following sections, detail the conflict in our approach, present their causes and details there solutions.

### 4.2.2 Conflict

By conflict we refer to problems which can arise during the verification of user preferences. For example, "Contradiction between the display preferences and the characteristics of used MD", this conflict can arise when user requests a display which is not supported by his used MD. For these problems (conflicts) that we will define later, we offer some solutions to solve them. At every type of conflict is associated a solution. It is represented as follows:

**Conflict (Type, Solution, Suggestion)**

***Type****:* represent the conflict which can arise. ***Solution****:* allows defining how to take action to resolve the conflict that occurred. ***Suggestion****:* represents the proposal of the user in cases where the system cannot find a solution to the conflict that occurred.

Our approach manages five conflicts which can be arising between the user preferences during the check of these last ones. The following two tables present our proposal to conflicts resolution. Table1 presents the conflicts and their causes, while Table2 presents the conflicts and their solutions in our proposal.

| **N° Conflict** | **Conflict** | **Cause** |
|---|---|---|
| 1 | a. Contradiction between TheRequested_Service_Preferences and access rights of the user | • The user requests a service which does not suit with these access rights. |
| 2 | b. Contradiction between the display preferences and the characteristics of used MD | • The user requests a display which is not supported by his used MD. |
| 3 | c. Various wishes of Display for the same service. | • This conflict can arise in two cases: **a.** The user did not specify display preferences. **b.** Display preferences are not suitable to the characteristics of MD. In these two cases the system will returns to the Context Ontology "Contology" for resolve it. |
| 4 | d. Absence of display preferences after checking the historic of the user. | • The user cannot specify display preferences, in this case the system will return to the historic of the user, and it cannot find display preference for favorite service. |
| 5 | e. Contradiction between the Display preferences requested and display capabilities expressed | • The user can request the service in a format not offered by the system. For example, if the user wants a list of restaurants in card format, while the system has this information in text format only. |

Table 1: Conflicts and Causes.

| Conflict | Solution |
|---|---|
| 1 | • The system returns to the user to inform him that he has not the right to access these services and asks consequently, suggestions for this problem. If the user does not give suggestions, the system stops. |
| 2 | • Our approach execute one of the following cases:<br>**a.** Uses the ontology "Contology" for searching and reasoning about a solution for the conflict, using the information of the precedents sessions, to extract the display preferences that agrees with the characteristics of the used MD.<br>**b**. if no, Returns to the user and demands suggestions.**c**. if no in the 2 previous alternative, he takes a default display preference which suits with the characteristics of the used MD. |
| 3 | • We propose using an arithmetic operation that gives us the number of specification of every encountered preference. The system will perform a comparison and it will retain the preference which has the maximum number of specification by the user. In the case of equality between preferences, we propose to use a default preference which suits with the characteristics of MD used. |
| 4 | • The system executes one of the following cases:<br>**a**. It returns to the user and asks for these suggestions, **b**. It uses a default preference. |
| 5 | • In this case the system executes one of the following cases:<br>**a**. Uses the ontology "Contology" for searching and reasoning about a solution for the conflict, using the information of the precedents sessions, to extract the display preferences that agrees with the characteristics of the used MD.**b**. it returns to the user and asks these suggestions,<br>**c**. if no in the 2 previous alternative, he takes a default display preference which suits with the characteristics of the used MD. |

Table 2: Conflicts and Solutions.

After detailing the context acquisition, defining what means context in our work, and presetting the context representation. In the following section we will present the ontology process building based on the method "METHONDOLOGY".

## 5 Ontology process building

This section presents the steps followed to build the ontology of context "ContoLogy", for this, we use a construction process in the development of the ontology starting from raw knowledge and arriving at an operational ontology represented by OWL. The main steps of this process are based on the methodology of ontology construction "METHONTOLOGY" [33] which is the basic support for the conceptualization of the ontology to create, through a series of semi-formal intermediate representations. The logic descriptions, is the used formalism to express the semi-formal ontology. OWL language for defining ontologies is chosen to codify the ontology using the Protégé OWL ontology editor. Finally, the inference RACER (Renamed Abox and Concept Expression Reasoner) system is used to test the consistency of the ontology throughout the development process. This process consists of five steps: (1) Specification of Requirements, (2) Conceptualization, (3) Formalization., (4) Ontology implementation, (5) Test & evolution of ontology. We start this part by the motivation of the build method choice. Then, we detail the steps process.

## 5.1 Ontology method build choice

Born of the needs of knowledge representation, ontologies are currently at the center of the research in knowledge engineering. The construction of ontology requires both a study of human knowledge and the definition of representation languages and the realization of systems to handle them. The knowledge engineering has given birth to the ontological engineering, where the ontology is the key item that needs to be addressed. Several studies propose methods of constructing ontologies. In this case, we have study some methods for creating ontologies such as: ENTERPRISE [37], TOVE [38] and METHONTOLOGY [33] and we present a comparative study in order to choose a method.Table3 summarizes the comparable study on the various methodologies and methods. Each cell in of the table may be filled with three types of values. Value "++" means that the method or methodology describes how to execute each task in the proposed activity (specification, conceptualization….)? When to do? Who should do it? ... Etc. The value "+" means that the just methodology identifies the process. The value "-" means that public documentation does not mention the activity.

“METHONTOLOGY” is the approach that provides the most precise descriptions of each activity. Most approaches are carried on of the activities of development, particularly on the implementation of the ontology, and they not interested in furthering other important aspects related to the management, development and evaluation of ontologies. This is because the field of conception of ontology is a relatively new field. However, low conformity with the formally established criteria does not mean poor quality methodology or method. The most approaches have drawbacks. According to table2, we choose “METHONTOLOGY” for the construction of our Context Ontology.

| **Criterias of comparison** | **TOVE** | **ENTER-EPRISE** | **METHO-NTOLOGY** | **OTK** |
|---|---|---|---|---|
| Specification | ++ | + | ++ | ++ |
| Acquisition of knowledge | + | + | ++ | ++ |
| Conceptualisati-on | ++ | - | ++ | + |
| Formalisation | ++ | - | ++ | ++ |
| Evaluation | + | + | ++ | + |
| supports tools | specifi c tools | specific tools | ODE, WebODE,Protégé-2000 | OntoE-dit |

Table 3: Comparison of methods for developing ontologies [39] [40].

## 5.2 Specification and Requirements.

The goal of the specification phase is to produce either an informal, semi-formal or formal ontology specification document written in natural language, using a set of intermediate representations or using competency questions, respectively. See figure1

**ONTOLOGY REQUIREMENT SPECIFICATION DOCUMENT**

**Domain** : context-aware application ( Ubiquitous applications)
**Date** : January, 15th 2014
**Conceptualized-by** : authors
**Implemented-by**: authors
**Purpose**: Context modeling ontology in context-aware applications to be used by our architecture of adaptation based on Web service.
**Level of Formality**: Semi- formal.
**Scope**: List of 33elements of substances:
**List of concepts** : ContextModel, ApplicationContext, ServicesApplication, ConflictContext………..etc
**At least information about the following properties:**IsConceredBy, HasSugg, AttachedTo, CausedBy, OccuredIn,
**Sourcesofknowledge**: Definitions of the context in the domain of context-aware applications.

Figure 1: Ontology Requirements Specification.

## 5.3 Conceptualization

In this step, we will structure the domain knowledge in a conceptual model that describes the problem and its solution in terms of the domain vocabulary identified in the ontology specification activity [Fernandez, 1997]. This phase comprises several stages which are: the Construction of: (1)Terms glossary,(2)Concepts classification diagram, (3)Binary relations diagram,(4)Dictionary concepts, (5)Tables of binary relations, (6) Attributes table, (7)Logical axioms table, (8) Instances Table.

### a) Construction of Terms Glossary:

This glossary contains the definition of all the terms relating to the field (concepts, instances, attributes, relations) which will be represented in final ontology, we have **128** terms, for example: UserContext and ContextModel are concepts, PreferredBy and CoveredByrepresent relations,…etc.The table4 provides an example of some used terms in the ontology:

| **Name of the term** | **Synonyms** | **Description** |
|---|---|---|
| ContextModel | The model of context | • Model all the concepts of the context related to the ubiquitous environment. |
| ApplicationContext | • - | • Represent the ubiquitous application |
| ServicesApplication | • - | • Represent the services offered by the application in question. |
| ……….. | • ……… …… | • ………………. |

Table 4: Glossary of Terms.

## b) Concepts Diagram

In this step, we build the diagram classification of concepts. The classification hierarchy of concepts demonstrates the organization of ontology concepts in a hierarchy that expresses the relationships in the sub-class (see figure2). A universal concept "Thing" that generalizes all the roots concepts of the different concept hierarchies is used to form one global hierarchy. To build the taxonomy of concepts, METHONTOLOGY proposes to use the four relationship,s: *Subclass-Of*, *Disjoint-Decomposition*, *Exhaustive-Decomposition*, and*Partition*. A concept C1 is a subclass of concept C2 if and only if every instance of C1 is an instance C2. for example, *CauseConflict* is a subclass of *ConflictContext*. A *Disjoint-Decomposition of a* C is a set of subclasses of C which not cover C and do not have common instances.

For example, the concepts: DevicesPreferences and NetworkPreferences constitute a *Disjoint-Decomposition* of the concept PreferencesContext. *Exhaustive-Decomposition* of a concept C is a set of subclasses of C which cover C and may have common instances.A *Partition* of a concept « C » is a set of subclasses of C which cover C and may have common instances have no common instance. For example, the concept CauseConflict and SolutionConflict constitute a *Partition* of the concept ConflictContext. Figure.2 presents the concepts classification diagram.

## c) Binary Relations Diagram:

A binary relation is used to connect two concepts together (a source concept and a target concept). This activity consists in building a binary relationship diagram (see figure3) which allows representing graphically the various relations existing between the various concepts of the same or different hierarchy.

## d) Concepts Dictionary:

The concept dictionary contains the domain concepts. For each concept we define its known Concept name, Instances, Attributes instance, Relationships (see table5 for some concepts).

| **Concept name** | **Instances** | **Attributes Instance** | **Relationships** |
|---|---|---|---|
| ContextModel | - | IDContMod Description | - |
| ConflictContext | Conflict1, conflct2 Conflict3, conflict4 Conflict5 | IDConf DescripConfl | HasSugg AttachedTo CausedBy OccuredIn |
| CauseConflict | C1,C2,C3,C4,C5 | IDCause DescripCause | HasSolution |
| SolutionConflict | S1,S2,S3,S4,S5 | IDSolution DescripSolution | ConcernCause |
| ……… ………… | …… ……. | ……… …… | ………… … |

Table 5: Concepts Dictionary.

## e) Table of Binary Relations:

This table defines for each relation used in the diagram of binary relations: Name relationship, Source concept, source cardinality (max), Target concept and inverse relationship (see table 6 for some relations).

| **Name relationship** | **Source concept** | **Source cardina-lity, (max)** | **Target concept** | **inverse relation-ship** |
|---|---|---|---|---|
| IsConceredBy | ServicesApplication | • N | Requested Service Preferences | Concern |
| HasSugg | ConflictContext | • N | Conflict Suggestion | Concern-Conf |
| Attached To | ConflictContext | • N | Display Preferences | Occur |
| …… ………… … | …… ………. | … … | …… …… | … …….. |

Table 6: Table of Binary Relations.

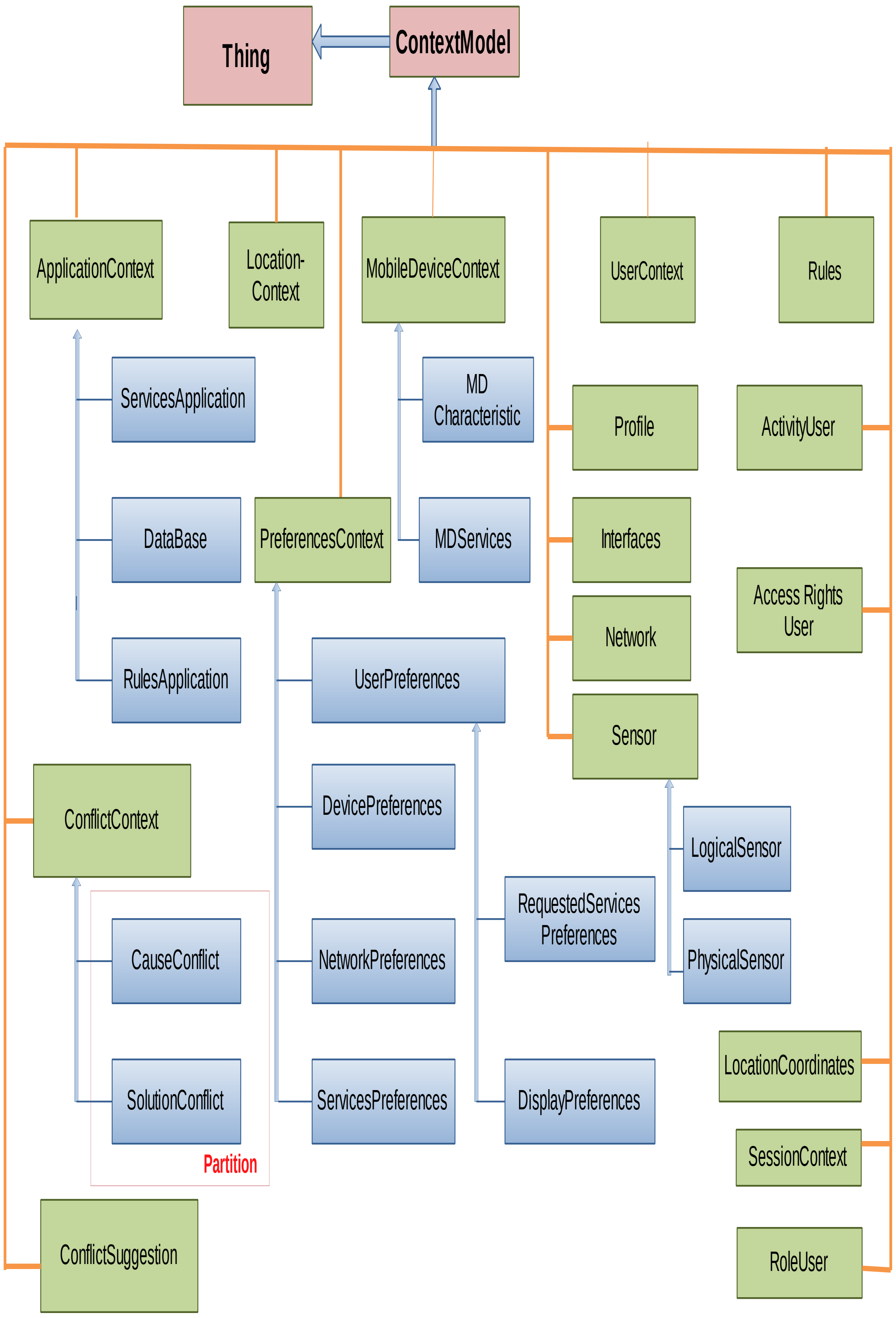


Figure 2: Concepts Classification Diagram.

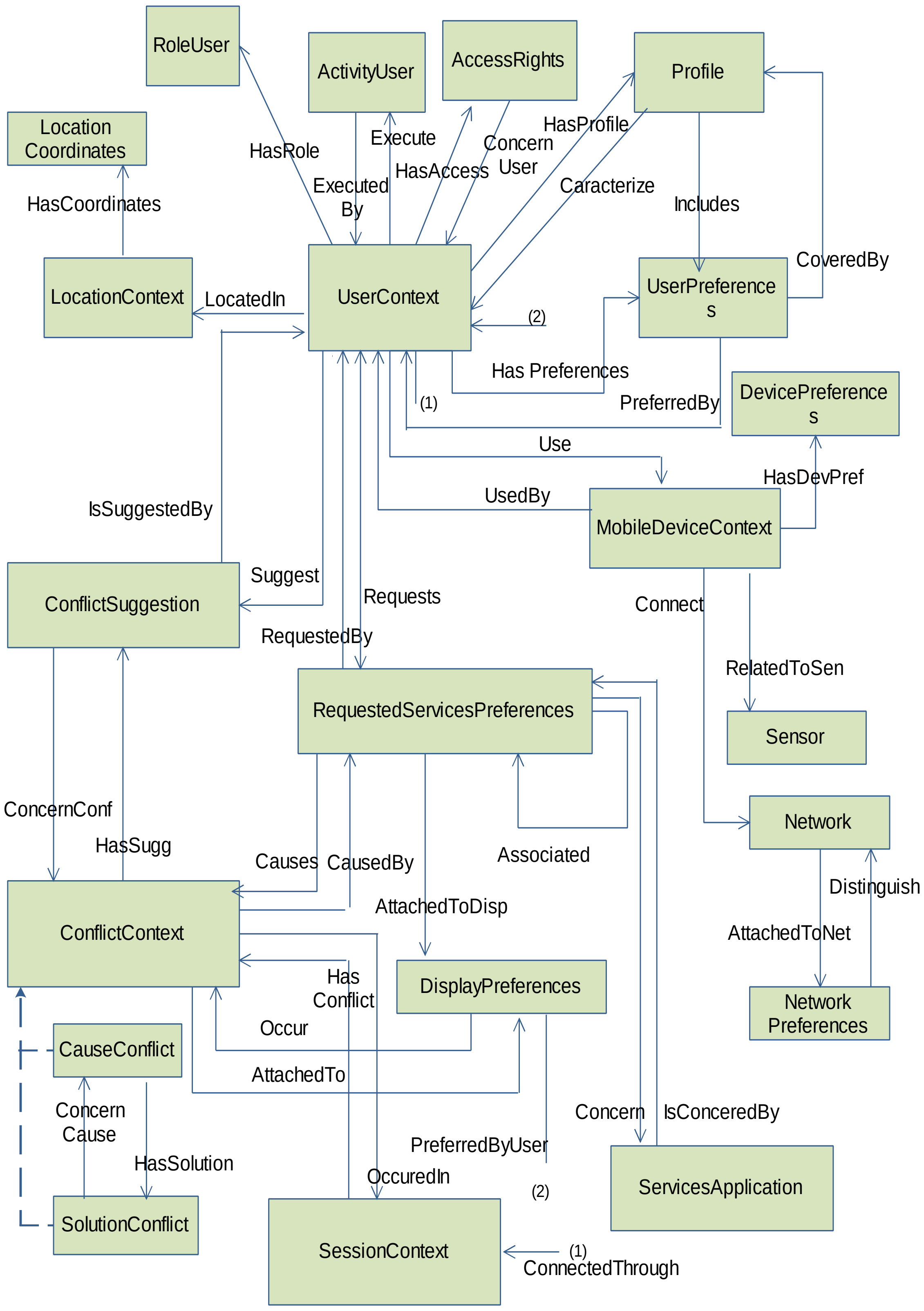


Figure 3: Binary Relations Diagram.

## f) **Attributes Table**:

The attribute table (see table 7 for some attributes) specifies for each attribute included in the dictionary of concepts, the set of constraints and restrictions on these values.

| Attribute name | Concept name | Value Type | Value range | Cardinality |
|---|---|---|---|---|
| IDContMod | Context Model | String | • - | • (1,1) |
| Description | Context Model | String | • - | • (1,1) |
| IDApp | ApplicationContext | String | • | • (1,1) |
| DescriptApp | ApplicationContext | String | • - | • (1,1) |
| ……………. | ……………… | ……… | ……… | …….. |

Table 7: Table of Attributes.

## g) Instances Table.

This table describes the known instances that are already identified in the dictionary of concepts. For each instance, specify the instance name, the concept where she belongs, these attributes and values that are associated with it. Table 8 illustrates some instances created.

| Instance Name | Concept name | Attributes | Values |
|---|---|---|---|
| ContextConflict_1 | Conflict Context | IDConf DescripConfl | Contradiction between the Requested_Service_Preferences and access rights of the user |
| CauseConflict_1 | CauseConflict | IDCause DescripCause | The user requests a Service which does not suit with these access rights. |
| SolutionConflict_1 | Solution Conflict | IDSolution DescripSolution | Suggestion |
| SolutionConflict_2 | Solution Conflict | IDSolution DescripSolution | Stop |
| SolutionConflict_3 | Solution Conflict | IDSolution DescripSolution | ContoLogy |
| ………… | ……………… | ……………….. | ……………………. |

Table 8: Instances Table.

## h) Logical Axioms Table

The table of axioms defines the concepts using logical expressions. Each axiom includes the name of the concept on which gate the axiom, a natural language definition and logical expression (see table9, for some logical axioms).

| Concept | Description | Expression logique |
|---|---|---|
| UserContext | A user has rights access, execute activities, request services, has a role, hasa profile, exist in a location, prefer display preferences ...... | ∀(X), UserContext(X) ⇒ ∃(Y), AccessRights (Y) ∧HasAccess (X, Y) ∧ ∃(Z), ActivityUser (Z) ∧ Execute (X, Z) ∧ ∃(W), RequestedServicePreferences (W) ∧ Requests (X, W) ∧ ∃(R),RoleUser (R) ∧ HasRole(X,R)∧ ∃(P),Profile (P) ∧ HasProfile(X,P) ∧ ∃(L),LocationContext (L) ∧ LocatedIn(X, L)∧ ∃ (D), DisplayPreferences (D) ∧ PreferredByUser (X, D)……. |
| …………. | …………… | ……………………… |

Table 9: Logical axioms.

## 5.4 Formalization

In this step, we use the formalism of description logic to formalize the conceptual model that we obtained in the previous stage of conceptualization. We Define the ContextModel as follows: ***ContextModel =(T ,A) with*** T= (Tbox) et A=(Abox)

## a) The TBox Construction:

We build the TBox concepts by defining concepts, roles and using constructors provided by description logics. For example, the definition « a 'ActivityUser' must be at least performed by a 'user' , can be written in description logic : ***ActivityUser≡ ∃ExecutedBy***

In addition, we build the TBox by specifying subsumption relations between the various concepts / roles; for example, specify that the class 'User Context ' is subsumed by the class' ContextModel we written: ***UserContext⊑ContextModel***

The definition of some concepts is illustrated in the table10 below.

| Concept | Definition | Subsumption relations |
|---|---|---|
| **ContextModel** | ≡ (UserContext ⊔MobileDeviseContext ⊔ LocationContext ⊔ ApplicationContext ⊔ ConflictContext ⊔ ConflictSuggestion ⊔ PreferencesContext ⊔ Profile ⊔ Intrefaces ⊔ Network ⊔ Sensor ⊔ Rules ⊔ ActivityUser ⊔ AcessRights ⊔ | ContextModel ⊑ ⊤ |

| | | |
|---|---|---|
| | Location Coordinates ⊔ RoleUser) | |
| **ConflictContext** | ≡ (CauseConflict ⊔ SolutionConflict) ⊓ ∃HasSugg.ConflictSuggestion ⊓ ∃CausedBy.RequestedServicesPreferences ⊓ ∃ AttachedTo.DisplayPreferences ⊓ ∃OccoredIn.SesionContext | ConflictContext ⊑ ContextModel |
| …………………… | ………………………… …………… | ……………… ………………… |

Table 10: Definition of TBox.

### b) The ABox Construction:

We describe the facts by using the assertional language, as follows: (1) **A(C)**: To specify that A is an instance of class C, for example: CauseConflict(CauseConflict_1). (2) **R (A1, A2)**: To specify that the two individuals A1 and A2 are connected by the relation R. For example: HasSolution (ConflictContext_1, Solutionconflict_1). In both Tables: Table11 and Table12, we define some assertions:

| Concept | Definition |
|---|---|
| **Conflict Context** | • ConflictContext(ConflictContext_1),ConflictContext(ConflictContext_2)<br>• …………………………… |
| **Cause Conflict** | • CauseConflict(CauseConflict_1)<br>• ……………………… |
| **Solution Conflict** | • SolutionConflict(SolutionConflict_1) …………………… |

Table 11: Concepts Assertional Part.

| Relation | Definition |
|---|---|
| **HasSolution** | HasSolution(ConflictContext_1, SolutionConflict_1) |
| ………………… …….. | ………………… |

Table 12: Relations Assertional Part.

## 6 The context rules description: conflicts manage rules

By using the ontology "ContoLogy", we can derive a new context. The context derived is an implicit context derived from explicit context. In our context ontology, derived based on rules in the form **antecedent** → **consequent**. Antecedent and consequent are composed of one or more concepts of context and the description of roles. Derived context can affect other contextual aspects. For example: ConflictContext is a context derived from MobileDeviceContext, UserContext and UserPreferences. In our work, we planned to resolve all conflicts which can arise when checking the user's preference. In the precedent section, we have defined five conflicts that may arise during the verification of user preferences. To manage these conflicts, we used the Semantic Web Rule Language (SWRL), we have defined five SWRL to derive conflicts, five SWRL to resolve these conflicts and we have created these rules under ProtégeOwl [41].

### 6.1 SWRL to Derive Conflicts

We define five rules to derive the five conflicts (see table N°9)

- ✓ ***Rule1: derive the Conflict1: "*** Contradiction between The Requested_Service_Preferences and access rights of the user " : *UserContext(?x) ⋀ RequestedServicesPreferences(?A) ⋀ AcessRightsUser(?AR) ⋀ differentFrom(?A, ?AR) ⋀ ConflictContext(?c) → Causes(?A, ?c)*
- ✓ ***Rule2: derive the Conflict2: "***Contradiction between the display preferences and the characteristics of used MD ": *UserContext(?x) ⋀ DisplayPreferences(?d) ⋀ MobileDeviceContext(?dm) ⋀ differentFrom(?dm, ?d) ⋀ ConflictContext(?c) → Occur(?d, ?c)*
- ✓ ***Rule3: derive the Conflict3: "***Various wishes of Display for the same service": *UserContext(?x) ⋀ RequestedServicesPreferences(?A) ⋀ MobileDeviceContext(?dm) ⋀ differentFrom(?dm, ?d) ⋀ sqwrl:isEmpty(?d) ⋀ ConflictContext(?c) → Causes(?A, ?c)*
- ✓ ***Rule4: derive the Conflict4: "***Absence of display preferences after checking the historic of the user": *UserContext(?x) ⋀ RequestedServicesPreferences(?A) ⋀ sqwrl:isEmpty(?d)⋀ Notprefered(?d, ?x) ⋀ ConflictContext(?c) → Causes(?A, ?c)*
- ✓ ***Rule5: derive the Conflict5: "*** Contradiction between the Display preferences requested and display capabilities expressed": *UserContext(?x) ⋀ RequestedServicesPreferences(?A) ⋀ MobileDeviceContext(?dm) ⋀ differentFrom(?dm, ?d) ⋀ sqwrl:isEmpty(?d) ⋀ ConflictContext(?c)* →Causes(?A, ?c)

## 6.2 SWRL to Resolve Conflicts

We define five SWRL for resolving the five Conflicts, see table N°9 for the description (values) of all parameters of the following rules.

- ✓ ***Rule6: resolve the Conflict1:*** *ConflictContext(ConflictContext_1) ⋀ CauseConflict(CauseConflict_1) → HasSolution(ConflictContext_1, SolutionConflict_1)⋀ HasSolution(ConflictContext_1, SolutionConflict_2).*

- ✓ ***Rule7: resolve the Conflict2:*** *ConflictContext(ConflictContext_2) ⋀ CauseConflict(CauseConflict_2) → HasSolution(ConflictContext_2, SolutionConflict_3)⋀ HasSolution(ConflictContext_2, SolutionConflict_1)⋀ HasSolution(ConflictContext_2, SolutionConflict_4)*

- ✓ ***Rule8: resolve the Conflict3:*** *ConflictContext(ConflictContext_3) ⋀ CauseConflict(CauseConflict_3) → HasSolution(ConflictContext_3, SolutionConflict_5) ⋀ HasSolution(ConflictContext_3, SolutionConflict_4)*

- ✓ ***Rule9: resolve the Conflict4:*** *ConflictContext(ConflictContext_4) ⋀CauseConflict(CauseConflict_4) → HasSolution(ConflictContext_4, SolutionConflict_1) ⋀ HasSolution(ConflictContext_4, SolutionConflict_4)*

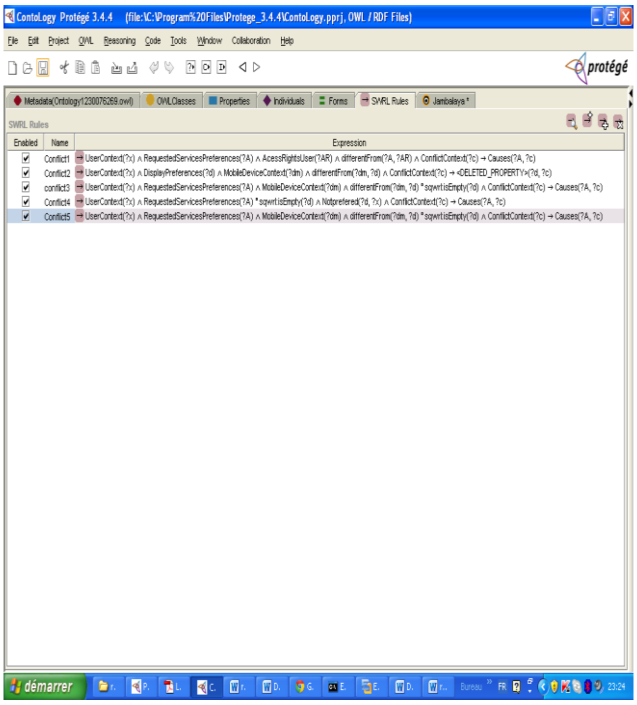
- ✓ ***Rule10: resolve the Conflict5:*** *ConflictContext(ConflictContext_5) ⋀ CauseConflict(CauseConflict_5) → HasSolution(ConflictContext_5, SolutionConflict_1)⋀ HasSolution(ConflictContext_5, SolutionConflict_3)⋀ HasSolution(ConflictContext_5, SolutionConflict_4)*

## 6.3 SWRL Rules Creation with Protégé:

We have used PROTÉGÉ 2000 to implement the precedent rules. Figure 4 show the creation of the SWRL rules under protégé

## 7 Ontology implementation

After the conception of the ontology "ContoLogy", we will implement our ontology. For this, we choose the editor Protégé OWL [41] and we used to formulate the ontology in the knowledge representation the language OWL. OWL represents a codification language used to implement the OWL ontology, and that, for all semantic functionalities than allows OWL which is richer than languages DAML + OIL & RDFS. In addition, we use to check the ontology the reasoner RACER (calculate the subsumption relation between concepts, and check the consistency of all concepts) [42].


ContoLogy Protégé 3.4.4 (file:\C:\Program%20Files\Protege_3.4.4\ContoLogy.pprj, OWL / RDF Files)

SWRL Rules

| Enabled | Name | Expression |
|---|---|---|
| ✓ | Conflict1 | UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ AcessRightsUser(?AR) ∧ differentFrom(?A, ?AR) ∧ ConflictContext(?c) → Causes(?A, ?c) |
| ✓ | Conflict2 | UserContext(?x) ∧ DisplayPreferences(?d) ∧ MobileDeviceContext(?dm) ∧ differentFrom(?dm, ?d) ∧ ConflictContext(?c) → <DELETED_PROPERTY>(?d, ?c) |
| ✓ | conflict3 | UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ MobileDeviceContext(?dm) ∧ differentFrom(?dm, ?d) ˚sqwrl:isEmpty(?d) ∧ ConflictContext(?c) → Causes(?A, ?c) |
| ✓ | Conflict4 | UserContext(?x) ∧ RequestedServicesPreferences(?A) ˚sqwrl:isEmpty(?d) ∧ Notprefered(?d, ?x) ∧ ConflictContext(?c) → Causes(?A, ?c) |
| ✓ | Conflict5 | UserContext(?x) ∧ RequestedServicesPreferences(?A) ∧ MobileDeviceContext(?dm) ∧ differentFrom(?dm, ?d) ˚sqwrl:isEmpty(?d) ∧ ConflictContext(?c) → Causes(?A, ?c) |

Figure 4: SWRL for Managing Conflicts.

PROTEGE OWL is a modular interface, developed at Stanford Medical Informatics, to edit, visualize, control (check constraints) ontologies [41]. PROTEGE OWL allows the definition of meta-classes which whose instances are classes, which allows you to create its own model of knowledge before building ontology. Many plugins are available or can be added by the user. The software architecture allow the insertion of plug-ins that can introduce new features (for example, the ability to import and export ontologies built in various operational representation languages such as OWL or specification of axioms) participated in the success of PROTEGE OWL, which includes a very large user community and is a reference for many other tools [43].

## 7.1 Implementation steps

First we start by creating concepts specified in the conceptualization step. After building classes, we create the properties for each of them see figure 5, and then we create restrictions on classes and properties see Figure6 and figure 7.

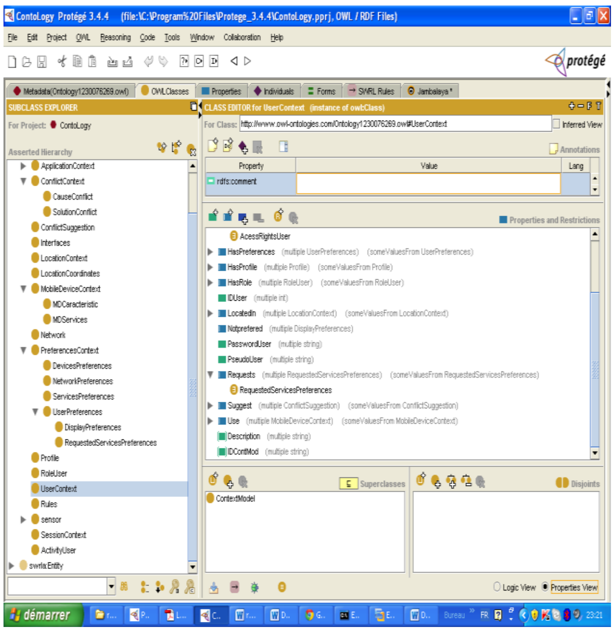


Figure 5: Contology classe and properties creation.

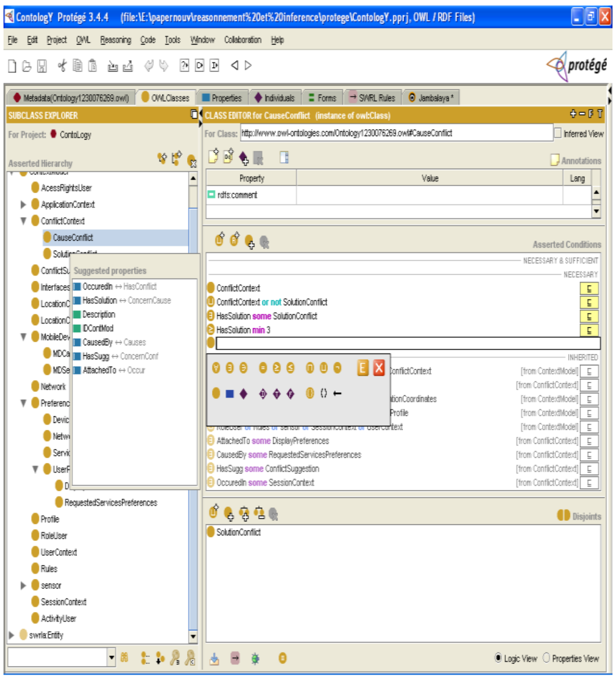


Figure 6: Contology restriction view1 with PROTÉGÉ.

After this step, we can transform the ontology to OWL form. An excerpt from the context model ontology in OWL is illustrated below:

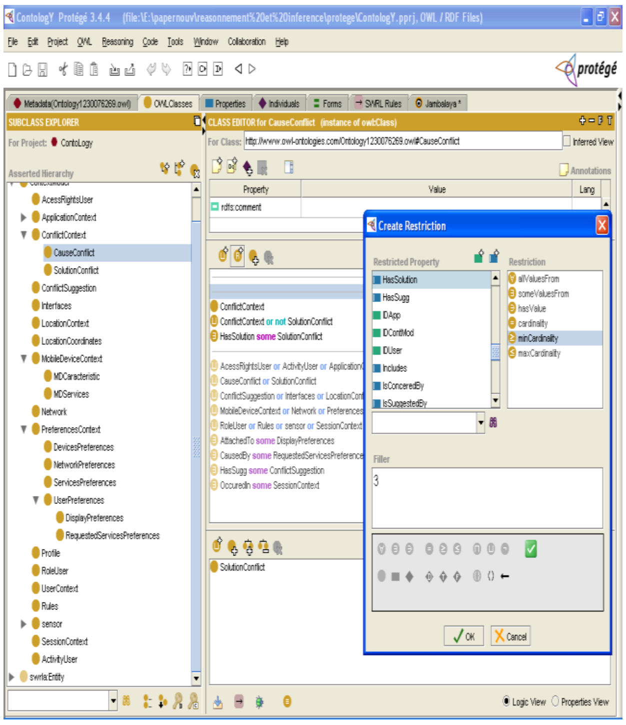
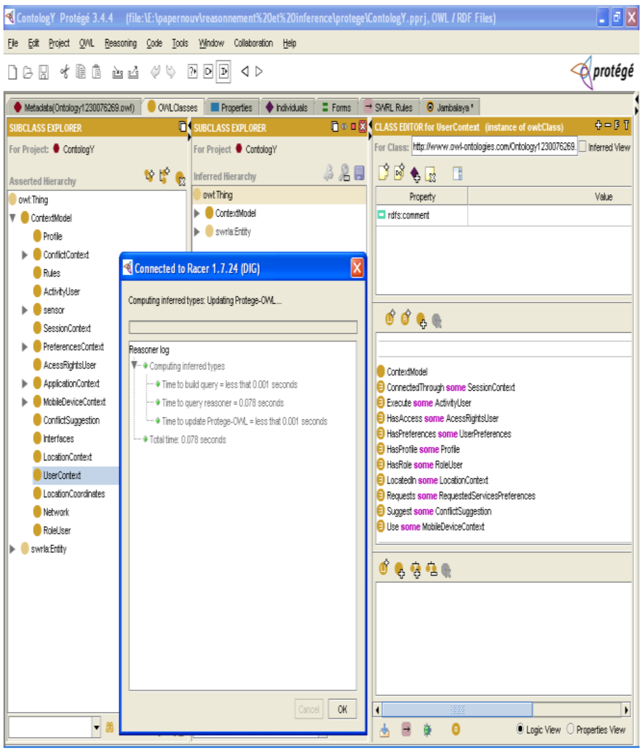


Figure 7: Contology restriction view2 with PROTÉGÉ.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/0
2/22-rdf-syntax-
ns#"xmlns:protege=http://protege.stanf
ord.edu/plugins/owl/protege#
        xmlns="http://www.owl-
ontologies.com/Ontology1230076269.
owl#"
    xmlns:swrl="http://www.w3.org/2
003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/
2003/11/swrlb#"
          ………..
</owl:Ontology>
<owl:Classrdf:ID="ServicesPreference
s">
<rdfs:subClassOf>
<owl:Classrdf:ID="PreferencesContext
"/>
</rdfs:subClassOf>
</owl:Class>
…….<owl:Classrdf:ID="Profile">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom>
<owl:Classrdf:ID="
UserPreferences"/>
</owl:someValuesF
rom>
<owl:onProperty>
<owl:ObjectPropert
yrdf:ID="Includes"/
>
</owl:onProperty>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
        ………
<owl:Restriction>
<owl:someValuesFr
om>
<owl:Classrdf:about
="#ConflictContext
"/>
</owl:someValuesF
rom>
<owl:onProperty>
    <owl:ObjectPr
opertyrdf:ID="Caus
es"/>….
```

## 7.2 The “Contology” Test

We used the system Racer to test the ontology "Contology", we distinguish three types of test: Inference, Consistency test and classification test; The first consists on remove the inconsistency between concepts, and this by using the subsumption test incorporated into the Racer system, against the second allows to check the existence of each concept instances; a concept C is satisfiable if and only if there is at least an interpretation I (instance) for the concept C. Racer is in the form of a server which can be accessed by TCP or HTTP. So we must first configure the connection to the server hosting the system Racer. We have carried all tests, and they are checked. Figure8 shows an example of inference test, figure 9 shows an example of consistency test and figure 10 shows an example of classification test.

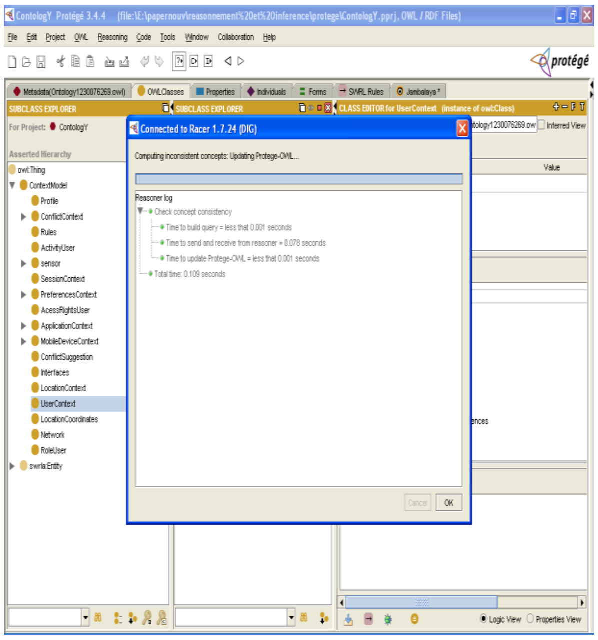

```
Connected to Racer 1.7.24 (DIG)

Computing inferred types: Updating Protege-OWL...

Reasoner log
Computing inferred types
    Time to build query = less that 0.001 seconds
    Time to query reasoner = 0.078 seconds
    Time to update Protege-OWL = less that 0.001 seconds
Total time: 0.078 seconds
```

```
ContextModel
ConnectedThrough some SessionContext
Execute some ActivityUser
HasAccess some AcessRightsUser
HasPreferences some UserPreferences
HasProfile some Profile
HasRole some RoleUser
LocatedIn some LocationContext
Requests some RequestedServicesPreferences
Suggest some ConflictSuggestion
Use some MobileDeviceContext
```

Figure 8: Test of inference.

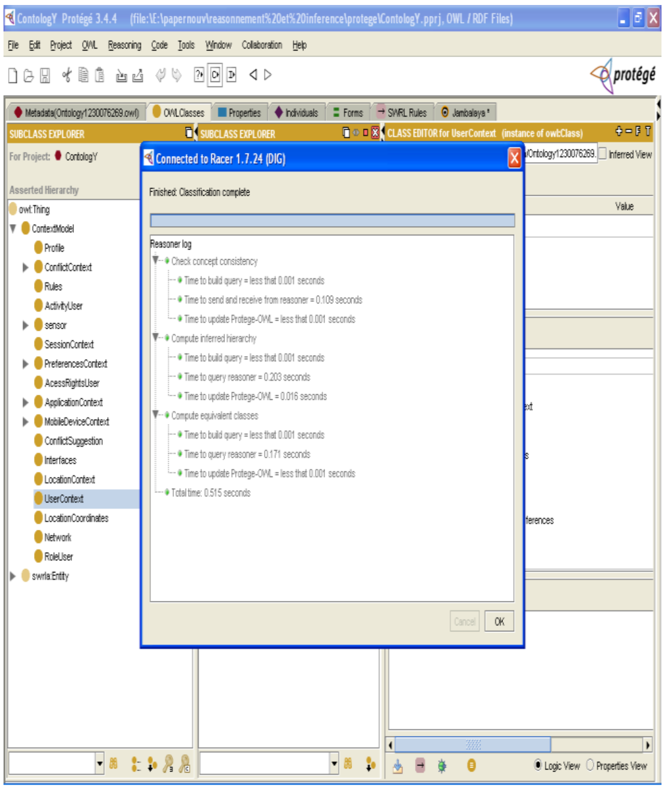

```
Connected to Racer 1.7.24 (DIG)

Computing inconsistent concepts: Updating Protege-OWL...

Reasoner log
Check concept consistency
    Time to build query = less that 0.001 seconds
    Time to send and receive from reasoner = 0.078 seconds
    Time to update Protege-OWL = less that 0.001 seconds
Total time: 0.109 seconds
```

Figure 9: Test of Consistency.

According to the tests we have applied to the ontology "ContoLgy", no error is produced during the test.


```
Connected to Racer 1.7.24 (DIG)

Finished: Classification complete

Reasoner log
Check concept consistency
    Time to build query = less that 0.001 seconds
    Time to send and receive from reasoner = 0.109 seconds
    Time to update Protege-OWL = less that 0.001 seconds
Compute inferred hierarchy
    Time to build query = less that 0.001 seconds
    Time to query reasoner = 0.203 seconds
    Time to update Protege-OWL = 0.016 seconds
Compute equivalent classes
    Time to build query = less that 0.001 seconds
    Time to query reasoner = 0.171 seconds
    Time to update Protege-OWL = less that 0.001 seconds
Total time: 0.515 seconds
```

Figure 10: Test of classification.

## 8 The context ontology exploitation: adaptation process

We exploit and use the ontology “ContoLogy” to adapt the user's initial request to the current context. Thus, we propose a web service based architecture to ensure the adaptation process. By use of the ontology "ContoLogy", adaptation can reasoning about the user's context and adapts the user's initial request to the current context. Among the different context parameters, we focus on: the location and the used Mobile Device (MD).

After having implemented the application, it is mandatory, for many reasons, to undergo it to the adaptation process. These reasons can be classified into four categories [44]: (1) Correctional Adaptation, (2) Adaptive Adaptation (3) Scalable Adaptation and (4) Perfective Adaptation. In our approach, we are interested to the adaptive adaptation in order to adapt ubiquitous applications to their execution environment. We adopt this kind of adaptation because the application is running properly, but its execution environment, hardware components or other applications or depending data are changing (e.g. the context of user). In this case, the application is adapted in response to changes in its execution environment. Consequently, to ensure this adaptation process, we use the context ontology “contology” to the adaptation composed of two main parts: static part and the dynamic part.

(1) **Static part:** This part is described by the ontology "Contology". It focuses, on one hand, on modeling the contextual information of users and their preferences and, on the other hand, on managing the potential conflicts which may arise between the users' preferences during their checking process.

(2) **Dynamic part:** the role of this part is to ensure the functional dynamic adaptation of context-sensitive applications to various user's contextual situations. The adaptation process adopted by this part is based on " Co*ntoLogy"* in order to offer a better respond to user. Also, this process is assured by the user's initial request adaptation to the context of use and user's preference using the ontology "ContoLogy". The methodology in our approach consists in three main steps: (1) the context of use modeling and the user's preferences managing, basing on a new context definition which separates the application data from the contextual data by by using "ContoLogy",, (2) the resolution of potential conflicts which may be occurred during managing of user's preferences and (3) the dynamic functional adaptive adaptation of web service-based context-aware applications. The accomplishment of the two last steps (2 and 3) is based on "ContoLogy".

In ubiquitous computing, applications are sensitive to the context (context-aware applications), user's access to various information's using different mobiles devices and in different localization, which implies, an overly dynamic, heterogeneous environment. To respond better to this challenge, we propose to use web service, for those benefits, such as:

1. The ultimate goal of the Web service approach is to transform the Web into a distributed computing system where programs (services) can interact intelligently by being able to automatically discover and negotiate with each other and consist into more complex services [45].
2. The establishment of web services facilitates the dialogue between heterogeneous environments. As web services can be implemented on different platforms and with different languages, they facilitate interoperability between heterogeneous systems and platforms, which is our case. [46]
3. Web Services [47] work with standard Web protocols (HTTP and TCP / IP) and XML. Many companies already have a Web infrastructure the staff have the knowledge and experience of maintenance. This is why the cost of access to Web services is much lower than that of previous technologies.[6]

The figure 11 shows the general architecture of the proposed approach.

As illustrated by the figure 11, the adaptation process to the context of use and the user's profile is accomplished in 16 steps explained bellow:

**(1)** ***Request***: the user sends his request to the platform via his Mobile Device (MD). The Module Context integration (CI) receives this request.

**(2)** ***Contextual information***: the module Context sensor sends contextual information of the user to the module Context integration, such us: the used MD, the localization.

**(3)** ***Contextual request***: in this step, the Module Context Integration increases the user request by the contextual information; the result of this step is a contextual request. The module (CI) sends this contextual request to the Preferences Management Web Service (PMWS).

**(4)** ***Preferences check***: In this step, the PMWS checks the contextual request using "ContoLogy". It checks the conformity between the user preferences and his access rights and the type of the used MD.

**(5)** ***Prefrences OK/ Conflict***: by consulting the ontology, the PMWS can detect that preferences are checked or can detect a conflict

**(6)** ***Soap Message: Conflit; Soap Message: Conflit:*** when a conflict arises, the PMWS sends a soap message containing the conflict to the Conflict Management Web Service (CMWP).

**(7)** ***Search Conflict Solution:*** using the Context Ontology, the CMWS Searches a solution for the detected conflict.

***(8)*** ***Solution conflict/ no solution:*** this step indicates whether or not there is a solution for the Conflict.

***(9)*** ***Ask suggestion:*** if no solution to the conflict, the CMWS asks a suggestion of solution for the conflict from the user.

***(10)*** ***Soap message: conflict solution:*** in this step, if the user sends a suggestion of solution for the conflict to the CMWS, it takes this solution and sends it to the PWSM.

***(11)*** ***Update conflict information:*** the CMWS updates the conflict information by adding the conflict information of the current session.

**(12)** ***Soap message: request updated:*** The PMWS sends the request of the user, after the verification, to the adapter web service (AWS).

**(13)** ***Search answer:*** the AWS search an answer for the request of the user.

**(14)** ***Soap message: answer:*** Once the answer is found, the AWS sends it to the PMWS.

**(15)** ***Answer adapted to the context:*** this later sends this answer adapted to the context to the user

**(16)** ***Update contextual information:*** finally, the PMWS updates the contextual information by adding the contextual information of the current session to the Context Ontology.

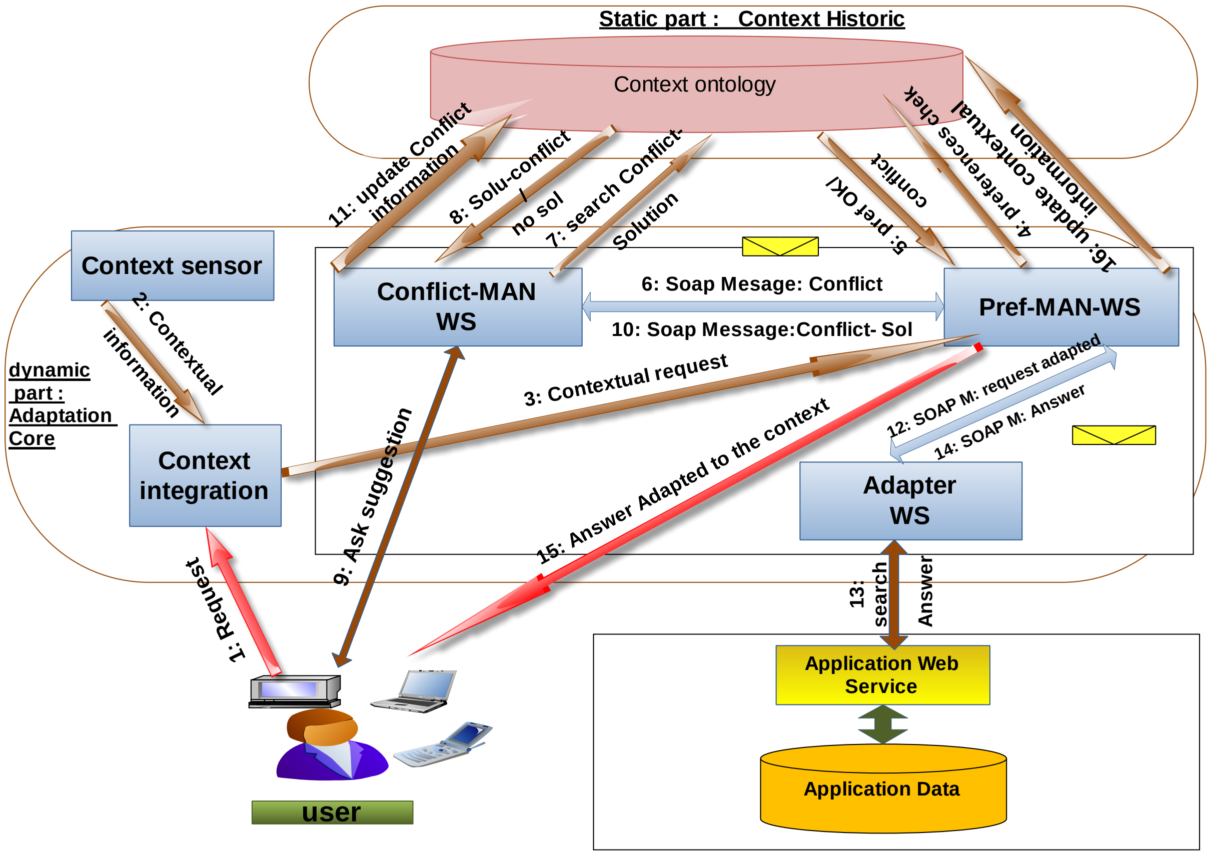

Static part : Context Historic
Context ontology
11: update Conflict information
8: Solu-conflict / no sol
7: search Conflict-Solution
5: Pref OK/ conflict
4: preferences chek
16: update contextual information
Context sensor
2: Contextual information
dynamic part : Adaptation Core
Context integration
Conflict-MAN WS
6: Soap Mesage: Conflict
10: Soap Message:Conflict- Sol
Pref-MAN-WS
3: Contextual request
12: SOAP M: request adapted
14: SOAP M: Answer
Adapter WS
15: Answer Adapted to the context
9: Ask suggestion
1: Request
13: search
Answer
Application Web Service
Application Data
user

Figure 11: Architecture of our approach.

## 8.1 Process adaptation presentation

In this section, we present the dynamic part of our approach to adapt the ubiquitous applications to the user's context and the user's profile, using "ContoLogy".

This part assures the functional dynamic adaptive adaptation of these applications sensitive to the context of use and the user's profile, it is assured by the adaptation of the initial request of user to the context of the current session in the various contextual situations. At the end, the user can meet the best answers to their expectations.

The context of use of a user witch accedes to a ubiquitous application, in addition to be composed of multiples aspects is very variable and in constant evolution, which makes the adaptation process of the application hard to accomplish. In order to ensure this adaptation process and to be able to change the behavior of such application sensitive to the context of use, we propose to use Web Services (WS) both during the development of this type of application and in the dynamic part of the adaptation.

We opted for web service for the advantages it procures. The dynamic part of our approach is composed of three Web Services: Preferences Manager Web Service (PMWS), Conflicts manager Web Service (CMWS) and Adapter Web Service (AWS) and two modules: Context integration and context sensor. This part assures the adapting of the user request to the context, resolving the conflicts and returning an answer adapted to the user's context.

### 8.1.1 Preferences Manager Web Service

This web service is charged of the preferences management. Consequently, it ensures checking of the user's preferences using the initial request of the user and "ContoLogy", the PMWS can reason on the user context. The PMWS can analyze the context of the user that appears in the contextual request of the user. Consequently, it verifies the conformity between the requested preferences and the context of use, mainly the used MD, localization and his accesses rights. This step can generate conflicts which can be detected by PMWS.

Also, it reformulates the initial request of user, in the case of conflicts, by adding the new preferences. It sends to the user the adapted answer to the context, and stored the new context for using it in the next sessions, when we receive the same context and request (see Figure 12 ).

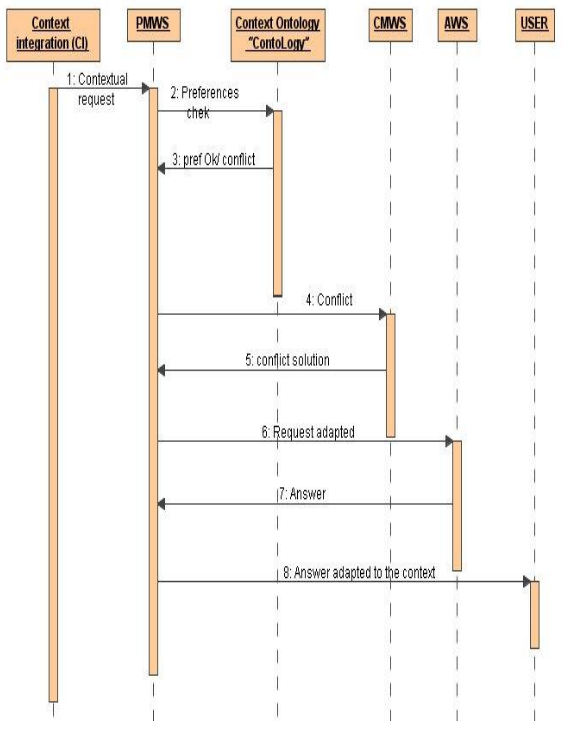


Figure 12: PMWS Sequence Diagram.

*Technical Example: see section 9.1*

### 8.1.2 Conflicts Manager Web Service

The role of this web service is to manage the conflicts that may arise between user preferences. The conflicts are managed, by our approach, according to the following sequence diagram (Figure 13). Specifically, our approach manages five conflicts (1) Contradiction between The Requested_Service_Preferences and access rights of the user. (2) Contradiction between the display preferences and the characteristics of used MD. (3) Various wishes of Display for the same service. (4) Absence of display preferences after checking the historic of the user. (5) Contradiction between the Display preferences requested and display capabilities expressed.) (see table 2). This web service executes the proposed solution for each conflict can be arose between the preferences of user (Table 3). After receiving a message containing the conflict which has occurred, *Conflicts Manager Web Service* reasons and infers a solution to conflict occurred by using "ContoLogy", if not; it implies the user to give his suggestions for this conflict. If there are no suggestions it takes a default solution, for each conflict (i.e. our approach proposes a determinate solution (see table 3). At the end, it sends a message which contains the solution of the conflict to the PMWS. Consequently, it updates the history of conflict information. This web service ensures: the resolution of conflicts using "Contology", and the storage of information of the occurred conflict.

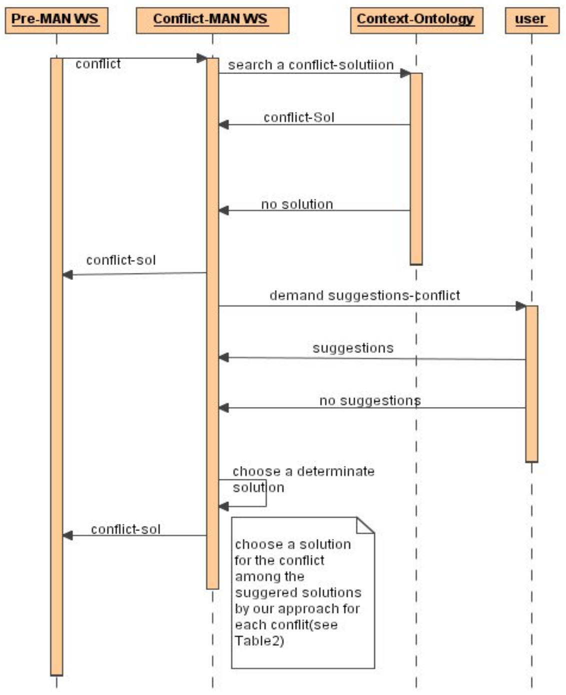

Pre-MAN WS
Conflict-MAN WS
Context-Ontology
user
conflict
search a conflict-solutiion
conflict-Sol
no solution
conflict-sol
demand suggestions-conflict
suggestions
no suggestions
choose a determinate solution
conflict-sol
choose a solution for the conflict among the suggered solutions by our approach for each conflit(see Table2)

Figure 13 .Conflicts Sequence Diagram.

*Technical Example: see section 9.1*

### 8.1.3 Adapter Web Service

Its role is to return an adapted request to the user. It executes the following steps: firstly, it accedes to the Web Services of the application and researching on the WSDL of these latter, in order to extract Web Services with their interfaces, their operations and the number of interfaces specific to each Web Service. Secondly, selecting the Web Service which answers better the request of the user. Then, it reformulates and sends to PMWS the adapted answer to the context of use.

### 8.1.4 Context Sensor

This module is responsible of the capture of the user context at a connection time, namely: localization, MD, session. Then, it sends this contextual information to the module "Context integration". It is composed of the two following Sub-modules:

1- *Logical context sensor*: a set of interfaces used by the user to enter his context.
2- *The physical context sensor*: a set of physical dispositive used to capture the context of the use.

### 8.1.5 Context Integration

This module receives the initial request of the user and reformulates it by adding the contextual information. Then, it sends this contextual request to PMWS.

## 8.2 Utilization of "CONTOLOGY":

In this section, we explain how the user communicates with our platform to get an adapted response to its context (figure14), the adopted communication process is accomplished in four main steps:

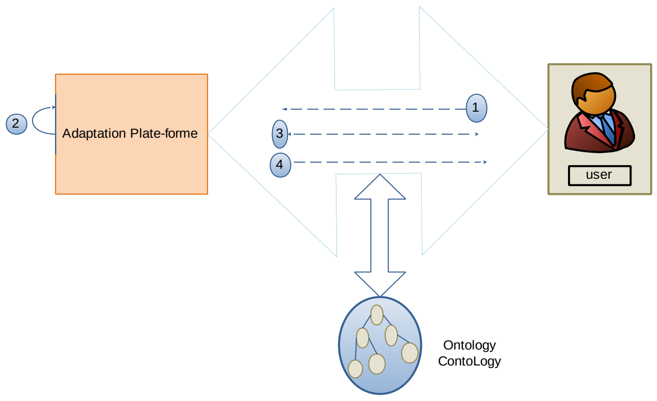


Figure 14: Communication between the user and platform using "ContoLogy".

(1) **Sending Request**: The user sends a request to the platform asking the available services and providing the necessary information (context, location, MD ............ ..).

(2) **initial request augmentation** : The platform, using context sensor module and the module context integration, increases the initial user request by adding contextual information, this contextual request will be sent to PMWS for checking preferences using " ContoLogy".

(3) **Conflicts resolving**: in the case of conflicts when checking the preferences, the platform using the CMWS and the ontology search a solution for the conflict, or demands a suggestion from the user.

(4) **Adapting Response**: after checking preferences and taking into account the context of the user, this latter receives a response adapted to his context.

In the flow, we presents two scenarios using " ContoLogy", in order to show how our p roposed approach uses "contology" to reason and infer new information for taking into account the context.

*a- Scenario1: Preferences Checking:*

The preferences checking process (figure 15) is accomplished in three main steps:

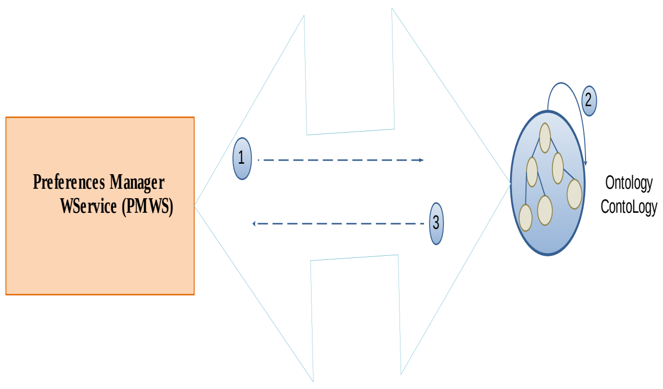


Figure 15: Scenario1: Checking of the preferences.

**(1) Contextual request**: PMWS uses "ContoLogy" to verify the contextual request of the user that contains the user's context namely the type of the used MD and the location. The PMWS checks the conformity of services requested by the user with their access rights, and display preferences with display capabilities offered by the used MD, and that using the information of the previous sessions stored in the ontology.

**(2) Reasoning and inference:** according to the contextual information that exists at "ContoLogy" we can check the user preferences, reasoning on the current context with the available information and also infer new user preferences in the case of conflict.

**(3) Chek result**: in this case, "Contology" can refer two answers. The first answer is: preferences OK, where preferences are checked. The second one is, a conflict has been arisen between user preferences, which must be resolved by the CMWS (see next scenario).

b- *Scenario2*: Conflicts Resolution:

Figure 16 illustrates the conflicts resolving process we propose. It is accomplished in seven steps:

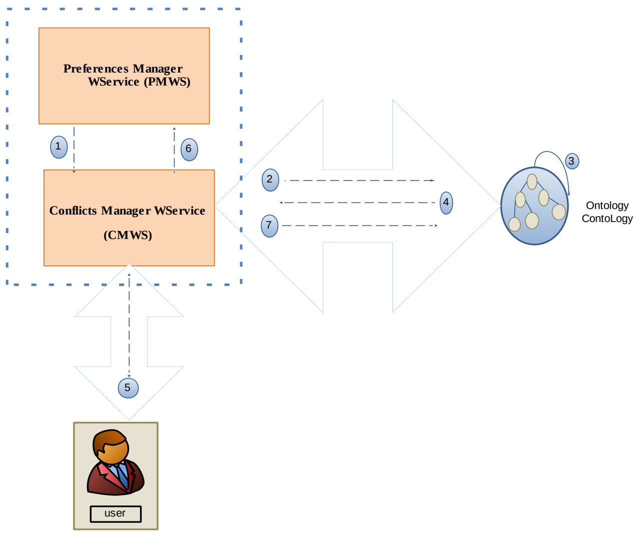


Figure 16: Scenario2: Conflicts resolution.

(1) **Conflict**: the previous scenario can cause a conflict, so it will be sent to the CMWS by the PMWS.

(2) **Searching for a solution to the conflict**: using the Context Ontology, the CMWS Searches a solution for the detected conflict.

(3) **Reasoning and inference about the conflict**: using "ContoLogy", the CMWS can reason about the conflict information of previous sessions and infer a solution to the current conflict.

(4) **Solution / no solution:** this step indicates whether or not there is a solution for the Conflict.

(5) **Conflicts suggestions**: if CMWS does not find a solution to the conflict in the ontology, it asks a suggestion of solution from the user. This latter can give a solution, change the request or does not responds.

(6) **Soap message: conflict solution:** in this step, if the user sends a suggestion of solution of the conflict to the CMWS, he takes this solution and sends it to the PWSM.
(7) **Update conflict information:** the CMWS updates the conflict information by adding the conflict information of the current session.

## 9 The case study: the travel booking application

In this section, we present using a case study, how we exploit the Ontology "ContoLogy" for the adaptation of the user request. For this, we have created a travel booking application to be used in the process adaptation, and we have implemented the dynamic part of our approach. We will present the different steps we followed during the implementation. Firstly, we present the environment and the tools that we used in the implementation. Secondly, we will present the application we have developed; finally, we detail the implementation steps, by a detailed example, from the reception of the request of the user passing through the resolution of conflicts, until reception of the adapted response by the user.

The environment and tools we used to implement the system Such: Microsoft visual studio( Visual Web Developer, Smart Device Applications, Web Forms, Windows Forms, XML Web Services, XML Support, C#) [48] , Protégé [49], OWL [50].

Travel booking is a web service-based application to manage a travel agency and Online reservation (see figure17).

It offers to user to make flight reservation and hotels reservation. This application is adapted by our architecture to the context and the profile of the user. Using this application the user can search for a flight, a hotel and car, and he can receive an answer adapted to his context, for example: adapted to: his location, the used MD, his city and the location of the airport. For example: the user can receive a list of hotel situated near the airport. For designing the agency services, we distinguished three web services:

(1) *Airline Service*: It offers services responsible for online managing of the flights reservations of customers.

(2) *Hotel Service:* It offers services which have like function, the online control of the hotels and reservations of the customers.

(3) *Location Car service:* It classifies all services responsible for online managing of cars and location.

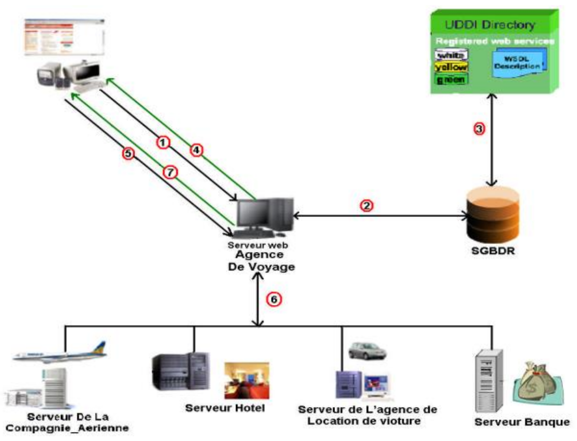


Figure 17: Global architecture of the application.

We have created a service portal that serves as a gateway to various web services. This portal does not store any data on its physical basis, but acts as a service provider. The application we have developed allows to a customer to avoid making several research on the web (airlines, hotel, car ...), to plan his travel. The portal we have implemented provides the interfaces necessary to planning travel through the use of web service technology. This application will be used by our system for the adapting to the context of use and the profile of user. The dynamic part of our approach ensures the process of adaptation, which will be the subject of the following section.

All web services related to the dynamic part which are necessary to validate our approach are created using Microsoft visual studio. More precisely, three web services have been created to handle the interaction and the messages between the user and the application. After the web service creation, a C# page will pop on which named service1.asmx.cs. The page contains the library that we need and the web service code behind. To create a web service method in *.net* environment, simply we write the [WebMthode] and after that we write the method .

### 9.1 Process Adaptation Unfolding

In this section, we detailed our approach to manage preferences and conflicts, and detail the process adaptation unfolding, by using an example which explain the interactions between web services of our architecture, the ubiquitous application (Travel booking application); the context ontology "ContoLogy" and the user. For this, we present an example which includes basically the following points: (1) Interaction between user and the dynamic part and the context ontology "ContoLogy". (2) The receipt and the check of the user request. (3) Resolution of conflicts. For this, we take a conflict that can occur and we explain how the system will handle this conflict and we will see how the system resolves this conflict step by step. (4) Adaptation of the answer of the application to the context information. In this case we will take as example:

- *The ConflictContext(ConflictContext_2)* =“Contradiction between the display preferences and the MD characteristics”
- causes by *CauseConflict(CauseConflict_2)=* “The user requests a display which is not supported by his used MD”
- With:
  - The solution *SolutionConflict_3*= “ContoLogy” witch means: reasons and infers a better solution from “ContoLogy”.
  - If no, then *SolutionConflict_1=”suggestion”*witch means, demands a suggestion from user.
  - If no, then *SolutionConflict_4=” default_display_preference”*

## a. Interaction between user, our dynamic part and context ontology:

At the first time when the user login to the system, the system asks him to be registered on it, by giving his personal information such as name, username and address, email and choose his services and preferences that he prefer. The system will get automatically the MD (Mobile Device) characteristics from the MD information files. The MD characteristics in the ontology will be look like:

```
default:MD_i0435     MD:MDid "MD_i0435"
                      MD:Class "MD" ;
                      MD:Type "Nokia";
                     MD:ImageD "0" ;
                     MD:TextD "1" .
```

- *User:*

```
Default: i0435     profil:id "profile_i0435" ;
              profile:Class "USERPROFILE"
              profil:FName "MM1" ;
              profil:LName "TT1" ;
              profil:UserName "us11" ;
              profil:Password "pass1" ;
              profil:address "adress AD" ;
              profil:email "AD@hotmail.com".
```

- *Service Preference “Show flight”:*

```
default:preser_i043501
               preser:Num_Ser "preser_i043501" ;
               preser:Class "ServicePreferences"
      preser:ser "Show flights" ;
              preser:serAso1 "preser_i043502" ;
              preser:serAso2 "0" ;
           preser:dispser disser_i043501_pre01" .
```

As we see here, this service has an associated service "preser_i043502"which is “Show hotel” service

- *Service Preference “Show hotel”:*

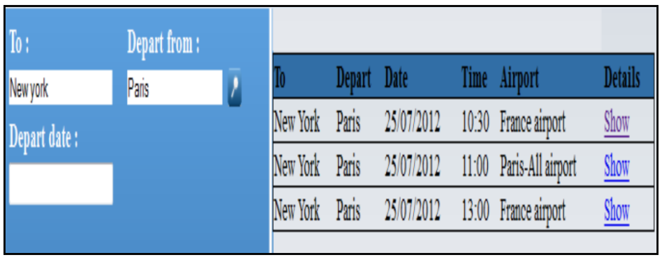
```
default:preser_i043502
     preser:Num_Ser "preser_i043502" ;
     preser:Class "RequestedServicePreferences "
     preser:ser "Show Hotels" ;
     preser:serAso1 "0" ;
     preser:serAso2 "0" ;
preser:dispser "disser_i043501_pre01".
```

- *Display Preference for: “Show flight” and “Show hotel”:*

```
default:disser_i043301_pre01
disser:Num_Dis "disser_i0433_pre01" ;
        preser:Class "DisplayPreferences "
        disser:default "disText" ;
        disser:disText "1" ;
        disser:disImage "1".
```

## b. Check of The User Request

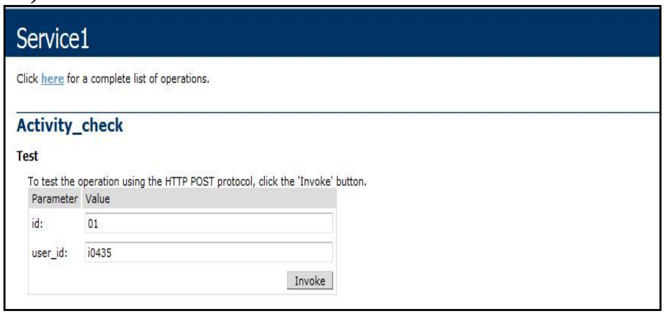
After user login, the next figure presents flight searching form will be displayed.



| To | Depart | Date | Time | Airport | Details |
|---|---|---|---|---|---|
| New York | Paris | 25/07/2012 | 10:30 | France airport | Show |
| New York | Paris | 25/07/2012 | 11:00 | Paris-All airport | Show |
| New York | Paris | 25/07/2012 | 13:00 | France airport | Show |

Figure 18: Flight searching result form.

After clicking on show details link, the PMWS receives the query and the contextual information for the user, and checks it with the user preferences and services on the ontology “ContoLogy” by the following steps:(1) PMWS receives the service ID and the contextual information ( localization and used MD) by the method “Service_check“. This method returns the associated_services and the display preference (figure 19).


Service1

Click here for a complete list of operations.

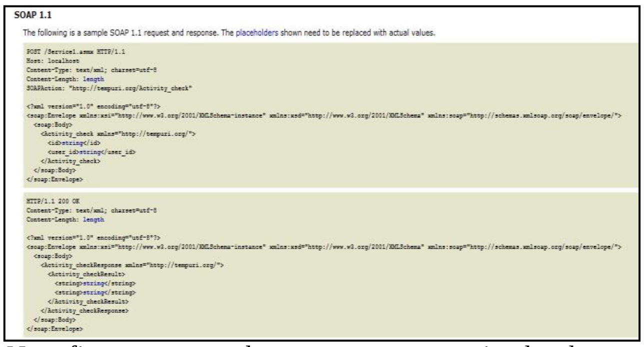
Activity_check

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

| Parameter | Value |
|---|---|
| id: | 01 |
| user_id: | i0435 |

Figure 19: Service_check method call.


SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Activity_check"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Activity_check xmlns="http://tempuri.org/">
      <id>string</id>
      <user_id>string</user_id>
    </Activity_check>
  </soap:Body>
</soap:Envelope>
```

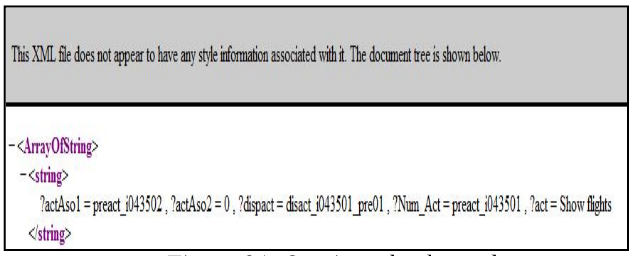
```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Activity_checkResponse xmlns="http://tempuri.org/">
      <Activity_checkResult>
        <string>string</string>
        <string>string</string>
      </Activity_checkResult>
    </Activity_checkResponse>
  </soap:Body>
</soap:Envelope>
```

2- Next figure presents the soap message receive by the PMWS

Figure 20: Service_check method SOAP 1.1.

3- In figure21, we find the result receive by the PMWS after checking the request of the user using "ContoLogy"



This XML file does not appear to have any style information associated with it. The document tree is shown below.

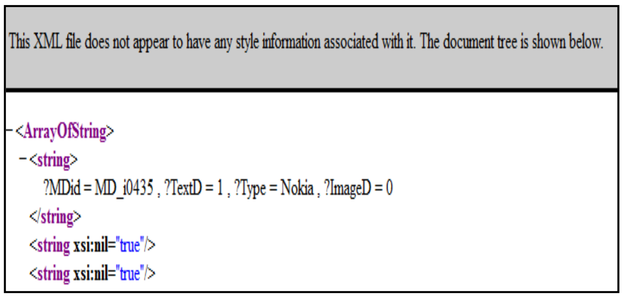
```
-<ArrayOfString>
  -<string>
     ?actAso1 = preact_i043502 , ?actAso2 = 0 , ?dispact = disact_i043501_pre01 , ?Num_Act = preact_i043501 , ?act = Show flights
   </string>
```

Figure 21: Service_check result.

4- In the next step, the PMWS compare the values that return from "MD_check" method, and the "display_check" method. In our example, the values will be not the same because:

- User MD does not support image display which its valu

(figure22).

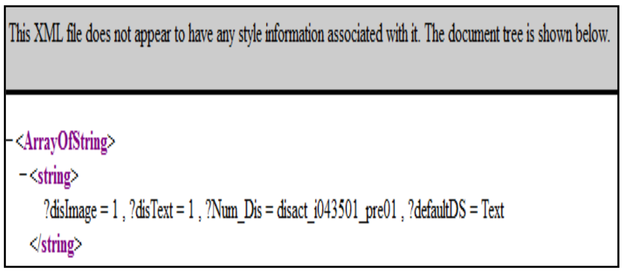


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
-<ArrayOfString>
  -<string>
     ?MDid = MD_i0435 , ?TextD = 1 , ?Type = Nokia , ?ImageD = 0
   </string>
   <string xsi:nil="true"/>
   <string xsi:nil="true"/>
```

Figure 22: MD_check result.

-Text and image forms in display preference have both the value 1 (figure 23).



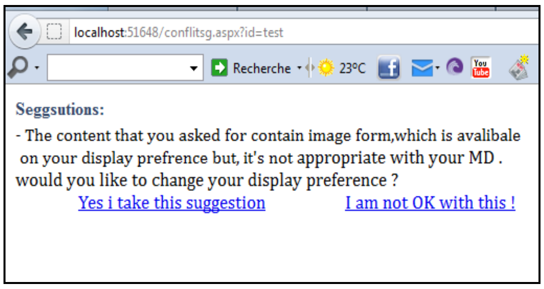
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
-<ArrayOfString>
  -<string>
     ?disImage = 1 , ?disText = 1 , ?Num_Dis = disact_i043501_pre01 , ?defaultDS = Text
   </string>
```

Figure23:Display_check result

## c. Conflict of md characteristics and display preferences

In this step the PMWS will detect the conflict between the display preference and the MD characteristics see figure23 and figure24. PMWS send the conflict to the CMWS, which it will consult the conflict and the solution will take to resolve it from the ontology "ContoLogy". The system will check the user history by History_check" method for similar service, and the preferences of that service. If there is not result from the user history, the system will demand the suggestion to the user. The suggestion will aim to change the display preference to this service to be appropriate with user MD (figure 24).



localhost:51648/conflitsg.aspx?id=test

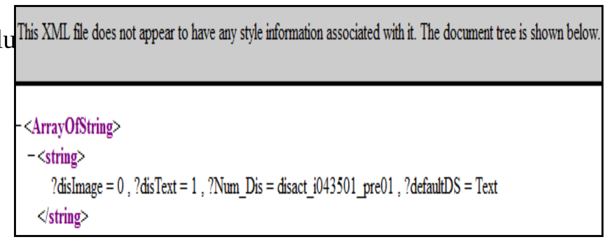
**Seggsutions:**

- The content that you asked for contain image form,which is avalibale on your display prefrence but, it's not appropriate with your MD . would you like to change your display preference ?

Yes i take this suggestion    I am not OK with this !

Figure 24: Conflict suggestion.

If the user chooses to take the suggestion, the CMWS sends to the PMWS the suggestion with method "change_cont_info" to update the display preference and change the display image to 0 values (figure 25).



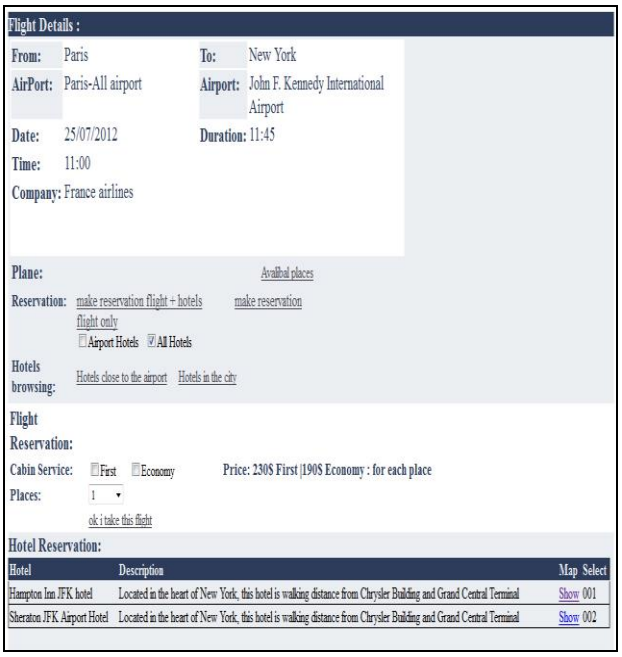
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
-<ArrayOfString>
  -<string>
     ?disImage = 0 , ?disText = 1 , ?Num_Dis = disact_i043501_pre01 , ?defaultDS = Text
   </string>
```

Figure 25: Check display result after the update.

## d. The User Request Adaptation

After updating display preference, the PMWS reformulates the user request by adding the contextual information and sends it to the AWS and gets the result from the travel-booking application (see figure26)



**Flight Details :**

| | | | |
|---|---|---|---|
| **From:** | Paris | **To:** | New York |
| **AirPort:** | Paris-All airport | **Airport:** | John F. Kennedy International Airport |
| **Date:** | 25/07/2012 | **Duration:** | 11:45 |
| **Time:** | 11:00 | | |
| **Company:** | France airlines | | |

**Plane:** Avalibal places

**Reservation:** make reservation flight + hotels    make reservation
flight only
☐ Airport Hotels ☑ All Hotels

**Hotels browsing:** Hotels close to the airport    Hotels in the city

**Flight Reservation:**

**Cabin Service:** ☐ First ☐ Economy    **Price: 230$ First |190$ Economy : for each place**

**Places:** 1

ok i take this flight

**Hotel Reservation:**

| Hotel | Description | Map | Select |
|---|---|---|---|
| Hampton Inn JFK hotel | Located in the heart of New York, this hotel is walking distance from Chrysler Building and Grand Central Terminal | Show | 001 |
| Sheraton JFK Airport Hotel | Located in the heart of New York, this hotel is walking distance from Chrysler Building and Grand Central Terminal | Show | 002 |

Figure 26: Result after the adaptation.

Figure 26 shows the result of the user request that it adapted to the user context and preferences. Our adaptation process is assured by the adaptation of the request of the user to their preferences.

According to all steps of this section, we can see the use of the ontology of the context "ContoLogy" for managing preferences and resolving Conflicts, in order to adapt the initial request of the user to his context of use and his profile, which includes his preferences.

## 10 Conclusion

The ubiquitous computing focuses on the use of two essential notions: user profile and context of use in order to satisfy better demands of nomadic users. Furthermore, a reliable modeling of such two notions and an adaptation of the application behavior to them are two required processes. In this paper, firstly, we presented a novel approach allowing, on one hand, modeling the context of use and the user profiles using an ontology, to support context representation and reasoning, and, on the other hand, resolving the conflicts using some proposed solutions. An architecture illustrating the dynamic adaptation of web service-based ubiquitous applications is also proposed. Secondly; we detailed a prototype implementation and system performance. Through this part in this paper, we tried to explain how we implement the web services, the ontology and shown up the adaptation process to resolve the conflicts by a detailed example.

As future directions to this work, we plan to:

1. Complete the implementation of the context acquisition module composed of two sub-modules: context sensor and context integration.

2. Use a probabilistic approach to represent the users' preferences. Because, it is a very complex challenge to represent the users' preferences with its contexts and the ambiguity posed by these ubiquitous applications. One of the considerations which generate abstraction data sources of information are cited for example: temporality, uncertainty, heterogeneity, online processing, and conflicting information. In the literature, several probabilistic (SVM, CPnet, HMM, HHMM, etc) are studied and we decide on the Hierarchical Hidden Markov Model (HHMM). HHMM is legible, easy for the preferences representation and does not require expertise in prior.
3. The cloud computing provides the next generation of Internet based, highly scalable ubiquitous computing systems in which computing resources are provided as a service. A new computing model that allows convenient access and on-demand network to a shared pool of configurable computing resources (eg, networks, servers, storage, applications and services) that can be rapidly provisioned. However, ubiquitous computing refers to a scenario in which computing is ubiquitous, particularly where devices that do not look like computers have computational capabilities. The idea is how to use cloud computing resources efficiently and earn maximum profits with ubiquitous systems?

## Acknowledgement

This work summarizes doctoral thesis research, supported by the University of Annaba and the University of Tebessa- Algeria.

## References

[1] Kosala.Y, MingXue.W, & Claus. P.2013. An extended ontology-based context model and manipulation calculus for dynamic Web service processes. Journal of Service Oriented Computing and Applications, ISSN 1863-2386. Springer-Verlag London.

[2] Rebei.I. 2012. Informatique ubiquitaire et pervasive . F2B506, Telecom Bretagne, 22 février.

[3] Dey A. K., Abowd. G. D., & Salber.D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context- Aware Applications. Human-computer Interaction, 16 : 97–166,.

[4] Schmidt. A., Aidoo. K. A., Takaluoma. A., U. Tuomela, K. V. Laerhoven, & W. V.de Velde. 1999. Advanced Interaction in Context. In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 89–101, London, UK, Springer- Verlag

[5] Held.A, Buchholz.S, & Schill .A. 2002. A Modeling of Context Information for Pervasive Computing Applications. In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI), Orlando, FL, USA, Jul 14-18

[6] Soukkarieh. 2010. SOUKKARIEH Bouchra "Technique de l'internet et ses langages : vers un système d'information Web restituant des services Web sensibles au contexte. thèse Doctorat, Université de Toulouse III, France, 30 avril.

[7] Ryan. N. 2006. ConteXtML: Exchanging contextual information between a mobile client and the fieldnoteserver. Httpwww.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html.

[8] Sheng .Q. Z & Benatallah. B .2005.ContextUML: A UML Based Modeling Language for Model- Driven Development of Context-Aware Web Services. In The 4th International Conference on Mobile Business(ICMB'05), IEEE Computer Society. Sydney, Australia. July 11-13.

[9] Henricksen.K and Indulska. J. 2004.Modelling and Using Imperfect Context Information. In PerCom Workshops, pp 33–37.

[10] Chevert, K., Mitchell, K., & Davies, N. 1999.Design of an object model for a context sensitive tourist GUIDE. Computers and Graphics 23, 6 883–891.

[11] Schmidt, A., Beigl, M., & Gellersen, H.-W. 1999. There is more to context than location. Computers and Graphics 23, 6, 893–901.

[12] Akman, V., & Surav, M. 1997. The use of situation theory in context modeling. Computational Intelligence 13, 3 427–438.

[13] Chahuara P. 2013. Contrôle intelligent de la domotique à partir d'informations temporelles multi-sources imprécises. Thèse doctorale. s.l., France : Université de Grenoble, 27 mars.

[14] Miao. LV, Chun.JIN, Yoshiyuki.H, & Jim. C. 2013. Ontology-based User Preferences Bayesian Model for Personalized Recommendation. Dalian University of Technology. China, Fukushima University. Japan, Florida Atlantic University .USA, Journal of Computational Information Systems 9: 16 6579–6586.

[15] Strang.T, Linnhoff-Popien.C, & Frank. K. 2003. CoOL: A Context Ontology Language to enable Contextual Interoperability. In J.-B. Stefani, I. Dameure, and D. Hagimon, editors, LNCS 2893 : Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003), volume 2893of Lecture Notes in Computer Science (LNCS), pp 236–247, Paris/France, November. Springer Verlag

[16] Chen, H., Perich, F., Finin, T., & Joshi, A. 2004. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, 22-25 August

[17] Gu, T. et al. (2004) "An ontology-based context model in intelligent environments". Proceedings of Communication Networks and Distributed Systems Modelling and Simulation Conference, San Diego (CA), USA.

[18] Chen, H., Finin, T. and Joshi, A. 2003.Using OWL in a Pervasive Computing Broker. In Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS) .

[19] Chen, H., Finin, T. and Joshi, A. (2004) "An ontology for context aware pervasive computing environments", Knowledge Engineering Review, Vol. 18, No. 3, pp.197–207.

[20] Kanellopoulos D. (2008) "An ontology-based system for intelligent matching of travellers' needs for airlines seats", International Journal of Computer Applications in Technology, Vol. 32, No.3, pp. 194-205.

[21] Kanellopoulos D., Panagopoulos A. (2008) "Exploiting tourism destinations' knowledge in an RDF-based P2P network". Journal of Network and Computer Applications (Elsevier Science), Vol. 31, No. 2, pp.179-200.

[22] Kanellopoulos D. (2009) "Adaptive multimedia systems based on intelligent context management", International Journal of Adaptive and Innovative Systems, Vol. 1, No.1, pp.30-43.

[23] Carrillo R.A. 2007.Agents ubiquitaires pour un accès adapté aux systèmes d'information : Le Framework PUMAS. Thèse pour obtenir le grade de docteur de l'université joseph fourier Spécialité : Informatique, préparée au Laboratoire l'Informatique de Grenoble présentée et soutenue publiquement le 5 mars.

[24] Belhanafi N. 2006. Ajout de mécanismes de réactivité au contexte dans les intergiciels pour composants dans le cadre d'utilisateurs nomades. Thèse présentée pour l'obtention du grade de Docteur de l'Institut National des Télécommunications Soutenue le 27 Novembre.

[25] Chaari.Tand Laforest. F. 2006. Adaptation in Context-Aware Pervasive Information Systems : the secas project, journal of pervasive computing and communications, vol.2, no. 2, june 2006. received: august 2 2005; revised: january 27.

[26] W3C. 2004. Recommendation W3COWL, 2004. http://www.w3.org/TR/owl-ref/

[27] Doulkeridis.C, Loutas.N, &Vazirgiannis.M .2006. A system architecture for context aware service discovery. J Electron Notes Theoretic ComputSci, pp 101–116.

[28] Kapitsaki G, Kateros D, Prezerakos G, & Venierris I. 2009. Model driven development of composite context-aware web applications. J InformSoftwTechnol 51:1244–1260

[29] Goslar K, & Schill A. 2004. modelling contextual information using active data structures. In: Proceedings of the EDBT workshops.Lecture notes in computer science, vol 3268. Springer

[30] Farrar S, & Langendoen DT.2010. An owl-dl implementation of gold- an ontology for the semantic web. Journal of Linguistic modeling of Information and Markup Languages 40:45–66

[31] Wang X, Zhang DQ, Gu T, & Pung H. 2004. Ontology based context modelling and reasoning using owl. In: Proceedings of the 2ndannual conference on pervasive computing and communications workshops. IEEE

[32] Horrocks I, Patel-Schneider F. 2003. Reducing owl entailment to description logic satisfiability. The Semantic Web—ISWC 2003.Lect Notes ComputSci 2870:17–29

[33] Fernandez-Lopez. M & al. 1997. Methontology: from ontological art towards ontological engineering. In Proceedings of the AAAI97 Spring Symposium. Series on ontological engineering. Stanford, CA, (pp, 33-40)

[34] Zacarias, M., Caetano, A., Pinto, S., & Tribolet, J. 2005.Modeling Contexts for Business Process Oriented Knowledge Support. In :Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T. (eds.) : Proceedings of the 3rd Conference on Professional Knowledge Management - Experiences and Visions (WM 2005) (Kaiserslautern, Germany, April 10-13,2005), DFKI, pp. 389-396

[35] Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., & Palinginis, A. 2004. Interplay of Content and Context. In : Koch, N., Fraternali, P., Wirsing, M. (eds.) : Proceedings of the 4th International Conference on Web Engineering (ICWE 2004)(Munich, Germany, July 26-30, 2004), Lecture

Notes in Computer Science, vol. 3140, Springer-Verlag, Berlin Heidelberg, pp. 187-200.

[36] Pittarello, F. 2005. Context-Based Management of Multimedia Documents in 3D Navigational Environments. Proc. of the 11th International Workshop on Multimedia Information Systems (MIS 2005). Lecture Notes in Computer Science 3665, Springer Verlag, 2005, pp. 146-162

[37] Sowa.j. 1984. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley. A , Reading, MA

[38] Baader.F et al,. 2003. The description logic handbook theory, Implementation and Applications, Cambridge University Press.

[39] Driouch.R. 2007. Proposition d'une architecture d'intégration des applications d'entreprise basée sur l'interopérabilité sémantique de l'EbXML et la mobilité des agents. Thèse présentée pour obtenir le diplôme de Doctorat en science.

[40] Keita.A. 2007. Conception coopérative d'ontologies pré-consensuelles : application au domaine de l'urbanisme. Thèse pour l'obtention du diplôme de Doctorat à l'institut national des sciences appliquées, Lyon, Ecole Doctorale Informatique et Information pour la Société 209 pages.

[41] N. Noy, R. W. Fergerson M. & A.Musen. 2000 The knowledge model of Protégé2000: combining interoperability and flexibility. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). JuanLes-Pins, France. Springer-Verlag, LNAI 1937, Berlin, Germany.

[42] Bachimont, B., J. Charlet& R. Troncy. 2004. Ontologies pour le Web Sémantique. Action spécifique 32 CNRS / STIC Web sémantique Rapport final.

[43] Troncy. Bachimont, B., J. Charlet& R.. 2004. Ontologies pour le Web Sémantique. Action spécifique 32 CNRS / STIC Web sémantique Rapport final.

[44] Ketfi. A., Belkhatir. N., P-Y Cunin. 2002. Adaptation Dynamique Concepts et Expérimentations. , In Proceedings of ICSSEA. In French.

[45] Kouadri Mostefaoui. S. & Hirsbrunner.B.2003. Vers une approche orientée contexte pour la découverte et la composition des services dans des environnements mobiles.

[46] Kadima. H. &Montfort.V. 2003. Les services Web: Techniques, démarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML, Dunod. Paris

[47] Ponge, J. 2004. comptabilité et substitution dynamique des web services. Mémoire de fin d'études, université Blaise Pascal Clermont II, juillet.

[48] Msdn. 2012 :. msdn.microsoft.com

[49] Protege 2000 Ontology Editor Home Page, http: //protege.stanford.edu

[50] w3c. 2012. www.w3.org.

# Evaluating the Dual Randomized Kaczmarz Laplacian Linear Solver

Erik G. Boman
Sandia National Laboratories[1]
Center for Computing Research USA
E-mail: egboman@sandia.gov

Kevin Deweese[2] and John R. Gilbert[2]
UC Santa Barbara, Department of Computer Science, USA
E-mail: kdeweese@cs.ucsb.edu and gilbert@cs.ucsb.edu

**Keywords:** Laplacian solver, randomized Kaczmarz, cycle basis

**Received:** May 28, 2015

*A new method for solving Laplacian linear systems proposed by Kelner et al. involves the random sampling and update of fundamental cycles in a graph. Kelner et al. proved asymptotic bounds on the complexity of this method but did not report experimental results. We seek to both evaluate the performance of this approach and to explore improvements to it in practice. We compare the performance of this method to other Laplacian solvers on a variety of real world graphs. We consider different ways to improve the performance of this method by exploring different ways of choosing the set of cycles and the sequence of updates, with the goal of providing more flexibility and potential parallelism. We propose a parallel model of the Kelner et al. method, for evaluating potential parallelism in terms of the span of edges updated at each iteration. We provide experimental results comparing the potential parallelism of the fundamental cycle basis and our extended cycle set. Our preliminary experiments show that choosing a non-fundamental set of cycles can save significant work compared to a fundamental cycle basis.*

*Povzetek:*

## 1 Introduction

### 1.1 Graph Laplacians

The Laplacian matrix of a weighted, undirected graph is defined as $L = D - A$, where $D$ is the diagonal matrix containing the sum of incident edge weights and $A$ is the weighted adjacency matrix. The Laplacian is symmetric and positive semidefinite. The Laplacian is also defined for directed graphs [2]. Because they are not symmetric, most of the solvers discussed in this paper do not apply, and efficient solution techniques remain an open problem. Solving linear systems on the Laplacians of structured graphs, such as two and three dimensional meshes, has long been important in finite element analysis (with applications in electrical and thermal conductivity, and fluid flow modeling [6]) and image processing (with applications in image segmentation, inpainting, regression, and classification [12, 20, 22]). More recently, solving linear systems on the graph Laplacians of large graphs, with irregular degree distributions, has emerged as an important computational task in network analysis (with applications to maximum flow problems [8], graph sparsification [24], and spectral clustering [17]).

Most applied work on Laplacian solvers has been on preconditioned conjugate gradient (PCG) solvers, including support graph preconditioners [4, 5, 13], or on specialized multigrid methods [20, 21]. Several of these methods seem efficient in practice, but none have asymptotic performance guarantees based on the size of the graph.

The theoretical computer science community has developed several methods, which we refer to generally as combinatorial Laplacian solvers. These solvers have good complexity bounds but, in most cases, no reported experimental results. Spielman and Teng [25] first showed how to solve these problems in linear times polylogarithmic work, later improved upon by Koutis, Miller, and Peng [19], but their algorithms do not yet have a practical implementation. An algorithm proposed by Kelner et al. [16] has the potential to solve these linear systems in linear times polylogarithmic work with a simple, implementable algorithm.

### 1.2 The dual randomized Kaczmarz algorithm

The inspiration for the algorithm proposed by Kelner et al. [16], which we refer to as Dual Randomized Kaczmarz (DRK), is to treat graphs as electrical networks with resistors on the edges. For each edge, the weight is the inverse

[1]Sandia is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

[2]Supported by Contract #618442525-57661 from Intel Corp. and Contract #DE-AC02-05CH11231 from DOE Office of Science.

of the resistance. We can think of vertices as having an electrical potential and a net current at every vertex, and define vectors of these potentials and currents as $v$ and $f$ respectively. These vectors are related by the linear system $Lv = f$. Solving this system is equivalent to finding the set of voltages that satisfies the net "injected" currents. Kelner et al.'s DRK algorithm solves this problem with an optimization algorithm in the dual space, which finds the optimal currents on all of the edges subject to the constraint of zero net voltage around all cycles. They use Kaczmarz projections [15] to adjust currents on one cycle at a time, iterating until convergence.

We will also refer to the Primal Randomized Kaczmarz (PRK) method that applies Kaczmarz projections in the primal space [26]. One sweep of PRK performs a Kaczmarz projection with every row of the matrix. Rows are taken in random order at every sweep.

DRK iterates over a set of *fundamental cycles*, cycles formed by adding individual edges to a spanning tree $T$. The fundamental cycles are a basis for the space of all cycles in the graph [11]. For each off-tree edge $e$, we define the resistance $R_e$ of the cycle $C_e$ that is formed by adding edge $e$ to the spanning tree as the sum of the resistances around the cycle,

$$R_e = \sum_{e' \in C_e} r_{e'}$$

which is thought of as approximating the resistance of the off-tree edge $r_e$. DRK chooses cycles randomly, with probability proportional to $R_e/r_e$.

The performance of the algorithm depends on the sum of these approximation ratios, a property of the spanning tree called the *tree condition number*

$$\tau(T) = \sum_{e \in E \setminus T} \frac{R_e}{r_e}.$$

The number of iterations of DRK is proportional to the tree condition number. Kelner et al. use a particular type of spanning tree with low tree condition number, called a *low stretch tree*. Specifically, they use the one described by Abraham and Neiman [1] with $\tau = O(m \log n \log \log n)$, where $n$ refers to the number of vertices and $m$ refers to the number of edges of the original graph. The work of one iteration is naively the cycle length, but can be reduced to $O(\log n)$ with a fast data structure, yielding $O(m \log n^2 \log \log n)$ total work.

### 1.3 Overview

The rest of the paper is organized as follows. In Section 2 we survey the related experimental work. Section 3 is an initial evaluation of the DRK algorithm as compared to PCG and PRK. We present our new ideas for improving the performance of DRK in Section 4. In this section we also consider how to perform cycle updates in parallel. Section 5 is an evaluation of the new ideas proposed in Section 4.

## 2 Related experimental work

As the DRK algorithm is a recent and theoretical result, there are few existing implementations or performance results. Hoske et al. implemented the DRK algorithm in C++ and did timing comparisons against unpreconditioned CG on two sets of generated graphs [14]. They concluded that the solve time of DRK does scale nearly linearly. However, several factors make the running time too large in practice, including large tree stretch and cycle updates with unfavorable memory access patterns. They cite experimental results by Papp [23], which suggest that the theoretically low stretch tree algorithms are not significantly better than min-weight spanning trees in practice, at least on relatively small graphs.

Chen and Toledo [7] experimented with an early and somewhat different combinatorial approach to Laplacians called support graph preconditioners. They demonstrated that support graph preconditioners can outperform incomplete Cholesky on certain problems. There has also been some experimental work in implementing the local clustering phase of the Spielman and Teng algorithm [27].

## 3 Initial evaluation of DRK and comparison to PCG and PRK

### 3.1 Experimental design

Our initial study of DRK measures performance in terms of work instead of time, and uses a somewhat more diverse graph test set than Hose et al. [14]. We implemented the algorithm in Python with Cython to see how it compared against PCG (preconditioned with Jacobi diagonal scaling) and PRK. However, we did not implement a low stretch spanning tree. Instead we use a low stretch heuristic that ranks and greedily selects edges by the sum of their incident vertex degrees (a cheap notion of centrality). In practice this works well on unweighted graphs. We also did not implement the fast data structure Kelner et al. use to update cycles in $O(\log n)$ work.

Our results do not include wall clock time, since our DRK implementation is not highly optimized. Instead we are interested in measuring the total work. For PCG the work is the number of nonzeros in the matrix for every iteration, plus the work of applying the preconditioner at every iteration (number of vertices for Jacobi). For PRK the work is the number of nonzero entries of the matrix for every sweep, where a sweep is a Kaczmarz projection against all the rows of $L$. As the DRK work will depend on data structures and implementation, we consider four different costs for estimating the work of updating a single cycle, which we refer to as cost metrics. The first metric is the cost of updating every edge in a cycle, which is included because it is the naive implementation we are currently using. The second metric relies upon the data structure described by Kelner et al., which can update the fundamental cycles in $O(\log n)$ work. This may be an overestimate when the cycle length is actually less than $\log n$. The third metric considers a hypothetical $\log$ of cycle length update method which we do not know to exist, but is included as a hopeful estimate of a potentially better update data structure. The last metric costs $O(1)$ work per cycle, which is included because we surely cannot do better than this.

**Metric 1.** cycle length (naive)

**Metric 2.** $\log n$ (using fast update data structure)

**Metric 3.** $\log(\text{cycle length})$ (optimistic)

**Metric 4.** $1$ (lower bound)

We ran experiments on all the mesh-like graphs and irregular graphs shown in Appendix Table 1. Mesh-like graphs come from more traditional applications such as model reduction and structure simulation, and contain a more regular degree distribution. Irregular graphs come from electrical, road, and social networks, and contain a more irregular, sometimes exponential, degree distribution. Most of these graphs are in the University of Florida (UF) sparse matrix collection [10]. We added a few 2D and 3D grids along with a few graphs generated with the BTER generator [18]. We removed weights and in a few cases symmetricized the matrices by adding the transpose. We pruned the graphs to the largest connected component of their 2-core, by successively removing all degree 1 vertices, since DRK operates on the cycle space of the graph. The difference between the original graph and the 2-core is trees that are pendant on the original graph. These can be solved in linear time so we disregard them to see how solvers compare on just the structurally interesting part of the graph.

We solve to a relative residual tolerance of $10^{-3}$. Accuracy in the solution is sacrificed in order to run more experiments and on larger graphs. The Laplacian matrix is singular with a nullspace dimension of one (because the pruned graph is connected). For DRK and PRK this is not a problem, but for PCG we must handle the non-uniqueness of the solution. Our choice for handling singularity is to remove the last row and column of the matrix. One could also choose to orthogonalize the solution against the nullspace inside the algorithm, but in our experience the performance results are similar.

We also ran a set of PCG vs. DRK experiments where the convergence criteria is the actual error within $10^{-3}$. Actual error can be calculated by knowing the solution in advance. One of the interesting results of the DRK algorithm is that, unlike PCG and PRK, convergence does not depend on the condition number of the matrix, but instead just on the tree condition number. Since higher condition number can make small residuals less trustworthy, we wondered whether convergence in the actual error yields different results.

## 3.2 Experimental results

We compare DRK to the other solvers by examining the ratio of DRK work to the work of the other solvers. The ratio of DRK work to PRK work is plotted in Figure 1, separated by graph type. Each vertical set of four points are results for a single graph, and are sorted on the x axis by graph size. The four points represent the ratio of DRK work to PRK work under all four cost metrics. Points above the line indicate DRK performed more work while points below the line indicate DRK performed less work. Similar results for the PCG comparison are shown in Figure 2. Another set of PCG comparisons, converged to the actual error, is shown in Figure 3.

An example of the convergence behavior on the USpowerGrid graph is shown in Figure 4. This plot indicates how both the actual error and relative error behave during the solve for both PCG and DRK. A steeper slope indicates faster convergence. Note this only shows metric 1 work for DRK.

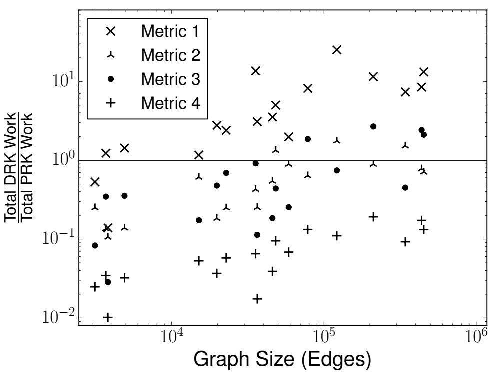
## 3.3 Experimental analysis

In the comparison to PRK (shown in Figure 1), DRK is often better with cost metrics 3 and 4. On a few graphs, mostly in the irregular category, DRK outperforms PRK in all cost metrics (all the points are below the line). In the comparison to PCG (shown in Figure 2), DRK fares slightly better for the irregular graphs, but on both graph sets these results are somewhat less than promising. PCG often does better (most of the points are above the line). Even if we assume unit cost for cycle updates, PCG outperforms DRK. The performance ratios also seem to get worse as graphs get larger.

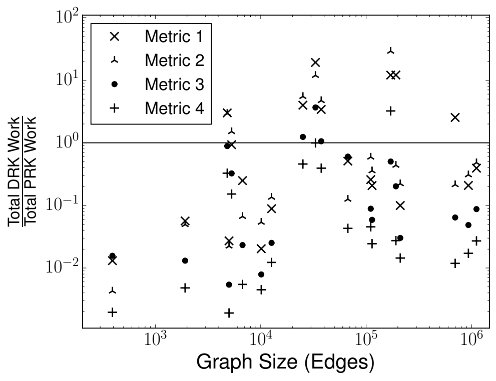
The results concerning the actual error (shown in Figure 3) are very interesting as they are quite different than those with the residual tolerance. For all of the mesh graphs, considering the actual error makes DRK look more promising. The relative performance of cost metrics 3 and 4 are now typically better for DRK than PCG. However, PCG is still consistently better with cost metrics 1 and 2. For some of the irregular graphs, the convergence behavior is similar, but for others things look much better when considering actual error. Informally the number of edges updated by DRK did not change much when switching convergence criteria, but PCG work often increased. The USpowerGrid example (shown in Figure 4) gives a sense of this. The residual error and actual error decrease similarly for DRK, but the actual error curve for PCG decreases much more slowly for the actual error.

## 4 New algorithmic ideas

We consider ways in which DRK could be improved by altering the choice of cycles and their updates. Our goals are both to reduce total work and to identify potential parallelism in DRK. To this end we are interested in measuring

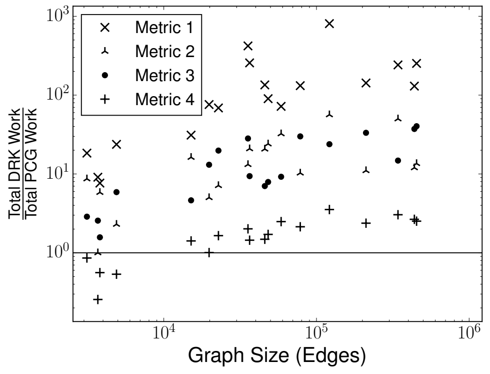


(a) Mesh-like Graphs

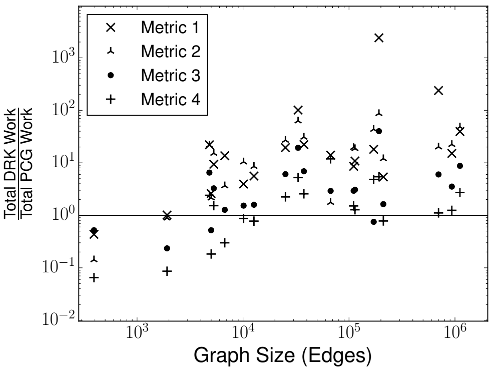


(b) Irregular Graphs

Figure 1: DRK vs. PRK: Relative work of DRK to PRK work under the four cost metrics is shown (PRK is better than DRK at points above the line).

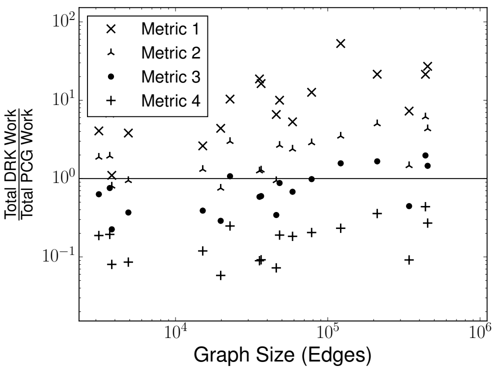


(a) Mesh-like Graphs

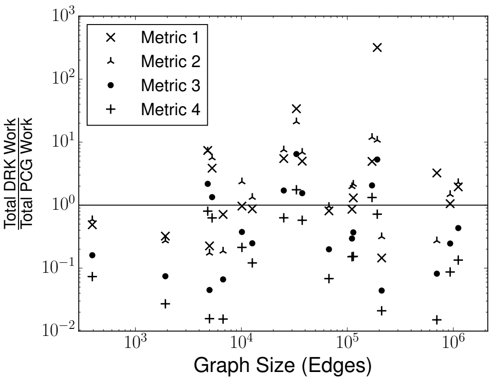


(b) Irregular Graphs

Figure 2: DRK vs. PCG: Relative work of DRK to PCG work under the four cost metrics is shown (PCG is better than DRK at points above the line).



(a) Mesh-like Graphs



(b) Irregular Graphs

Figure 3: DRK vs. PCG Converged to Actual Error: Relative work of DRK to PCG work under the four cost metrics is shown, convergence tolerance is norm of actual error within $10^{-3}$.

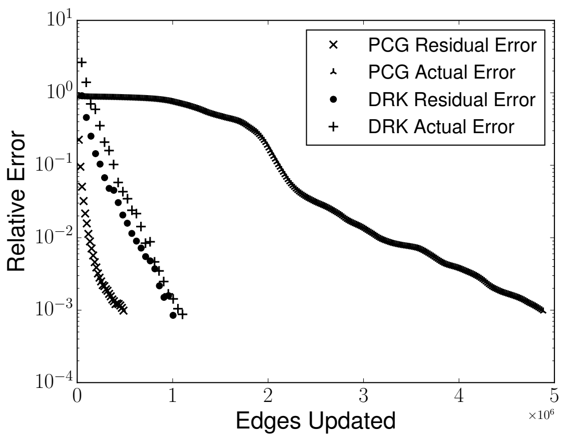


Figure 4: DRK and PCG Convergence Behavior on US-powerGrid: Relative residual error and actual error are shown for both solvers over the iterations required for convergence.

the number of *parallel steps*, the longest number of steps a single thread would have to perform before before convergence, maximized over all threads. Parallel steps are measured in terms of the four cost metrics described in Section 2. We will also define the *span* [9], or critical path length, which is the number of parallel steps with unbounded threads.

## 4.1 Expanding the set of cycles

Sampling fundamental cycles with respect to a tree may require updating several long cycles which will not be edge-disjoint. It would be preferable to update edge-disjoint cycles, as these updates could be done in parallel. The cycle set we use does not need to be a basis, but it does need to span the cycle space. In addition to using a cycle basis from a spanning tree, we will use several small, edge-disjoint cycles. We expect that having threads update these small cycles is preferable to having them stand idle.

### 4.1.1 2D grid example

A simple example of a different cycle basis is the 2D grid graph, shown in Figure 5. In the original DRK, cycles are selected by adding off-tree edges to the spanning tree as in Figure 5(a). As the 2D grid graph is planar, the faces of the grid are the regions bounded by edges, and we refer to the cycles that enclose these regions as *facial cycles*. We consider using these cycles to perform updates of DRK, as the facial cycles span the cycle space of a planar graph [11]. Half of these cycles can be updated at one iteration and then the other half can be updated during the next iteration, in a checkerboard fashion, as in Figures 5(b)(c). Furthermore, to speed up convergence, smaller cycles can be added together to form larger cycles (in a multilevel fashion) as in Figure 5(d).

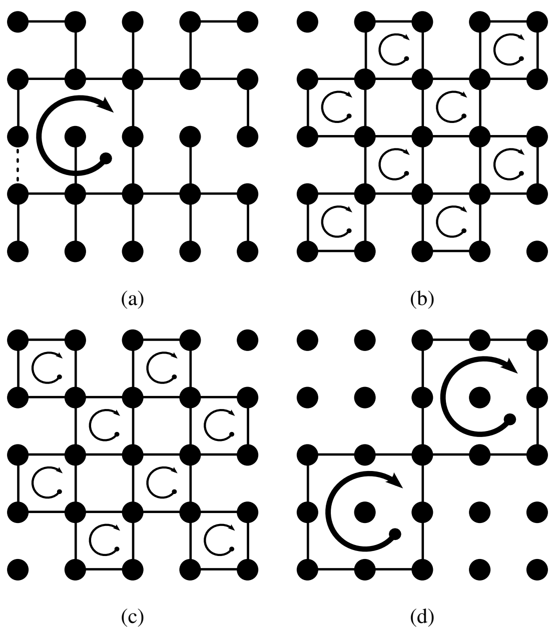


Figure 5: Grid Cycles: (a) Fundamental cycles are formed by adding edges to the spanning tree. (b-c) First level facial cycles are shown, grouped into edge-disjoint sets. (d) Second level facial cycles are formed by adding smaller facial cycles.

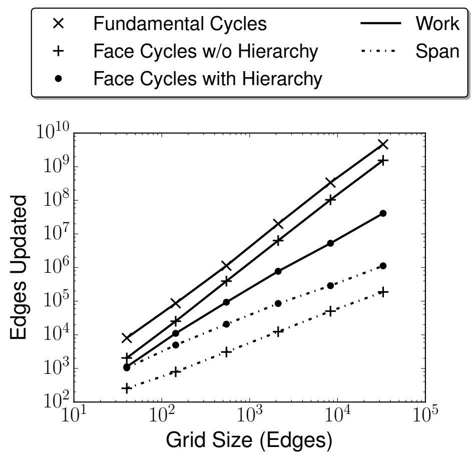


Figure 6: Grid Cycle Performance: Work and span of DRK using facial cycles and fundamental cycles for two dimensional grids of various sizes.

We implemented such a cycle update scheme using the grid facial cycles, and performed experiments to see how the facial cycles affected the total work measured in both the number of cycles updated (metric 4) and edges updated (metric 1). With the facial cycles, the span per iteration is the cost of updating two cycles at each level. We ran experiments with and without the hierarchical combination of the facial cycles against the original set of fundamental cycles. In the case of the fundamental cycles we use H trees [3], which have optimal stretch $O(\log n)$. Solutions were

**Algorithm 1** Local Greedy Finder.

```
function LOCAL-GREEDY(G)
    for $e_{i,j} \in E$ do
        if $e_{i,j}$ unmarked then
            $p_{i,j}$ = Truncated-BFS$(G \setminus (e_{i,j}), i, j, max\_edges)$
            Add $p_{i,j} + e_{i,j}$ to cycle set
            Mark all edges in $p_{i,j} + e_{i,j}$
        end if
    end for
end function
```

calculated to a residual tolerance of $10^{-6}$. The accuracy here is slightly better than the rest of the experiments since these experiments were faster.

The results shown in Figure 6 indicate that the facial cycles improve both the work and span. Using a hierarchical update scheme reduces the total number of edges updated. However as this requires updating larger cycles it has a worse span than simply using the lowest level of cycles.

### 4.1.2 Extension to general graphs

We refer to small cycles we add to the basis as local greedy cycles. Pseudocode for finding these cycles is shown in Algorithm 1. We construct this cycle set by attempting to find a small cycle containing each edge using a truncated breadth-first search (BFS). Starting with all edges unmarked, the algorithm selects an unmarked edge and attempts to find a path between its endpoints. This search is truncated by bounding the number of edges searched so that each search is constant work and constructing the entire set is $O(m)$ work. If found, this path plus the edge forms a cycle, which is added to the new cycle set, and all edges used are marked. Appendix Table 1 shows the number of local greedy cycles found for all the test graphs when the truncated BFS was allowed to search 20 edges. Greedy cycles were found in all the graphs except for tube1, all of whose vertices had such high degree that searching 20 edges was not enough to find a cycle.

Adding additional cycles to the cycle basis means we need new probabilities with which to sample all the cycles. Since in the unweighted case, the stretch of a cycle is just its total length, it seems natural to update cycles proportional to their length.

## 4.2 Cycle sampling and updating in parallel

In the original DRK algorithm, cycles are chosen one at a time with probability proportional to stretch. We propose a parallel update scheme in which multiple threads each select a cycle, at every iteration, with probability proportional to cycle length. If two threads select cycles that share an edge, one of the threads goes idle for that iteration. In Figure 8, threads 1, 2, and 4 select edge-disjoint cycles. However the third processor selects a cycle which contains edge 3, which is already in use by the cycle on thread 1. Processor 3 sits this iteration out while the other processors update their cycles.

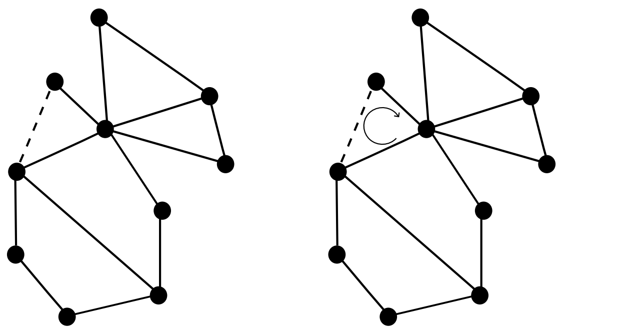


Figure 7: Local Greedy Cycles: An edge is selected on the left and a local greedy search is performed to find the cycle on the right.

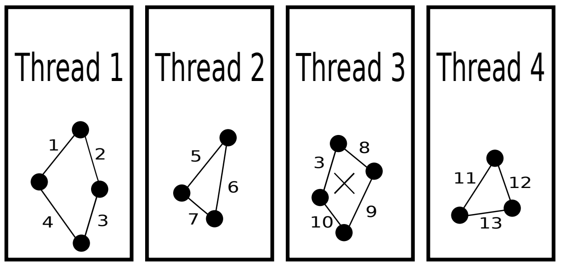


Figure 8: Example of Processors Selecting Cycles: Threads 1, 2, and 4 select edge-disjoint cycles, but thread 3 selects a cycle with edge 3 already in use. Thread 3 will go idle for an iteration.

We compute several measures of parallel performance. The first is simply the number of iterations. The second is the total work across all threads at every iteration. Lastly we measure the span, or critical path length. This is the maximum of the work over all threads, summed over all the iterations.

We envision threads working in a shared memory environment on a graph that fits in memory. This might not be realistic in practice as there must be some communication of which edges have already been used which might be too expensive relative to the cost of a cycle update. However we are simply interested in measuring the potential parallelism, thus we ignore any communication cost.

The parallel selection scheme conditions the probabilities with which cycles are selected on edges being available

$$p(C_e) = \frac{1}{\tau}\frac{R_e}{r_e}p(e' \in C_e \text{ available}).$$

This scheme creates a bias towards smaller cycles with less conflicting edges as more threads are added, which can increase total work.

## 5 Experiments and results

### 5.1 Experimental design

We performed experiments on a variety of unweighted graphs from the UF Sparse Matrix Collection (the same set as in Section 2, shown in Appendix Table 1). Again we distinguish between mesh-like graphs and irregular graphs. We also use a small test set for weak scaling experiments, consisting of 2D grids and BTER graphs.

We continue to use our Python/Cython implementation of DRK, without a guaranteed low stretch spanning tree or a cycle update data structure. The code does not run in parallel, but we simulate parallelism on multiple threads by selecting and updating edge-disjoint cycles at every iteration as described above.

Our experiments consist of two sets of strong scaling experiments, the spanning tree cycles with and without local greedy cycles, up to 32 threads. We set a relative residual tolerance of $10^{-3}$. Again we sacrifice accuracy to run more experiments on larger graphs. We consider the same four cycle update cost metrics as in Section 2: cycle length, $\log n$, $\log(\text{cycle length})$, and unit cost update. However in the case of the local greedy cycles, which cannot use the $\log n$ update data structure, we always just charge the number of edges in a cycle. For all the cost models, we measure the total work required for convergence and the number of parallel steps taken to converge. For metric 4 these will be the same. A condensed subset of the scaling results is shown in Appendix Table 2.

### 5.2 Experimental results

First, we examine the effects of using an expanded cycle set in the sequential algorithm. We estimate the usefulness of extra cycles as the length of the largest cycle in the fundamental set normalized by the number of cycles in the fundamental set. This is because we suspect the large cycles to be a barrier to performance, as they are updated the most frequently, and at the highest cost. The performance of the local greedy cycles for the two different graph types, using metrics 1 and 4, is shown in Figure 9. These plots show the ratio between the work of the expanded cycle sets as a function of the estimated usefulness. Points below the line indicate that adding local greedy cycles improved performance.

We examine how the local greedy cycles perform as graph size increases with weak scaling experiments on 2D grid graphs and BTER graphs. The 2D grids used for this experiment are the same as in Figure 6, and the BTER graphs were generated with the parameters: average degree of 20, maximum degree of $\sqrt{n}$, global clustering coefficient of 0.15, and maximum clustering coefficient of 0.15. The performance of cost metric 1 as graph size scales is shown in Figure 10.

Figure 11 shows examples of our results on three of the graphs. In Figure 11(a) the parallel steps (with the four different metrics) is plotted as a function of the number of threads used for the barth5 graph. The total edges (metric 1) is at the top of the plot, while the unit cost (metric 4) is at the bottom. These results are shown for both fundamental and extended cycle sets. Figure 12 shows the effect of adding threads to the total work.

To measure the parallel performance across multiple graphs we look at the average speedup of the parallel steps across all graphs. Speedup is defined as the sequential work using one thread over the number of parallel steps using a number of multiple threads. The speedup with and without extended cycles is shown in Figure 13. Note that without local greedy cycles metric 2 and metric 4 speedup are the same as the costs differ by $\log n$. We compare the speedup between the different cycle sets for the different graph types in Figure 14. Results are shown only for metric 1. The speedup of using 8 threads without local greedy is plotted against the speedup of using 8 threads with local greedy.

### 5.3 Experimental analysis

In the sequential results shown in Figure 9, there seems to be a threshold of largest cycle length above which local greedy cycles can be useful, but below which there is not much difference. However, there is not a clear scaling with the cycle length ratio, indicating that this is still a crude guess as to where the extended cycle set is useful. Also note that mesh-like graphs tend to have larger girth (max cycle length) than irregular graphs, leading to local greedy cycles working better on meshes. The local greedy cycle improvement is slightly better for metric 1 where we count every edge update. At the other extreme, when updating large cycles is the same cost (unit) as small cycles added by local greedy, the local cycles are less effective. However there is still an improvement in number of cycles updated. We were unable to find some measure of the usefulness of a single local greedy cycle.

The weak scaling experiments shown in Figure 10 show that, with the exception of a BTER outlier, the work scales nicely with graph size. Also results for both cycle sets scale similarly. The extended cycle set benefits the 2D grid graphs while the BTER graphs see little improvement or are worse. This is consistent with Figure 9 since the BTER graphs are irregular and the 2D grids are mesh-like.

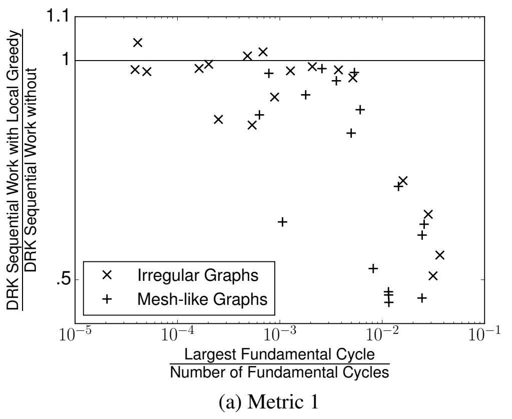


Figure 9: Sequential Comparison of Cycle Set Work: The ratio of DRK work with and without local greedy cycles, on one thread, is plotted against an estimate of the usefulness of extra cycles. Points below the line indicate that adding local greedy cycles helped.

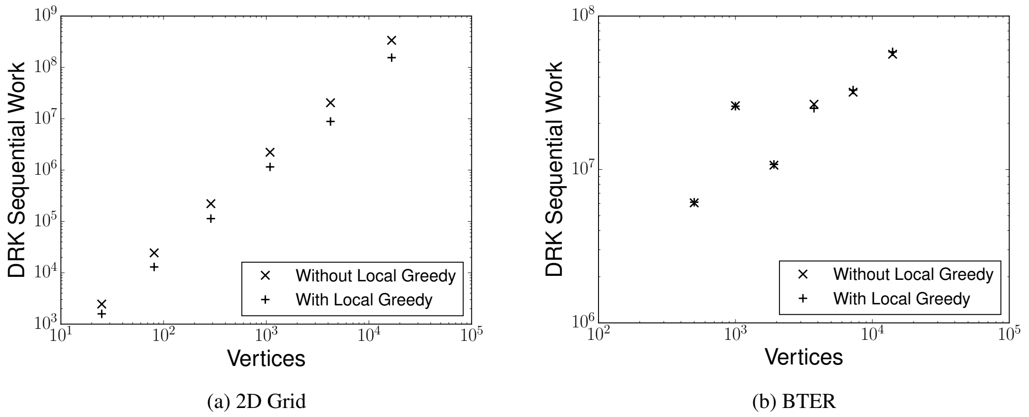


Figure 10: Weak Scaling of Cycle Set Work Under Cost Metric 1: The DRK work with and without local greedy cycles, on one thread, is plotted against the graph size in vertices.

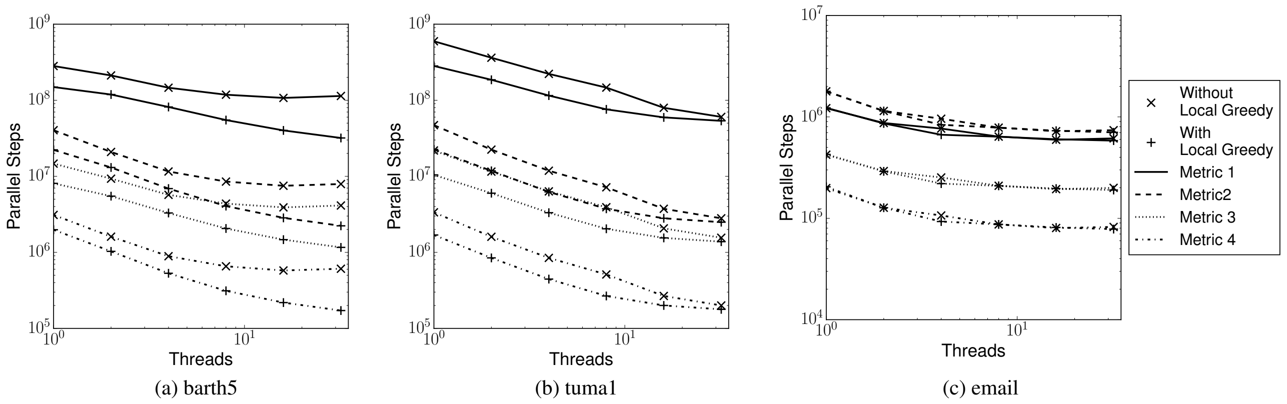


Figure 11: Parallel Steps Scaling (shown for three example graphs): As threads are added, parallel steps decreases for both cycle sets (steeper slope indicates better scaling).

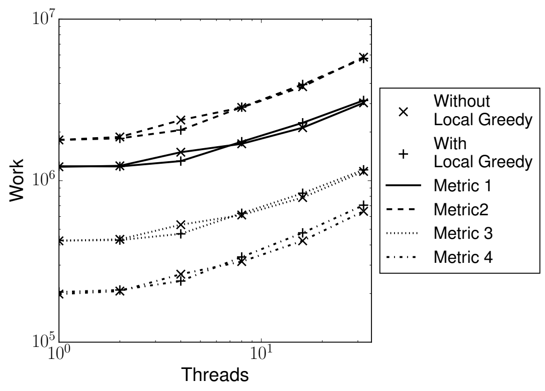


Figure 12: Total Work Scaling of email Graph: As threads are added the total work increases for both cycle sets (ideally it would stay constant).

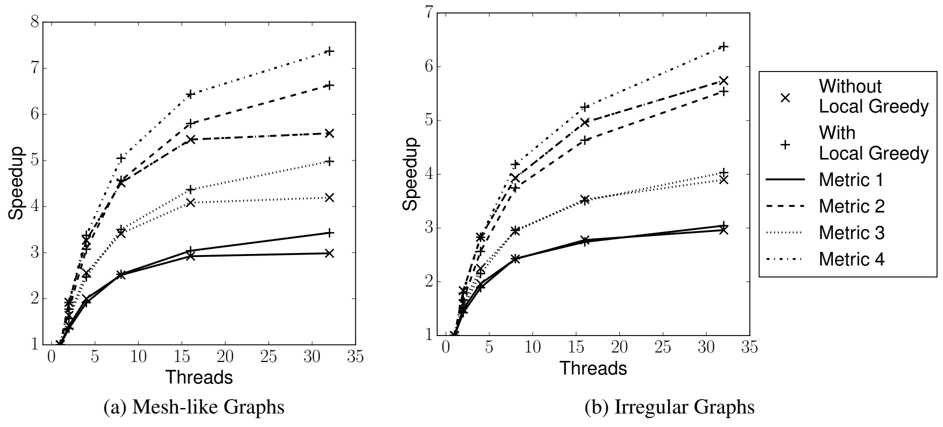


Figure 13: Average Parallel Steps Speedup: The ratio of sequential work on one thread to parallel steps on multiple threads is plotted up to 32 threads.

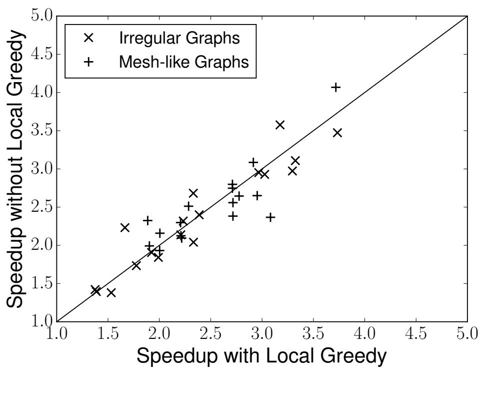


Figure 14: 8 Thread Speedup Comparison: The ratio of the 8 thread speedups of both cycle sets are plotted for all graphs (below the line local greedy speedup is better).

The scaling of parallel steps plots show a variety of different behavior on the example graphs. On the mesh-like barth5 graph (shown in Figure 11(a)), the local greedy cycles improve both sequential performance and the scaling of parallel steps performance. At the left of this plot we see the extra cycles improve sequential results. Then as threads are added in parallel, the steeper slope indicates the local greedy cycles improved the scaling of the parallel steps. On the tuma1 graph (shown in Figure 11(b)), the local greedy cycles improve sequential performance, but result in similar or worse scaling. At the left of this plot we see the extra cycles improve results sequentially, but when scaled to 32 threads performance is similar. On the email graph (shown in Figure 11(c)), the local greedy cycles do not improve sequential performance, and scaling is poor with both cycle sets. There is little difference between the different cycle sets in this plot. Furthermore scaling is poor and quickly flattens out by about four threads. For a better understanding of the poor parallel steps scaling on the email graph, we examine the total work scaling (shown in Figure 12), showing how much extra work we have to do when skewing the probability distribution. This extra work quickly increases, limiting the parallel performance.

In the average parallel steps speedup plot (shown in Figure 13), we see similar speedup for both cycle sets. On mesh-like graphs the local greedy cycles do slightly better on all cost metrics beyond 16 threads. However on the irregular graphs, only with cycle cost metric 4 do the local greedy cycles perform better, and under metric 3 they perform worse. (Again note that without local greedy cycles metric 4 and metric 3 speedups are the same). We hypothesized that giving the solver smaller, extra cycles would improve the parallel performance compared to the fundamental cycles. However this seems to only be true for mesh-like graphs, and even then the improvement is minimal. An interesting thing to note is that the speedup is better with the $\log n$ cost model. This is probably due to overcharging small cycles, which is less problematic when there are more threads to pick potentially larger cycles.

Taking a snapshot of the parallel steps speedup results on eight threads (shown in Figure 14), we see that there are some irregular graphs which do not have much speedup for either cycle set (bottom left of the plot). However there are mesh-like and irregular graphs which enjoy a speedup for both cycle sets (top right of the plot). It is difficult to say on which graphs will different cycles aid with parallelism.

## 6 Conclusion

We have done an initial comparison of Kelner et al.'s DRK algorithm with PCG and PRK. These preliminary results, measuring algorithm work by number of edges touched or by number of cycles updated, do not at present support the practical utility of DRK. For mesh-like graphs, PCG usually takes less work than DRK, even if DRK is charged only one unit of work per cycle update. This suggests that the fast cycle update data structure proposed by Kelner et al. (or any undiscovered fast update method) will not be enough to make DRK practical. It does seem that DRK is an improvement to PRK on several graphs, mostly irregular graphs. One promising result of these experiments is that DRK converges to small actual error similarly to residual error, while PCG sometimes does not. More PCG iterations are required when solving to a low actual error, while DRK work does not increase very much. More work should be done to understand this behavior.

The experiments in this paper were limited to unweighted graphs for simplicity. Experiments with weighted graphs should be run for more complete results. An open question is whether there is a class of graphs with high condition number, but with practical low stretch trees, where DRK will perform significantly better in practice.

We suggest techniques for improving DRK in practice. One possible improvement is to use a spanning set, including non-fundamental cycles, to accelerate convergence. Using facial cycles of a two-dimensional grid graph greatly reduces the required number of edge updates compared to the fundamental cycle basis. We try to generalize these cycles by finding small local greedy cycles. These cycles can accelerate convergence, especially for mesh-like graphs. It is difficult to measure the usefulness of any one cycle in the basis, so it is difficult to determine where and which extra cycles are useful.

We also consider how DRK could be implemented in parallel to take advantage of simultaneous updates of edge disjoint cycles. We describe a model in which threads select cycles, and go idle if a conflicting edge is found. While this can increase total work, it can often reduce the number of parallel steps. However there is a limit to this parallelism. Furthermore, scaling behavior seems to be similar with or without local greedy cycles.

## References

[1] I. Abraham and O. Neiman (2012) Using petal-decompositions to build a low stretch spanning tree, *Proceedings of the 44th annual ACM Symp. on Theory of Comp.*, ACM, New York, NY, USA, pp. 395–406.

[2] R. Agaev and P. Chebotarev (2005) On the spectra of nonsymmetric Laplacian matrices, *Linear Algebra and its Appl.*, Elsevier, pp. 157–168.

[3] N. Alon, M. Karp, D. Peleg, and D. West (1995) A graph-theoretic game and its application to the k-server problem, *SIAM Journal on Comp.*, SIAM, pp. 78–100.

[4] M. Bern, J. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo (2006) Support-graph preconditioners, *SIAM Journal Matrix Anal. Appl.*, SIAM, pp. 930–951.

[5] E. G. Boman, D. Chen, B. Hendrickson, and S. Toledo (2004) Maximum-weight-basis preconditioners, *Numerical Linear Algebra Appl.*, Wiley, pp. 695–721.

[6] E. G. Boman, B. Hendrickson, and S. Vavasis (2008) Solving elliptic finite element systems in near-linear time with support preconditioners, *SIAM Journal on Numerical Anal.*, SIAM, pp. 3264–3284.

[7] D. Chen and S. Toledo (2003) Vaidya's preconditioners: Implementation and experimental study, *Elec-*

*tronic Transactions on Numerical Anal. [electronic only]*, pp. 30–49.

[8] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng (2011) Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs, *Proceedings of the 43rd annual ACM Symp. on Theory of Comp.*, ACM, San Jose, CA, USA, pp. 273–282.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein (2009) *Introduction to Algorithms*, MIT Press and McGraw-Hill.

[10] T. A. Davis and Y. Hu (2011) The University of Florida sparse matrix collection, *ACM Transactions on Mathematical Software*, ACM, pp. 1:1–1:25.

[11] R. Diestel (2012) *Graph Theory*, Springer.

[12] L. Grady (2006) Random walks for image segmentation, *IEEE Transactions on Pattern Anal. and Machine Intelligence*, IEEE, pp. 1768–1783.

[13] K. Gremban (1996) *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

[14] D. Hoske, D. Lukarski, H. Meyerhenke, and M. Wegner (2015) Is nearly-linear the same in theory and practice? A case study with a combinatorial Laplacian solver, *Computing Research Repository [electronic only]*, http://arxiv.org/abs/1502.07888.

[15] S. Kaczmarz (1937) Angenäherte auflösung von systemen linearer gleichungen, *Bulletin International de l'Academie Polonaise des Sciences et des Lettres*, pp. 355–357.

[16] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu (2013) A simple, combinatorial algorithm for solving SDD systems in nearly-linear time, *Proceedings of the 45th ACM Symp. Theory of Comp.*, ACM, Palo Alto, CA, USA, pp. 911–920.

[17] N. L. D. Khoa and S. Chawla (2015) A scalable approach to spectral clustering with SDD solvers, *Journal of Intelligent Info. Sys.*, Springer, pp. 289–308.

[18] T. G. Kolda, A. Pinar, T. Plantenga, and C. Seshadhri (2014) A scalable generative graph model with community structure, *SIAM Journal on Scientific Comp.*, SIAM, pp. C424–C452.

[19] I. Koutis, G. L. Miller, and R. Peng (2014) Approaching optimality for solving SDD systems, *SIAM Journal on Comp.*, SIAM, pp. 337–354.

[20] I. Koutis, G. L. Miller, and D. Tolliver (2011) Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing, *Computer Vision and Image Understanding*, Elsevier, pp. 1638–1646.

[21] O. E. Livne and A. Brandt (2012) Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver, *SIAM Journal on Scientific Comp.*, SIAM, pp. B499–B522.

[22] J. McCann and N. S. Pollard (2008) Real-time gradient-domain painting, *ACM Transactions on Graphics*, ACM, pp. 93:1–93:7.

[23] P. A. Papp (2014) *Low-Stretch Spanning Trees*, Undergraduate thesis, Eötvös Loránd University, Budapest, Hungary.

[24] D. A. Spielman and N. Srivastava (2011) Graph sparsification by effective resistances, *SIAM Journal on Comp.*, SIAM, pp. 1913–1926.

[25] D. A. Spielman and S. Teng (2004) Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, *In Proceedings of the 36th Annual ACM Symp. on Theory of Comp.*, ACM, New York, NY, USA, pp. 81–90.

[26] T. Strohmer and R. Vershynin (2009) A randomized Kaczmarz algorithm with exponential convergence, *Journal of Fourier Anal. and Appl.*, Springer, pp. 262–278.

[27] Z. A. Zhu, S. Lattanzi, and V. S. Mirrokni (2013) A local algorithm for finding well-connected clusters, *Proceedings of the 30th International Conference on Machine Learning*, JMLR Workshop and Conference Proceedings, Atlanta, GA, USA, pp. 396–404.

| Graph<br>(Collection) | Nodes | Edges | 2-core<br>Nodes | 2-Core<br>Edges | Greedy<br>Cycles | Probability of<br>Selecting Greedy | Largest<br>Cycle Length |
|---|---|---|---|---|---|---|---|
| jagmesh3 (HB) | 1.09k | 3.14k | 1.09k | 3.14k | 1.92k | 0.2419 | 77 |
| lshp1270 (HB) | 1.27k | 3.70k | 1.27k | 3.70k | 2.17k | 0.4712 | 95 |
| rail_1357<br>(Oberwolfach) | 1.36k | 3.81k | 1.36k | 3.81k | 1.85k | 0.2507 | 55 |
| 50 x 50 grid | 2.50k | 4.90k | 2.50k | 4.90k | 2.40k | 0.5000 | 120 |
| data (DIMACS10) | 2.85k | 15.1k | 2.85k | 15.1k | 7.43k | 0.1760 | 92 |
| 100 x 100 grid | 10.0k | 19.8k | 10.0k | 19.8k | 9.80k | 0.5000 | 230 |
| 20 x 20 x 20 grid | 8.00k | 22.8k | 8.00k | 22.8k | 3.57k | 0.1941 | 122 |
| L-9 (A-G Monien) | 18.0k | 35.6k | 18.0k | 35.6k | 17.6k | 0.4992 | 411 |
| tuma1<br>(GHS_indef) | 23.0k | 37.2k | 22.2k | 36.5k | 10.7k | 0.0610 | 420 |
| barth5 (Pothen) | 15.6k | 45.9k | 15.6k | 45.9k | 29.9k | 0.1765 | 375 |
| cti (DIMACS10) | 16.8k | 48.2k | 16.8k | 48.2k | 7.27k | 0.0501 | 172 |
| aft01 (Okunbor) | 8.21k | 58.7k | 8.21k | 58.7k | 26.6k | 0.6680 | 105 |
| 30 x 30 x 30 grid | 27.0k | 78.3k | 27.0k | 78.3k | 8.35k | 0.1399 | 202 |
| wing (DIMACS10) | 62.0k | 122k | 62.0k | 122k | 27.9k | 0.0301 | 605 |
| olesnik0<br>(GHS_indef) | 88.3k | 342k | 88.3k | 342k | 220k | 0.1327 | 363 |
| tube1 (TKK) | 21.5k | 438k | 21.5k | 438k | 0 | 0.0000 | 102 |
| fe_tooth (DIMACS10) | 78.1k | 453k | 78.1k | 453k | 217k | 0.3673 | 286 |
| dawson5 (GHS_indef) | 51.5k | 480k | 20.2k | 211k | 19.8k | 0.0941 | 165 |

(a) Mesh-like Graphs

| Graph<br>(Collection) | Nodes | Edges | 2-core<br>Nodes | 2-core<br>Edges | Greedy<br>Cycles | Probability of<br>Selecting Greedy | Largest<br>Cycle Length |
|---|---|---|---|---|---|---|---|
| EVA (Pajek) | 8.50k | 6.71k | 314 | 492 | 84 | 0.2346 | 18 |
| bcspwr09 (HB) | 1.72k | 2.40k | 1.25k | 1.92k | 651 | 0.3276 | 54 |
| BTER1<br>$d_{avg} = 10, d_{max} = 30$<br>$cc_{max} = .3, cc_{global} = .1$ | 981 | 4.85k | 940 | 4.82k | 510 | 0.0465 | 18 |
| USpowerGrid (Pajek) | 4.94k | 6.59k | 3.35k | 5.01k | 1.68k | 0.2997 | 80 |
| email (Arenas) | 1.13k | 5.45k | 978 | 5.30k | 362 | 0.0433 | 11 |
| uk (DIMACS10) | 4.82k | 6.84k | 4.71k | 6.72k | 1.97k | 0.2488 | 211 |
| as-735 (SNAP) | 7.72k | 13.9k | 4.02k | 10.1k | 3.83k | 0.0822 | 9 |
| ca-GrQc (SNAP) | 4.16 | 13.4k | 3.41k | 12.7k | 4.43k | 0.2315 | 22 |
| BTER2<br>$d_{avg} = 10, d_{max} = 70$<br>$cc_{max} = .3, cc_{global} = .1$ | 4.86k | 25.1k | 4.54k | 24.8k | 2.69k | 0.0468 | 17 |
| gemat11 (HB) | 4.93k | 33.1k | 4.93k | 33.1k | 9.72k | 0.0011 | 42 |
| BTER3<br>$d_{avg} = 15, d_{max} = 70$<br>$cc_{max} = .6, cc_{global} = .15$ | 4.94k | 37.5k | 4.66k | 37.2k | 4.79k | 0.0518 | 18 |
| dictionary28 (Pajek) | 52.7k | 89.0k | 20.9k | 67.1k | 20.2k | 0.1410 | 36 |
| astro-ph (SNAP) | 16.7k | 121k | 11.6k | 111k | 13.2k | 0.0786 | 18 |
| cond-mat-2003<br>(Newman) | 31.2k | 125k | 25.2k | 114k | 32.5k | 0.1533 | 23 |
| BTER4<br>$d_{avg} = 15, d_{max} = 30$<br>$cc_{max} = .6, cc_{global} = .15$ | 999 | 171k | 999 | 171k | 33 | 0.0002 | 7 |
| HTC_336_4438 (IPSO) | 226k | 339k | 64.1k | 192k | 32.9k | 0.0339 | 990 |
| OPF_10000 (IPSO) | 43.9k | 212k | 42.9k | 211k | 122k | 0.3146 | 53 |
| ga2010 (DIMACS10) | 291k | 709k | 282k | 699k | 315k | 0.1466 | 941 |
| coAuthorsDBLP<br>(DIMACS10) | 299k | 978k | 255k | 934k | 297k | 0.1524 | 36 |
| citationCiteseer<br>(DIMACS10) | 268k | 1.16M | 226k | 1.11M | 150k | 0.0484 | 56 |

(b) Irregular Graphs

Appendix Table 1: Statistics of All Graphs Used in Experiments.

| Graph and Metric | Sequential Work (with Local Greedy) | 2 Thread Parallel Steps (with Local Greedy) | 8 Thread Parallel Steps (with Local Greedy) |
|---|---|---|---|
| jagmesh3 (Metric 1) | 2.73M (1.72M) | 2.02M (1.26M) | 1.07M (28.3K) |
| jagmesh3 (Metric 4) | 127K (101K) | 69.7K (51.2K) | 632K (17.4K) |
| lshp1270 (Metric 1) | 6.80M (4.35M) | 4.87M (3.50M) | 3.41M (2.29M) |
| lshp1270 (Metric 4) | 192K (150K) | 104K (85.1K) | 61.0K (41.9K) |
| rail_1357 (Metric 1) | 1.64M (1.26M) | 1.21M (909K) | 653K (550K) |
| rail_1357 (Metric 4) | 119K (114K) | 63.8K (57.0K) | 143K (143K) |
| 50 x 50 grid (Metric 1) | 9.42M (4.32M) | 9.42M (4.43M) | 3.55M (1.50M) |
| 50 x 50 grid (Metric 4) | 213K (125K) | 115K (62.5K) | 45.0K (20.0K) |
| data (Metric 1) | 17.9M (16.4M) | 13.1M (13.1M) | 8.43M (7.41M) |
| data (Metric 4) | 815K (878K) | 416K (470K) | 185K (168K) |
| 100 x 100 grid (Metric 1) | 84.0M (38.1M) | 59.7M (29.1M) | 27.2M (13.1M) |
| 100 x 100 grid (Metric 4) | 1.11M (610K) | 560K (310K) | 180K (90.0K) |
| 20 x 20 x 20 grid (Metric 1) | 64.7M (63.3M) | 45.3M (46.0M) | 30.9M (28.6M) |
| 20 x 20 x 20 grid (Metric 4) | 1.55M (1.61M) | 816K (856K) | 448K (416K) |
| L-9 (Metric 1) | 820M (382M) | 557M (266M) | 346M (124M) |
| L-9 (Metric 4) | 3.92M (2.03M) | 2.07M (1.04M) | 1.10M (396K) |
| tuma1 (Metric 1) | 597M (282M) | 362M (186M) | 147M (75.9M) |
| tuma1 (Metric 4) | 3.36M (1.69M) | 1.60M (845K) | 512K (267K) |
| barth5 (Metric 1) | 282M (149M) | 212M (119M) | 118M (54.9M) |
| barth5 (Metric 4) | 3.11M (1.98M) | 1.61M (1.03M) | 655K (312K) |
| cti (Metric 1) | 204M (195M) | 142M (143M) | 87.7M (103M) |
| cti (Metric 4) | 3.87M (3.89M) | 2.02M (2.09M) | 1.01M (1.20M) |
| aft01 (Metric 1) | 127M (118M) | 90.8M (88.3M) | 45.4M (43.5M) |
| aft01 (Metric 4) | 4.38M (4.32M) | 2.19M (2.22M) | 763k (738k) |
| 30 x 30 x 30 grid (Metric 4) | 401M (394M) | 284M (283M) | 152M (142M) |
| 30 x 30 x 30 grid (Metric 4) | 6.51M (6.62M) | 3.35M (3.40M) | 1.35M (1.27M) |
| wing (Metric 1) | 4.98B (4.16B) | 3.41B (2.99B) | 2.58B (2.08B) |
| wing (Metric 4) | 21.8M (18.8M) | 12.2M (10.8M) | 8.31M (6.70M) |
| olesnik0 (Metric 1) | 2.69B (1.71B) | 1.99B (1.36B) | 978M (630M) |
| olesnik0 (Metric 4) | 33.8M (24.6M) | 16.9M (1.28M) | 5.47M (3.62M) |
| tube1 (Metric 1) | 2.74B (N/A) | 1.98B (N/A) | 1.59B (N/A) |
| tube1 (Metric 4) | 56.1M (N/A) | 32.0M (N/A) | 22.9M (N/A) |
| fe_tooth (Metric 1) | 6.56B (5.76B) | 4.46B (4.09B) | 3.04B (2.87B) |
| fe_tooth (Metric 4) | 65.5M (62.1M) | 34.3M (32.7M) | 19.8M (18.8M) |
| dawson5 (Metric 1) | 1.49B (1.45B) | 1.06B (1.05B) | 650M (658M) |
| dawson5 (Metric 4) | 24.9M (24.7M) | 12.8M (12.8M) | 6.11M (6.19M) |

(a) Mesh-like Graphs

| Graph and Metric | Sequential Work (with Local Greedy) | 2 Thread Parallel Steps (with Local Greedy) | 8 Thread Parallel Steps (with Local Greedy) |
|---|---|---|---|
| EVA (Metric 1) | 41.4K (25.1K) | 22.5K (23.8K) | 18.5K (15.1K) |
| EVA (Metric 4) | 6.28K (4.08K) | 2.83K (3.14K) | 1.88K (1.88K) |
| bcspwr09 (Metric 1) | 308K (242K) | 199K (165K) | 130K (115K) |
| bcspwr09 (Metric 4) | 26.2K (24.9K) | 12.5K (12.5K) | 4.99K (4.99K) |
| BTER1 (Metric 1) | 2.26M (2.25M) | 1.78M (1.76M) | 1.59M (1.64M) |
| BTER1 (Metric 4) | 246K (253K) | 243K (253K) | 333K (397K) |
| USpowerGrid (Metric 1) | 1.06M (887K) | 735K (557K) | 297K (279K) |
| USpowerGrid (Metric 4) | 73.8K (77.1K) | 36.9K (33.5K) | 10.1K (10.1K) |
| email (Metric 1) | 1.22M (1.23M) | 871K (862K) | 639K (638K) |
| email (Metric 4) | 199K (204K) | 127K (127K) | 87.0K (87.0K) |
| uk (Metric 1) | 7.30M (3.88M) | 5.51M (3.29M) | 3.04M (1.62M) |
| uk (Metric 4) | 160K (108K) | 84.8K (61.2K) | 33.0K (18.8K) |
| as-735 (Metric 1) | 911K (887K) | 498K (550K) | 293K (267K) |
| as-735 (Metric 4) | 201K (201K) | 96.6K (109K) | 48.3K (44.2K) |
| ca-GrQc (Metric 1) | 2.98M (3.39M) | 1.92M (2.16M) | 923K (950K) |
| ca-GrQc (Metric 4) | 413K (509K) | 201K (242K) | 71.7K (75.1K) |
| BTER2 (Metric 1) | 11.3M (11.7M) | 8.21M (8.31M) | 6.52M (6.60M) |
| BTER2 (Metric 4) | 1.30M (1.39M) | 876K (894K) | 658K (667K) |
| gemat11 (Metric 1) | 63.7M (62.2M) | 50.4M (49.4M) | 45.7M (44.9M) |
| gemat11 (Metric 4) | 3.30M (3.22M) | 2.38M (2.33M) | 2.07M (2.04M) |
| BTER3 (Metric 1) | 18.5M (19.0M) | 14.6M (14.1M) | 13.4M (12.4M) |
| BTER3 (Metric 4) | 2.13M (2.27M) | 1.59M (1.54M) | 1.39M (1.29M) |
| dictionary28 (Metric 1) | 38.0M (33.8M) | 24.4M (24.6M) | 16.4M (15.2M) |
| dictionary28 (Metric 4) | 3.20M (3.18M) | 1.65M (1.78M) | 941K (878K) |
| astro-ph (Metric 1) | 25.3M (25.9M) | 15.7M (16.3M) | 8.63M (8.57M) |
| astro-ph (Metric 4) | 4.45M (4.74M) | 2.27M (2.43M) | 1.01M (1.01M) |
| cond-mat-2003 (Metric 1) | 38.8M (40.9M) | 27.4M (30.3M) | 21.0M (20.5M) |
| cond-mat-2003 (Metric 4) | 4.57M (5.35M) | 2.62M (3.10M) | 1.74M (1.72M) |
| BTER4 (Metric 1) | 25.4M (26.4M) | 15.1M (15.1M) | 8.53M (8.01M) |
| BTER4 (Metric 4) | 6.78M (7.06M) | 3.67M (3.67M) | 1.84M (1.73M) |
| HTC_336_4438 (Metric 1) | 14.2B (13.7B) | 8.50B (8.89B) | 4.10B (3.67B) |
| HTC_336_4438 (Metric 4) | 32.2M (32.0M) | 15.9M (16.9M) | 6.80M (6.09M) |
| OPF_10000 (Metric 1) | 47.3M (49.1M) | 31.8M (33.2M) | 16.0M (16.5M) |
| OPF_10000 (Metric 4) | 6.86M (8.66M) | 3.34M (4.29M) | 857K (1.07M) |
| ga2010 (Metric 1) | 8.77B (6.16B) | 6.88B (4.97B) | 3.21B (2.59B) |
| ga2010 (Metric 4) | 40.8M (33.5M) | 20.8M (16.9M) | 6.48M (5.35B) |
| coAuthorsDBLP (Metric 1) | 539M (552M) | 356M (371M) | 264M (237M) |
| coAuthorsDBLP (Metric 4) | 44.4M (51.3M) | 23.5M (26.3M) | 15.3M (13.8M) |
| citationCiteseer (Metric 1) | 1.08B (1.07B) | 716M (723M) | 506M (482M) |
| citationCiteseer (Metric 4) | 74.4M (76.2M) | 40.7M (41.8M) | 25.3M (24.2M) |

(b) Irregular Graphs

Appendix Table 2: Condensed Results of Scaling Experiments.

# Parameter Tuning of PI-controller with Bat Algorithm

Dušan Fister
University of Maribor, Faculty of Mechanical Engineering
Smetanova 17, 2000 Maribor
E-mail: dusan.fister@student.um.si

Riko Šafarič, Iztok Jr. Fister and Iztok Fister
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor

**Keywords:** PI-controller, nature-inspired algorithms, optimization

**Received:** January 6, 2016

*Correct input controller parameter settings are vital and in constant connection with output functions - e.g. robotic positioning. Optimal positioning of robotic arm automatically provides a high level of safety and functionality. The first prevents robot from hurting any people around or even itself, while the second ensures robot advantage. In order to improve both safety and functionality, we propose two nature-inspired algorithms for parameter tuning of PI-controller and test them on the laboratory robotic manipulator. However the manipulator is not designed to perform a real robotic work, it offers a detailed approach of positioning control. Our goal is to access the positioning control unit and combinatorially set the input controller parameters with the help of two implemented algorithms. This principle is called automatic parameter tuning, which firstly tests the corresponding setting, then evaluates it and finally tries to improve former result with new one.*

*Povzetek: Za natančno pozicioniranje zahteva robotski regulator pravilno nastavljene parametre. Ti zagotavljajo varno in funkcionalno delovanje robota. S testiranji, opisanimi v nadaljevanju, želimo določiti optimalne konstante regulatorja z algoritmom za nastavljanje parametrov, ki sloni na nelinearnem, dvoosnem robotskem mehanizmu. Implementirana optimizacijska algoritma temeljita na vzorih iz narave in določata parametre avtomatsko, brez človeške interakcije. Naš pristop je iterativen, kar pomeni, da se želimo s kombinatoričnim ugotavljanjem čimbolj približati idealni rešitvi, ki pa je sicer ne moremo doseči. Avtomatski postopek nastavljanja parametrov z optimizacijskim algoritmom predstavlja odskočno desko za zagotavljanje varnosti ter funkcionalnosti, povečanega obsega dela robota ter višje natančnosti in kakovosti izdelkov.*

## 1 Introduction

A robot is typically an electro-mechanical device controlled by a computer program. It operates in an environment which can be changed using actions for whether it is delegated. The robot performs repeated actions that were before executed by humans. They have been displaced and upgraded by robots especially by performing dangerous tasks, e.g., coating cars in the automotive industry, aeronautics, etc.

A robotic arm, for instance, is moved and positioned using a closed-control loop. The control loop consists of a controller and a control plant. The controller is a part of the electrical scheme, which controls a mechanical part of the robotic arm (control plant). The control plant consists of electrical motors to lift and lower the arm. However, this process is subject of gravity forces.

Typically, the control loop is implemented by so called PID-controllers in the real-world applications (Fig. 1). This controller calculates an error value $e$ between desired input value $y_{des}$ and actually measured output $y_{act}$. Then, a derivative and integral of the error signal is calculated. Actually, the output signal $u$ is obtained as follows

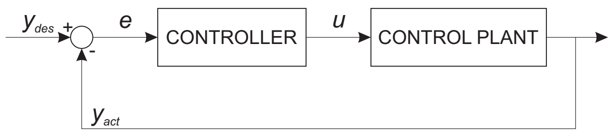


Figure 1: Scheme of a robot.

$$u = K_P(y_{des} - y_{act}) + K_I \int (y_{des} - y_{act}) + K_D(y_{des} - y_{act})'. \quad (1)$$

Eq. (1) consists of three parts, i.e., proportional, integral and derivative terms weighted by corresponding proportional gain $K_P$, integral gain $K_I$ and derivative gain $K_D$, respectively. The control signal $u$ is sent to the control plant in order to obtain the new output $y$ that serves as the new $y_{act}$ value for generating the new error signal $e$. This process continues until equilibrium is achieved. Let us notice that the desired values are input variables that represent a reference generator output, while the desired values are obtained as a feedback of the control plant on the input variables. There are more types of the reference generators, e.g., micro-controllers, DSP, FPGA, etc. A discrete equation of PID-controller can be written (Eq. 2).

$$\begin{aligned} u(k) =& u(k-1) + q_0 \cdot e(k) + \\ & q_1 \cdot e(k-1) + q_2 \cdot e(k-2), \end{aligned} \tag{2}$$

where

$$q_0 = K_P \cdot (1 + K_I + K_D), \tag{3}$$

$$q_1 = -K_P \cdot (1 + 2 \cdot K_D - K_I) \; and \tag{4}$$

$$q_2 = K_P \cdot K_D. \tag{5}$$

Only PI-controller type is used for our application. In line with this, $K_D$ gain should be set to zero. Eq. 3-5 can be then simplified to new form (Eq. 6-8):

$$q_0 = K_P \cdot (1 + K_I), \tag{6}$$

$$q_1 = -K_P \cdot (1 - K_I) \; and \tag{7}$$

$$q_2 = 0. \tag{8}$$

In the next chapters, parameters $q_0$ and $q_1$ of PI-controller will be optimized.

There are few strategies for parameter tuning of robotic controller. Using Bode plotting [16] and root locus method [4] a linear controller can be tuned. For non-linear control plants, an iterative approach of tuning should be employed. In this method, random parameters are entered into the robotic controller and according to mechanic response of manipulator little corrections are made through more iterations. A new set is then entered into controller and so on. The basic and the simplest strategy of tuning is a manual approach. Requires an experienced and patient engineer, what makes this strategy time-consuming. It can be automated using the micro-controller, which makes the process faster up to few times, but then an optimization algorithm is required. Every algorithm is intended to find an optimal solution of the problem, so the question is, how fast can algorithm approach to an ideal solution? Today, many algorithms are widely-known. They differ to each other by the ease of use, complexity, principle of working and most importantly, convergence speed. The last parameter could be simply compared to algorithm's efficiency. The fact is, faster than the algorithm is searching, more local is the search space becoming and slower it is searching, more global it can go.

In previous century a greater demand of quality, quantity and efficiency of making products was sensed. As a result, an optimization has been became more and more important. Using computer guided optimization, controlling machines have became easier and even more precisely to use. Optimization algorithms are today frequently and widely used in order to maximize quality and quantity of products, to minimize the production costs as well as increase the functionality, safety and duration of services. For solving the real-world problems, where the domain-specific knowledge is absent, a general problem solvers have been emerged. Today, evolutionary and swarm intelligence algorithms act increasingly in that role.

Basic principles of evolutionary algorithms (EA) were discovered a bit longer ago. In 1871, Charles Darwin published an article about natural selection [2]. Alan Turing was the first who successfully implemented the algorithm [20], that based on results of Charles Darwin's work. He implemented an optimizational algorithm, named *genetic search*, and has later also strived for other topics of artificial intelligence. His work was upgraded by John Holland in 1988, who created a *genetic algorithm* (GA) that is today one of the most often worldwide used evolutionary algorithm [13].

On the other hand, a new way of optimization was being commenced in 1995, called the Swarm Intelligence (SI). Particle Swarm Algorithm (PSO) invented by Russell Eberhart and James Kennedy [3] became quickly widely used. Interestingly, the SI-based algorithms use the biological and social relations, since individuals collaborate between each other by learning of experiences. Many SI-based algorithms are known, e.g. ant colony, bee colony, bird flocks, bats, cuckoo search, termites and fish schools. The Bat algorithm (BA) is one of the newest, since it was proposed in 2010 by Yang. The BA quickly widened for testing purposes on various applications [23]. Firstly on numeric and discrete applications, after that also for multi-objective optimization [24]. The possibility of constrained optimization was proved by Gandomi et. al. [10]. The BA was hybridized with Differential Evolution strategies in 2013 by Fister et. al. [9] as well as by Random Forest Regression method [8]. The self-adaptation was proposed by Fister et. al. [5] and [6], which presents one of the most successful BA variants.

The remainder of the paper is divided into next sections: Section 2 presents a control plant and controller. Section 3 describes both nature-inspired algorithms used in this study, i.e., GA and BA.

## 2 Control plant

In this study, the 2 degree-of-freedom (2 DOF) Selective Compliance Assembly Robot Arm (SCARA) [18] depicted in Fig. 2 was taken into account. The robot arm is controlled in a 2-dimensional space and parameters of this controller are tuned by an optimization algorithm. Since 2005, four different optimization methods were proposed for parameter tuning of the same robot. Albin Jagarinec developed an adaptive regulator in 2005 [15], while Marko Kolar the fuzzy controller in the next year [17]. A neural sliding-mode controller was implemented on the system in the same year by Jure Čas [21]. Finally, the genetic algorithm was tested by Tomaž Slanič [19].

The control plant is responsible for moving the robotic arm that is enabled by two direct-current ESCAP 28 D2R

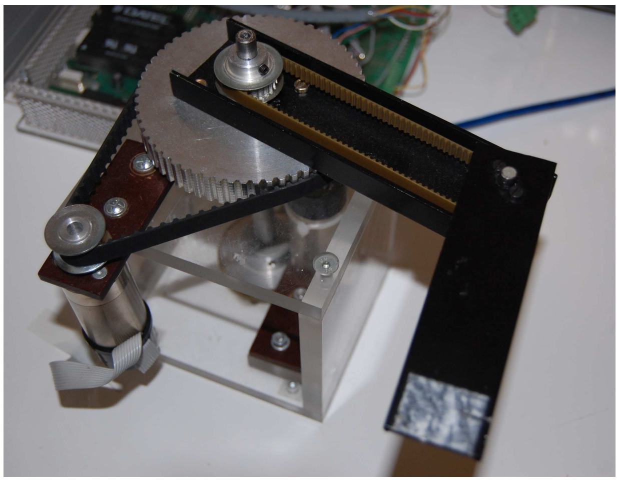


Figure 2: 2-DOF robot.

11 motors and the appropriate power electronics. Opposite to power electronics, two incremental decoders are connected, which transform an incremental encoder's signals from both motors to angles of rotation. These are both processed in a custom input/output interface card based on a digital signal processor (DSP-2 Roby) [22]. Besides decoding signals, the DSP-2 Roby serves also as motor's driver. In addition, the DSP-2 Roby retrieves other basic information, like angles of rotation and time to personal computer, which plots the step responses of the robot and outputs usable information for evaluation of robot arm behavior. Moreover, the optimization algorithm is also being run on this processor.

## 2.1 Robot's model

The robot's model can be written using Eq. (9) basing on the principles of Lagrangian mechanics, as follows

$$\begin{bmatrix} J_{m1}N_1 + \frac{a_1+a_2\cos(q_2)}{N_1} & \frac{a_3+a_2\cos(q_2)}{N_1} \\ \frac{a_3+a_2\cos(q_2)}{N_2} & J_{m2}N_2 + \frac{a_3+J_{3o}}{N_2} \end{bmatrix} \cdot \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} + \begin{bmatrix} \frac{-a_2\dot{q_2}(2\dot{q_1}+\dot{q_2})+\sin(q_2)}{N_1} \\ \frac{a_2\sin(q_2)\dot{q_1}^2}{N_2} \end{bmatrix} = \begin{bmatrix} \tau_{mot1} \\ \tau_{mot2} \end{bmatrix}, \tag{9}$$

where $J$ means moment of inertia, $l$ length of handles and $m$ mass of handles with gears. Parameters $q_i$, $\dot{q}_i$ and $\ddot{q}_i$ mean position (angle of rotation), velocity and acceleration of specific axis [21]. Eq. 10 presents the meaning of parameters $a_1$, $a_2$ and $a_3$.

$$\begin{aligned} a_1 &= I_1 + I_2 + I_4 + m_2 \cdot l_{1T}^2 + (m_3 + m_4) \cdot l_1^2 + m_4 \cdot l_{2T}^2 \\ a_2 &= m_4 \cdot l_1 \cdot l_{2T} \\ a_3 &= I_4 + m_4 \cdot l_{2T}^2 \end{aligned} \tag{10}$$

As seen, equation is a two dimensional, which tends the control plant of robot as non-linear. The equation presents the motor's torque, necessary for correct motion of robot's peak. The full elaboration of this equation is presented in [21]. Note that only proportional and integral gains (PI-controller) were used in our study.

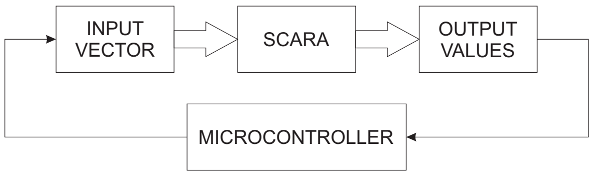


Figure 3: Searching for the optimal parameter setting of the PI-controller.

## 2.2 Optimization problem

In general, the optimization problem is defined as a quadruple $OP = \langle I, S, f, goal \rangle$ [11], where $I$ presents a set of instances that can be arisen on the input, $S$ is a set of feasible solutions, $f$ objective function and *goal* denotes if the minimum or maximum of the objective function is searched for. The input vector is expressed as

$$\mathbf{x} = \{q_{0,1}, q_{1,1}, q_{0,2}, q_{1,2}\}, \tag{11}$$

where $q_{0,1}$ and $q_{1,1}$ mean the controller input parameters for the first axis and $q_{0,2}$ and $q_{1,2}$ for the second axis of a robotic manipulator. The task of the optimization is to maximize the fitness function, i.e., $max(f_i)$. The fitness function evaluates three different measures obtained as a feedback **y** from the control plant consisting of:

– $Over_i$ - actual overshoot,

– $Ess_i$ - actual steady state error and

– $Time_i$ - actual settling time.

According to mentioned measures, the fitness function is expressed as follows:

$$\begin{aligned} f(\mathbf{y}) = \sum_{i=1}^{2} \frac{1}{2}(E_{1i}(1 - |P_i - Over_i|) + \\ E_{2i}(1 - Time_i) + E_{3i}(1 - Ess_i)), \end{aligned} \tag{12}$$

where $E_{ij}$ are initialized constants representing weights that determine an influence of the specific outputs for each axis in Eq. 12. Obviously, the sum of these three constants of specific axis is:

$$\sum_{i=1}^{3} E_{ij} = 1, \tag{13}$$

where $i$ is the specific output variable and $j$ the specific axis.

The optimization problem can now be defined as searching the best input values in order to obtain the best output values estimated by the fitness function. The process is graphically presented in Fig. 3, from which it can be seen that the optimization algorithm puts the input vector **x** to SCARA robot arm controller that moves and its position is then being measured. The output vector **y** is obtained after the moving and positioning the arm. The value of fitness function is determined from this vector.

## 3 Algorithms for parameter tuning of PI-controller

The majority of real-world problems with which human are confronted today are NP-hard. This means that the time complexity of solving these problems increases by increasing a problem size. The problem size is estimated by the number of input variables. As a results, when the number of variables increased to some high value, the user can wait for the results indefinitely. Therefore, engineers responsible for solving these in practice are not interested for their optimal solutions, but they are satisfied with an approximate optimal solutions obtained in real-time as well. Consequently, a lot of heuristic algorithms have been emerged that are able to obtain the non-optimal solutions, but well enough for practical applications.

The stochastic nature-inspired population-based algorithms are heuristic methods that can be applied to problem domains, where no domain-specific knowledge has yet been discovered. In this study, we focus on searching for an optimal parameter tuning of PI-controller with EA and SI-based algorithms. Precisely, a comparative study of the bat algorithm (BA) and genetic algorithm (GA) for solving this problem has been performed, where the former belongs to a class of SI-based algorithms, while the latter is the member of EA-family.

In the remainder of the paper, characteristics of both algorithms are illustrated in details.

## 3.1 Bat algorithm

As already mentioned, BA is one of the newest representatives of SI-based algorithms. Since 2010, its reputation and visibility are highly rising. Bat algorithm is easy to implement and applicable to various applications. It offers solid results by solving of the low-dimensional problems and that is one of the reasons to be applied to tune the parameters of PI-controller. Thus, high convergence of the BA algorithm is expected.

### 3.1.1 Fundamentals of Bat algorithm

Bats are night animals. Nature has given them ability to navigate in darkness, using a so-called sonar, named *echolocation*. This phenomenon consists of generating an ultrasonic pulse, which echoes from obstacles and prey, bouncing back to the bat, who calculates the distance to either obstacle or prey. More information on bats behavior and their abilities can be found in [14].

The BA algorithm treats bats as a swarm of bats, searching for a prey. Since bats search for the prey individually, BA emphasizes the phenomena of echolocation by converging the whole swarm by approaching the found prey. This means that one random individual can achieve the whole swarm to divert for food. From the engineer's point of view, more food means higher fitness function and better solution of the problem. The whole swarm is converging to the best solution during generations by changing their current positions.

### 3.1.2 Model of Bat algorithm

The moving of bats, their attitude and acting in a swarm presented in previous section can be modelled using simple mathematical equations. The whole modelling process is described in [23], so only main results are shown here. At first, three different variables should be defined describing bat moving as follows:

- frequency of pulse $Q_i^{(t)}$,
- velocity $\mathbf{v}_i^{(t)}$ and
- position $\mathbf{x}_i^{(t+1)}$,

where $Q_i^{(t)}$ represents actual pulse frequency, $\mathbf{v}_i^{(t)}$ velocity of an individual bat and $\mathbf{x}_i^{(t+1)}$ position of the $i$-th bat at generation $t$.

The flight of a bat can be summarized in Eqs. 14-16:

$$Q_i^{(t)} = Q_{min}^{(t)} + (Q_{max}^{(t)} - Q_{min}^{(t)}) \cdot \beta, \tag{14}$$

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t)} + \left[\mathbf{x}_i^{(t)} - \mathbf{x}_{best}^{(t)}\right] \cdot Q_i, \tag{15}$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t)}. \tag{16}$$

Output pulse frequency can vary in the interval $Q_i^{(t)} \in [Q_{min}, Q_{max}]$. The random number $\beta \in [0, 1]$ specifies the output pulse and $\mathbf{x}_{best}^{(t)}$ presents the current best solution.

The BA search process consists of two components, i.e., exploration and exploitation. Exploration means a discovering of the new solutions, while the exploitation directs the search in the neighborhood of the existing solutions. Both processes cannot be run simultaneously because they typically depend on the variation operators, while balancing between exploration and exploitation performs a control parameter setting. There are more optimal parameter settings.

In the BA, the exploration and exploitation components of the search process are balanced by using two exploration strategies and parameter $r_i^{(t)}$. The first exploration strategy expressed by Eq. 16 is more explorative in its nature, while the second strategy expressed as

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \epsilon \cdot \bar{A}^{(t)}, \tag{17}$$

implements the random walk, i.e., a kind of the local search that is more focused on the exploitation of the current best solution. Let us notice that $\mathbf{x}_{new}$ in the equation presents new best solution, if applicable, and $\mathbf{x}_{old}$ presents current best solution. $\epsilon$ is the random number in range (-1,1) and $\bar{A}^{(t)}$ is the average loudness.

The last strategy is applied according to the pulse rate $r_i^{(t)}$.The pulse rate is normally being changed during generations, where simulates nature behavior of bats outputting loud pulses with low pulse rate when searching for preys and outputting silent pulses with high pulse rate when approaching to the prey.

### 3.1.3 Pseudocode of Bat algorithm

A pseudo-code of the BA algorithm is illustrated in Algorithm 1. This algorithm consists of the following elements:

- initialization of bat population (function 'init_bat' in line 1),
- generation of new solution according to Eq. 16 (function 'generate_new_solution' in line 6),
- the local search step according to Eq. 17 and parameter $r_i^{(t)}$ (function 'improve_the_best_solution' in lines 7-9),
- evaluation of the new solution (function 'evaluate_the_new_solution' in line 10),
- save the best solution conditionally (in lines 12-15),
- find the best solution (in line 15).

BA is a population algorithm, what means that a population size and maximal number of generations should be pre-defined. During the optimizational process, the new position is being calculated for every bat and generation value is incremented. The execution of algorithm stops when maximal number of generations are reached.

**Algorithm 1** Original Bat algorithm.

**Input:** Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1 \ldots Np$, $MAX\_FE$.
**Output:** The best solution $\mathbf{x}_{best}$ and its corresponding value $f_{max} = \max(f(\mathbf{x}))$.

```
1: init_bat();
2: $eval$ = evaluate_the_new_population;
3: $f_{max}$ = find_the_best_solution($\mathbf{x}_{best}$);
4: while termination_condition_not_met do
5:     for $i = 1$ to $Np$ do
6:         $\mathbf{y}$ = generate_new_solution($\mathbf{x}_i$);
7:         if $\mathrm{rand}(0,1) > r_i$ then
8:             $\mathbf{y}$ = improve_the_best_solution($\mathbf{x}_{best}$)
9:         end if{local search step}
10:        $f_{new}$ = evaluate_the_new_solution($\mathbf{y}$);
11:        $eval = eval + 1$;
12:        if $f_{new} \geq f_i$ and $\mathrm{N}(0,1) < A_i$ then
13:            $\mathbf{x}_i = \mathbf{y}$; $f_i = f_{new}$;
14:        end if{save the best solution conditionally}
15:        $f_{max}$=find_the_best_solution($\mathbf{x}_{best}$);
16:    end for
17: end while
```

## 3.2 Genetic algorithm

As already mentioned, GA was one of the first optimization algorithms, belonging to a family of EA [1]. Although GA is similar as BA a population-based algorithm, it differs in comparison to BA a lot.

### 3.2.1 Fundamentals of Genetic algorithm

GA searches for the global optimum using unique genetic operators. There are three common operators, i.e., a selection, a crossover and a mutation. The parent's selection is a part of algorithm, where so called parent genomes (solution of the problem) are chosen to enter into crossover procedure. The two common parent selections are roulette-wheel selection and tournament selection [12]. Crossover is the most important part of the algorithm, since it enables the two parents to vary their genetic material in order to improve the existing solutions. The last operator, mutation could be represented as a rescue method, which prevents algorithm from trapping in local optima, which usually stops improving process. A task of the survivor selection is to determine the better solutions for surviving and transferring their good characteristics in the next generations.

Nevertheless, the basic GA consists of six elements:

- initialization of individuals,
- parent's selection,
- crossover,
- mutation,
- evaluation and
- survivor's selection.

Initialization is a process, where random (starting) population is generated and evaluation is process where the fitness value determining success of each individual is calculated. The optimization runs until the number of generations has not reached its maximum specified value, or its optimal value was found or enough quality solution was discovered. The most important issue by using the EA is the representation of individuals. Although the original GA uses binary representation, where solutions are represented as binary strings, the real-coded GA is becoming more important today. This type of GA is applied also in this study.

### 3.2.2 Pseudocode of Genetic algorithm

The pseudo-code of the GA is presented in Algorithm 2, from which it can be seen the following elements of the GA:

- initialization of initial population (function 'init_population_with_random_candidate_solutions' in line 1),
- parent's selection (function 'select_parents' in line 4),
- crossover operator (function 'recombine_pairs_of_parents' in line 5),
- mutation operator (function 'mutate_the_resulting_offspring' in line 6),
- evaluation function (function 'evaluate_new_candidates' in line 7),

– survivor's selection (function 'select_individuals_for_the_next_generation' in line 8).

The main evolutionary cycle is performed until the 'termination_condition_not_met' is not met. Each solution is represented as real-coded vector $\mathbf{x}_i^{(t)} = \{x_{i,j}^{(t)}\}$, where $i = 1, \ldots, NP \wedge j = 1, \ldots, D$, $NP$ is a population and $D$ a dimension of the problem.

**Algorithm 2** Evolutionary algorithm.

```
1: init_population_with_random_candidate_solutions;
2: eval = evaluate_each_candidate;
3: while termination_condition_not_met do
4:     select_parents;
5:     recombine_pairs_of_parents;
6:     mutate_the_resulting_offspring;
7:     eval += evaluate_new_candidates;
8:     select_individuals_for_the_next_generation;
9: end while
```

### 3.3 Parameter tuning of PI-controller with nature-inspired algorithms

In order to solve parameter tuning of PI-controller, the nature-inspired algorithms need to be modified properly. As we know, the results of our optimization problem depend on the input vector $\mathbf{x}_i$ according to Eq. 11. This means that only a representation of solution for the nature-inspired algorithms must be changed in order to adapt them for solving the parameter setting of PI-controller. In other words, the solution in these algorithms is represented as follows

$$\mathbf{x}_i^{(t)} = \{x_{i,j}\}, \text{ for } i = 1, \ldots, NP \wedge j = 1, \ldots, D, \qquad (18)$$

where $x_{i,1} = q_{0,1}$, $x_{i,2} = q_{1,1}$, $x_{i,3} = q_{0,2}$ and $x_{i,4} = q_{1,2}$, and $NP$ denotes a population size and $D$ is a dimension of the problem. All the other elements of the original nature-inspired algorithms do not demand any modifications.

## 4 Results

The goal of our experimental work was to show that the stochastic nature-inspired algorithms can successfully be applied for searching the optimal parameters of PI-controller that controls the robotic arm SCARA. In line with this, two population-based algorithms were developed and customized to this problem, i.e., BA and GA. The development was performed on a personal computer (PC) with installed Windows operating system and MATLAB/Simulink in language C/C++. The algorithms were loaded onto DSP2-Roby interface card, where these were also executed, while the results were received from the card via USB connection to the PC and displayed in MATLAB/Simulink.

The algorithm's parameters as presented in Table 1 were used during the simulation.

| Parameter | Setting |
|---|---|
| $NP$ | 10 |
| $n_{gen}$ | 10 |
| $Q$ | [0.5,1.5] |
| $\beta$ | [0,1] |
| $r_i$ | 0.1 |
| $A_i$ | 0.9 |

(a) BA

| Parameter | Setting |
|---|---|
| $NP$ | 10 |
| $n_{gen}$ | 10 |
| $p_c$ | 0.8 |
| $p_m$ | 0.01 |
| Par.sel. | Tour.$m = 2$ |
| Sur.sel. | Fittest |

(b) GA

Table 1: Parameter setting by nature-inspired algorithms.

Let us notice that both algorithms used the same number of fitness function evaluations, i.e., $MAX\_FE = 10 \times 10 = 100$, where $MAX\_FE = NP \times n_{gen}$. This number of fitness function evaluations is relatively low, but this setting enables algorithms to run in the real-time. In order to limit the search space rationally, lower and upper bounds as presented in Table 2 were considered during the optimization for both algorithms.

| Parameter | Bounds | |
|---|---|---|
| | Lower | Upper |
| $q_{0,1}$ | 0 | 400 |
| $q_{1,1}$ | 0 | 40 |
| $q_{0,2}$ | 0 | 400 |
| $q_{1,2}$ | 0 | 40 |

Table 2: Limited values of parameters.

In the table, parameters $q_{0,1}$ and $q_{1,1}$ denote limited parameter values for axis 1, while $q_{0,2}$ and $q_{1,2}$ limited values for axis 2.

The results of the optimization using algorithms BA and GA are illustrated in Table 3, from which it can be observed that 10 independent runs were conducted for each of the algorithm in test. This is normally, when dealing with stochastic algorithms, where we are usually not interested for the best solution, but rather the average results of the optimization. However, in our case, we made an exception here and considered the best solutions. The obtained results for each of two axis are presented together with the corresponding average values and the average values according to all runs are presented. The best results are marked in the table as bold.

As can be seen from the table, the best result of the BA was obtained in seventh run, while the GA was the best in the third run. When comparing the results of both algorithms with each other, it can be observed that the result achieved by GA was slightly better than this achieved by the BA. However, when these results were estimated statistically using the Wilxocon signed rank non-parametric test with confidence $\alpha = 0.05$, it turned out that the BA outperformed the results of GA significantly (p-value$= 0.03 < 0.05$).

| Run | BA | | | GA | | |
|---|---|---|---|---|---|---|
| | Axis-1 | Axis-2 | Average | Axis-1 | Axis-2 | Average |
| 1 | 0.9729 | 0.9726 | 0.9727 | 0.9804 | 0.9600 | 0.9702 |
| 2 | 0.9795 | 0.9497 | 0.9646 | 0.9729 | 0.9556 | 0.9642 |
| 3 | 0.9702 | 0.9847 | 0.9775 | 0.9726 | 0.9845 | **0.9786** |
| 4 | 0.9757 | 0.9802 | 0.9780 | 0.9455 | 0.9653 | 0.9554 |
| 5 | 0.9778 | 0.9592 | 0.9685 | 0.9638 | 0.9443 | 0.9540 |
| 6 | 0.9724 | 0.9790 | 0.9757 | 0.9782 | 0.9640 | 0.9711 |
| 7 | 0.9746 | 0.9810 | **0.9778** | 0.9752 | 0.9727 | 0.9740 |
| 8 | 0.9647 | 0.9887 | 0.9767 | 0.9774 | 0.9409 | 0.9592 |
| 9 | 0.9787 | 0.9733 | 0.9760 | 0.9721 | 0.9569 | 0.9645 |
| 10 | 0.9748 | 0.9430 | 0.9589 | 0.9662 | 0.9568 | 0.9615 |
| Average | 0.9741 | 0.9712 | 0.9726 | 0.9704 | 0.9601 | 0.9653 |

Table 3: Comparing the results of BA and GA.

The corresponding best results as obtained by BA and GA algorithms are presented in Table 4.

| Alg. | Axis-1 | | Axis-2 | | Eff. |
|---|---|---|---|---|---|
| | $q_{0,1}$ | $q_{1,1}$ | $q_{0,2}$ | $q_{1,2}$ | |
| BA | 257.064 | 18.9201 | 163.155 | 15.1531 | 0.9778 |
| GA | 56.4213 | 3.27043 | 108.571 | 7.68699 | 0.9786 |

Table 4: The best results obtained by BA and GA.

As can be observed from the table, there is not only one optimal solution, because both sets of input parameters were very different when compared between each other. However, it seems that more important is a relation between both pairs of input parameters.

In order to show how the optimal values are changed during the optimization, the convergence graphs in Fig. 4 are drawn that depict a changing of the best solution during one run according to increasing of generations for each of the observed algorithm.

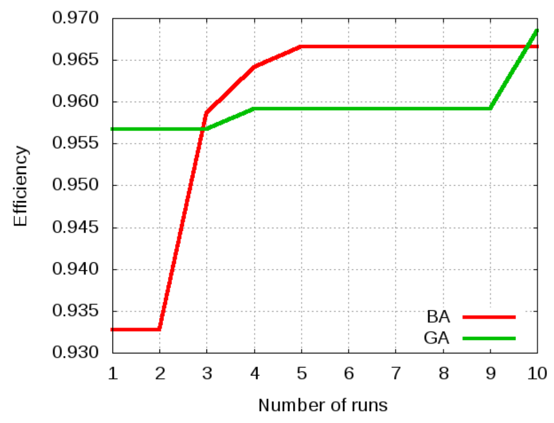


Figure 4: Convergence graph by BA and GA.

As can be seen from the graph, the BA showed a rapid convergence to the optimal value, which is already found in fifth generation. After this generation the algorithm shows signs of stagnation. On the other hand, the GA converge to the optimal value slower. Therefore, it can improve the results also in later generations.

## 5 Conclusion

The aim of our experimental work was to show that stochastic nature-inspired population-based algorithms can successfully be applied to tuning parameters of PI-controller of the robot arm SCARA. Two nature inspired algorithms were taken into consideration, i.e., the BA and GA algorithms. The former is simple and easy to implement and because of its rapid convergence interesting to use in robotics.

The results of experiments showed that both the algorithms can be used for optimal tuning parameters of PI-controller. Although the BA algorithm significantly outperformed the results of GA according to the best obtained results for each of two axis in ten runs, it turned out that the GA converges slower than the BA. This means that the GA demands the larger population as well as higher number of generations. On the other hand, increasing the population size and/or the maximum number of generations causes increasing the optimization time that can dramatically affect the real-time response of the system. In this context, the BA algorithm is more appropriate for this optimization as GA.

In the future, the BA algorithm could be hybridized with differential evolution mutation strategies [9] and thus the results of optimization would be improved. An adaptation of control parameters represents additional method, where these are encoded into representation of solutions and undergo acting the variation operators [7].

## References

[1] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] C. Darwin. R.(1859): On the origin of species by means of natural selection. *Murray. London*, 1871.

[3] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[4] W. R. Evans. Control system synthesis by root locus method. *American Institute of Electrical Engineers, Transactions of the*, 69(1):66–69, 1950.

[5] I. Fister, S. Fong, J. Brest, and I. Fister. A novel hybrid self-adaptive bat algorithm. *The Scientific World Journal*, 2014, 2014.

[6] I. Fister, S. Fong, J. Brest, and I. Fister. Towards the self-adaptation in the bat algorithm. In *Proceedings of the 13th IASTED international conference on artificial intelligence and applications*, 2014.

[7] I. Fister, D. Strnad, X.-S. Yang, and I. Fister Jr. Adaptation and hybridization in nature-inspired algorithms. In *Adaptation and Hybridization in Computational Intelligence*, pages 3–50. Springer, 2015.

[8] I. Fister Jr, D. Fister, and I. Fister. Differential evolution strategies with random forest regression in the bat algorithm. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1703–1706. ACM, 2013.

[9] I. Fister Jr, D. Fister, and X.-S. Yang. A hybrid bat algorithm. *Elektrotehniški vestnik*, 2013.

[10] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari. Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, 22(6):1239–1255, 2013.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[12] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

[13] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[14] D. R. Griffin. Listening in the dark: the acoustic orientation of bats and men. 1958.

[15] A. Jagarinec. *Adaptivni regulator z mehko logiko za dvoosni SCARA mehanizem*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[16] L. H. Keel and S. P. Bhattacharyya. A bode plot characterization of all stabilizing controllers. *Automatic Control, IEEE Transactions on*, 55(11):2650–2654, 2010.

[17] M. Kolar. *Vodenje SCARA robota z mehko logiko*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2005.

[18] H. Makino, N. Furuya, K. Soma, and E. Chin. Research and development of the scara robot. In *Proceedings of the 4th International Conference on Production Engineering*, pages 885–890, 1980.

[19] T. Slanič. *Genetski regulator za dvoosnega SCARA robota*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2006.

[20] A. M. Turing. Intelligent machinery, a heretical theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, page 105, 1948.

[21] J. Čas. *Izdelava zveznega nevronskega sliding-mode regulatorja za teleoperiranje SCARA robota*. Diplomsko delo : Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2006.

[22] M. Čurkovič. *Vgrajeni sistemi DSP/FPGA v sistemih vodenja*. Magistrsko delo Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2010.

[23] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[24] X.-S. Yang. Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3(5):267–274, 2011.

# PCARD Platform for mHealth Monitoring

Matjaž Depolli, Viktor Avbelj and Roman Trobec
Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana
E-mail: matjaz.depolli@ijs.si, viktor.avbelj@ijs.si, roman.trobec@ijs.si

Jurij Matija Kališnik
Department of Cardiovascular Surgery, University Medical Center Ljubljana, Zaloška Cesta 2, 1000 Ljubljana
E-mail: jurij-matija.kalisnik@mf.uni-lj.si

Korošec Tadej
Društvo distrofikov Slovenije, Linhartova 1/III, p.p. 2618, 1001 Ljubljana
E-mail: info@drustvo-distrofikov.si

Antonija Poplas Susič
Community Health Centre Ljubljana, Metelkova ulica 9, 1000 Ljubljana
E-mail: antonija.poplas-susic@zd-lj.si

Uroš Stanič
Kosezi d.o.o., Cesta na Laze 7, 1000 Ljubljana
E-mail: uros.j.stanic@gmail.com

Aleš Semeja
Terme Dobrna d.d, Dobrna 50, 3204 Dobrna, Slovenia
E-mail: ales.semeja@terme-dobrna.si

**Keywords:** mHealth, cloud computing, ECG, pilot study

**Received:** December 11, 2015

*The introduction of information and communication technologies (ICT) into the integrated healthcare system could increase the self-management of health and therefore increase the efficacy and decrease the costs of overall health management. A personal mobile health monitoring system (PCARD) has been developed, which uses moderately-priced and user-friendly technological solutions, e.g. wireless body sensors for data acquisition, advanced algorithms for data analysis, widely available smart phones for visualization of measurements, and the existing communication infrastructure for data transfer. The solution is unobtrusive, works with existing devices, and provides useful information to both direct users and to the health care system. The PCARD system starts with measurement of ECG signal that incorporates significant information about the global health state. It then continues with display of the signal and its analysis on a personal terminal, such as smartphone and on Cloud-based storage, processing, and visualization software. Four pilot studies have been designed to validate to which extent the continuous measured ECG data could contribute to improved quality and efficiency of the healthcare. Also, the level of safety and reliability, the acceptance from users, and the potential for commercialization will be validated in the scope of the pilots.*

*Povzetek: Uvedba informacijskih in komunikacijskih tehnologij (IKT) v celostno zdravstveno oskrbo lahko poveča sposobnost samoupravljanja zdravja s čimer poveča učinkovitost ter zmanjša stroške zdravstvene oskrbe. Razvit je bil osebni mobilni sistem spremljanja zdravja (PCARD), ki privablja z zmerno ceno in uporabniku prijaznimi tehnološkimi rešitvami, kot so brezžične telesni senzorji za zajemanje podatkov, napredni algoritmi za analizo podatkov, vizualizacija meritev na široko dostopnih pametnih telefonih in uporaba obstoječe komunikacijske infrastrukture za prenos podatkov. Rešitev je nevsiljiva, deluje z obstoječimi napravami, ter nudi koristne informacije tako za neposredne uporabnike kot tudi za sistem zdravstvenega varstva. Sistem PCARD temelji na merjenju EKG-signala, ki vključuje pomembne informacije o svetovnem zdravstvenem stanju. Nato nadaljuje s prikazom signala in njegovo analizo na osebnem terminalu, ter končuje s skladiščenjem, obdelavo in vizualizacijo v oblaku. Štiri pilotne študije so bile zasnovane, da bi preverili, v kolikšni meri bi lahko neprestano merjenje EKG prispevajo k izboljšanju kakovosti in učinkovitosti zdravstvenega varstva. Poleg tega bodo študije prispevale tudi k večji stopnji varnosti in zanesljivosti, analizi nivoja uporabniškega sprejemanja mobilnega EKG merilnika, in potrditvi potenciala za komercializacijo PCARD sistema.*

## 1 Introduction

Future mHealth solutions based on wearables for monitoring ECG, vital signs, and activity of subjects will provide an important means of healthcare. Although such wearables are also usable in clinical environments, they are most efficient when used in subjects' everyday activities. As such, they are perfect for discovering arrhythmias, measuring the impact of drugs on arrhythmias, documenting ischemia, following up on the adherence to drug therapies, checking up the results of ablation procedure, evaluating syncope and light-headedness [1], etc. Aided by the computer analysis of the provided rich set of measured data, the mHealth based monitoring could also tackle a large set of comorbidities (e.g. diabetes, cardio-oncology, cerebrovascular disease, and other neurological disorders affecting patient's mobility).

The PCARD mHealth solution could be applied in medicine (e.g. follow-ups of patients in tertiary level, or motoring of patients with palpitations in primary level), in wellness and health centers, at home for personal use, in protection of professionals during stressful and physically intensive tasks - for example firemen in action, and in sports, both amateur and professional. Although the mHealth solution could be usable almost entirely on its own, it should be integrated into the existing health care system for bigger impact.

Existing cloud-based mHealth solutions [2] are mostly aimed at gathering, storing, using, and sharing health information online. Most of them suggest various sensors to users but these sensors are all specialized to perform a single task. There is no integration of multiple sensors and no integration of these solutions with the modern electronic health records. The PCARD platform integrates users, caregivers, and medical community. It provides safe data transfer, secure data storage and manipulation, and application services for a completely integrated solution while it allows for manual interventions to the system if they prove to be necessary.

To evaluate PCARD, four pilot studies will be implemented. The resulting integrated health care model will advance the current health care system by providing additional links, data and knowledge pathways between the patients, their family and other informal care givers, and formal care givers.

This paper is based upon Kališnik et al. [3], extended with the following new contributions: (i) the personal computer software for ECG analysis has been added to PCARD, (ii) personal terminal software description has been expanded, (iii) the pilot studies performances have been expanded.

The rest of this article is as follows. First the method is explained – the PCARD platform for constant monitoring of vital ECG parameters. Next, the main constituents of the PCARD platform are described - the sensor device, personal terminal software, cloud-based software, and personal computer software. Then the pilot studies are presented, on which the PCARD platform will be evaluated. Finally, in Conclusion, the obtained results are summarized and the required future steps for PCARD implementation are listed.

## 2 Methods

We propose PCARD platform - a scheme of an mHealth system for mobile monitoring, which is schematically shown in Figure 1. The system comprises a small wearable device, Android application, protocols for data transfer, and software on the cloud.

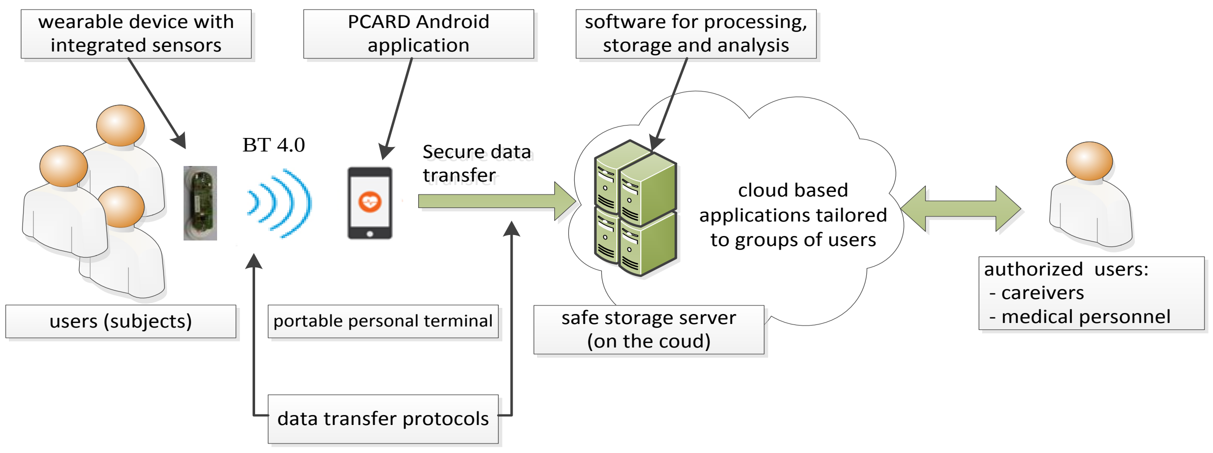

wearable device with integrated sensors
PCARD Android application
software for processing, storage and analysis
BT 4.0
Secure data transfer
cloud based applications tailored to groups of users
users (subjects)
portable personal terminal
safe storage server (on the coud)
authorized users:
- careivers
- medical personnel
data transfer protocols

Figure 1: PCARD platform as adapted for the follow-up of cardiac patients.

The wearable device is a wireless ECG sensor, it has measuring and communicating capabilities, microcontroller, and software designed specifically for it. The application on ubiquitous Android based mobile devices (such as tablets and smartphones) has two functionalities: to act as a link between the wearable device and the cloud, and to display the measured data. The cloud software residing on secure computer servers takes care of the measurement storage, analysis, and interface for various cloud applications for all the intended PCARD users – subjects themselves, the authorized caregivers and authorized medical personnel. Standard technologies are used for the transmission of the measurements, e.g. Internet, Bluetooth Smart, Wi-Fi, SSL, and SOAP. Usage of cloud and standard communication building blocks offers an inexpensive implementation, as well as wide availability of the system.

The design of the system takes into account the existing technical standards, allowing easy connection of various wearables and their immediate replacement if an improved version becomes available. Besides the ECG sensor, the system architecture allows the inclusion of additional sensors on the same wearable or the additional wearables, which could help improve the monitoring of the patient's condition, for example, sensors for remote monitoring of respiratory acoustic phenomena (cough, obstruction), or sensors for activity detection, etc. Some of the measurements are already feasible with our custom sensors [4].

Based on the graphic presentation of a critical vital parameter and its recent changes, it is possible to evaluate the effectiveness of a treatment and to foresee a possible deterioration. An alarm can be implemented to alert the medical personnel on the high possibility of deterioration before the monitored vital parameter reaches a critical value. Based on the simultaneous evaluation of multiple variables, the automatic analysis can provide the threat level and its trend (MEWS) [5]. The analysis of vital functions in a longer time period allows for the implementation of cognitive methods, for example, analysis of a cardiogram over longer time period contributes to the personalized patient's threat level[6][7][8].

## 2.1 ECG body sensor

The heart of the PCARD system is its small and lightweight wearable device, which is fixed to the subject's skin using standard self-adhesive electrodes. The device measures ECG with high resolution, which is suitable for both personal and clinical use. In addition to the ECG, the device also senses its environment, including position and movement of the subject [4][9], and subject's skin temperature, thus providing information about the measurement conditions. This device is also suitable for inclusion of other features, such as: EEG, vascular pressure, skin resistance and respiratory rate measurements. Such an electrode represents an important worldwide technological breakthrough.

With a single charge of the built-in battery, the device can operate continuously for more than three days. The device itself is extremely simple to use, to maintain and clean, as it requires no setup, it exposes no cables or switches, and is enclosed in smooth biocompatible plastics. With no movable parts it is extremely robust, can be made watertight, and by itself poses no absolutely risk to the users. It is intended for wide individual use, is affordable and is the basic building block of the mHealth technological network.

With an appropriate placement of the device on the chest, good visibility of all electrocardiographic waves (P, QRS and T) can be achieved with the quality sufficient for medical analysis. Therefore, PCARD can be used to help identifying arrhythmias and other cardiac conditions [5]. In contrast, the measurements from implanted ECG recorders (Implantable Loop Recorders [10]), often record P waves that are poorly visible or not visible at all. With an ECG sensor fixed by standard electrodes on the chest as is case with PCARD, the placement of electrodes can be easily modified and fine-tuned to maximize the recording quality of the desired electrocardiographic waves, and allows for a better quality ECG recording.

To maximize their potential, wearable devices should be non-disruptive to their users; to this end, PCARD device is made small, multifunctional and wireless. It can be worn under any kind of garments. We have already successfully prototyped a differential wireless sensor for measuring body surface potential on short distances. One of the prototypes is shown in Figure 2 with the raw measured ECG signal displayed on the Android based smartphone.

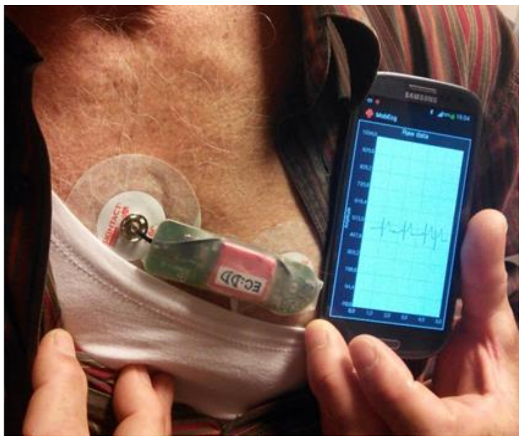


Figure 1: Prototype personal equipment of PCARD system: small body sensor and Smartphone.

## 2.2 Personal terminal

The necessary link between the ECG sensor and the ECG analysis available to appropriate experts is a personal terminal – a device with computing and interconnection capabilities that the user carries around at all times. Ubiquitous smart phones can readily fulfill this role [12] and the PCARD makes extensive use of Android powered smart phones.

Personal terminal connects to the ECG sensor via Bluetooth Smart protocol and records everything that the ECG sensor measures. This protocol offers sufficient bandwidth, data encryption and is low-power, meaning it conserves well the ECG sensor's and the personal terminal's batteries, which are the limiting factor for autonomy length.

For an integrated care solution, the measured data should be available to the patient herself as well as to the medical personnel. Therefore, the personal terminal analyses and displays the data to the user in real-time and in a user-friendly fashion on the personal terminal display, while it also forwards the un-processed data to the cloud via Internet connection. The data can be sent either in near real-time (with only several seconds of delay) or in larger packets, for example once a day, depending on the monitoring purpose. It can also be processed before sending to lower the data size and to provide only the pre-defined statistics to the cloud.

Software running on the personal terminal could be extended to provide alerting in case it detected life-threatening heart conditions. The alerting could be directed either to the user, to the nearby health care provider or even to the emergency dispatch center. Furthermore, software could be extended to provide assistance for physical training or for workers that often face life-threatening situations, such as firemen, policemen, etc.

## 2.3 Secure data transmission and storage

Bluetooth Smart technology enables encryption of transferred data between the ECG sensor and the personal terminal. The measured data are only temporarily stored on the personal terminal storage and are accessible to be processed by other software. Transfer of data to the cloud is again encrypted and largely depersonalized – personal data is never transmitted with the measurement. It is also only not required (although it can be) to be stored on the cloud. In the end, aside from the users themselves, only authorized medical personnel possess the personal information of the subjects of measurements.

## 2.4 Cloud software

Medical and informal caregivers can access the measured data and their visualization on the cloud according to their permissions, which are managed with a safe and reliable accounting system. Since the PCARD system is on the cloud, customized interfaces can be provided for various medical personnel profiles to aid them in using, viewing, and analyzing the data. Private users also get custom interface to access their own data, although they may instead use only their personal terminal without ever connecting to the cloud.

The data on the cloud also offer a unique opportunity for scientific exploration to advance the medical knowledge. Never before have ECG measurements of multi-day length and enriched with subject's activity data been available in large quantity. PCARD offers an interesting new Bigdata problem – immense quantities of novel data type to be explored with statistics and data-mining algorithms. Although processing this data will be difficult, the state-of-the-art Bigdata techniques should be able to tackle it. If not, the Bigdata is being heavily researched and should provide adequate tools soon enough.

Opening the PCARD interfaces for custom made applications and add-ons that communicate either with the cloud or the personal terminal software provide the opportunity to extend the pre-designed use cases of PCARD. A more suitable representation of the measured data for the laymen may be discovered, or the possibility for the patient to better monitor his or her vital functions added. We have already developed an Android application that provides a comfortable option for tele-monitoring the heart activity, and display the real-time data from the electrode. The options for using the measured data are endless, however, they will have to be approached with caution, since sensitive personal data are at stake and should be protected with great care.

## 2.5 Personal computer software

During the pilot studies, future cloud software functionality is emulated by modified NeuroEKG software for personal computers [13]. NeuroEKG (see Figure 3) is a software package for semi-automatic analysis of ECG and correlated bio-signals, e.g., blood pressure. It will be used within the pilot studies for the prototyping, testing, and evaluation of algorithms and procedures.

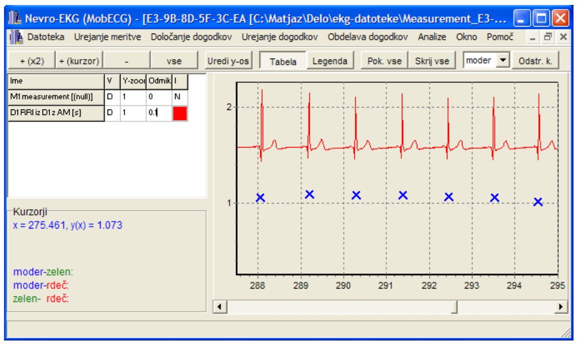


Figure 2: A sample screen from NeuroEKG application with an ECG channel (red curve), measured with PCARD system and the detected individual heart beats (blue x-es).

To pave the way for greater integration of mHealth services into health care and the daily lives, the medical personnel should first get a better understanding of what mHealth offers. Pilot studies will partially also focus on familiarizing the participating medical staff with the PCARD capabilities and limitations. Their understanding of ECG processing should evolve in this time, and their requirements towards the software should mature. NeuroEKG will serve well to showcase the software capabilities, to manually try out various algorithms, and to discuss ways of automating software processing. Thus the pilot studies will help mature the requirements for cloud-based software and for possible future evolution of PCARD system.

## 2.6 Pilot studies of PCARD platform applications

In order to show that the proposed system for the monitoring of mobile health is applicable at the various stages of the integrated health care, four pilot studies were designed for validation and evaluation of the medical, scientific, social and industrial impacts.

### 2.6.1 POAF - UKC Ljubljana

Postoperative atrial fibrillation (POAF) is a common complication of cardiac surgery. It results in many complications and increased healthcare resources [14]. Despite substantial findings in prediction and prevention of POAF, there is still some uncertainty about the risk stratification and the management of POAF.

Department of Cardiovascular Surgery of the University Medical Center Ljubljana (UKC Ljubljana) extends previous studies [15][16] and introduces a clinical study about the mechanisms of atrial fibrillation through recognizing dynamics of heart rhythm and electrophysiological properties of the heart after cardiac operation. This clinical study will use the PCARD system for constant monitoring of patients after a surgery and evaluation of malignancy of the rhythm and estimation of electrophysiological properties. Within the pilot study, a tablet showing current heart rhythm will be posted near the patient's bed and the analysis of the ECG will be done off-line. If the study is able to demonstrate constant on-line monitoring with automatic recognition of certain events is beneficial, the required functionality will be implemented in PCARD to provide on-line analysis.

The constant all-day ECG monitoring with PCARD will start immediately after the surgery, and end on the fifth day after the surgery. Atrial fibrillation is expected to occur in some of the monitored patients in this time frame. It is theorized that such monitoring should enable preventive activity, based on detected anomalies prior to the fibrillation itself.

### 2.6.2 Physicians - HCL Ljubljana

Community Health Centre Ljubljana (HCL) is a development oriented institution in primary health care with 1400 employees and with more than 400000 registered patients. The patients of HCL are treated within the medical doctrine and the ethics defined in it. HCL wishes to ensure a high-quality and time-optimal access to health care services for all of their users in all segments of acting. The patients of HCL come from the Slovene capital Ljubljana and its periphery. This is a diverse environment that includes urban and rural areas.

The proposed pilot system could trigger the penetration of the ICT and mHealth solutions in main-stream medicine in the primary care level and could improve and integrate the health care. PCARD will be used in parallel to current procedures of treating patients with unconfirmed palpitations or other heart rhythm disturbances. The patients participating in the study will already be provided with a more comprehensive care in terms of preventive and curative treatments than they would be using only the existing primary care.

### 2.6.3 Monitor - Terme Dobrna

Terme Dobrna (TD) is a health resort, well recognized by the two pillars: the positive impact of their natural healing factors on the health and the advanced medical treatment practices. TD is the only Slovenian thermal spa that holds international accreditation by DNV GL for quality and safety of its medical services. Medical center within the resort operates in fields of inflammatory rheumatic diseases, degenerative soft tissue rheumatism, post injury and post operation rehabilitation, and rehabilitation of neurologic (post stroke) and oncologic patients (mostly in fields of gynecology and urology). A part of TD is also *Institute for applied research in medical rehabilitation*, which aims to develop advanced, modern, evidence based, and software supported medical services that could be offered to their customers and therefore represents an ideal partner for a pilot study.

Pilot study will partly cover TD's wish to provide additional services to their customers. The goal will be to evaluate the hypothesis that visit to TD improves the persons overall health state. Within the pilot study, some of the customers could be provided with constant or intermittent health status monitoring for the time of their visits. ECG monitoring should provide enough data to evaluate or follow up on the eventual changes in person's health status. While the most important result of the study will be weather ECG monitoring can help evaluate health state changes, the customers receiving the monitoring within the study will already benefit, by receiving valuable health state information.

### 2.6.4 MDS - Rehabilitation Izola

Muscular Dystrophy Association (MDS) of Slovenia cares for the constant health and rehabilitation of their members. The people with special needs, such as those with muscular dystrophy, are exposed to health induced life threatening situations much more easily than the healthy population and therefore much more interested in the introduction and application of ICT-supported health care devices and services.

The study will focus on personalized health state monitoring of people staying in the MDS rehabilitation center. During the rehabilitation period, PCARD will be used for ECG monitoring, which will provide enough data to assess and analyze the heart condition. Treatment will be personalized and will include all the specificity of individual patients.

The study will shed more light on the usability of PCARD for relatively healthy population. Besides the studied assessment of short-term trend in health status, study will open up new options in: follow-up examinations for assessment of the long-term health status trends, and the impact of the physical activities on the health status. The latter could help in the preparation of personalized rehabilitation programs for more demanding customers.

#### 2.6.5 Requirements and targets of pilot studies

The requirements and performances of the PCARD platform pilots are shown in Table 1. The success of PCARD will be validated through the opinions of all the participating actors, that is, its direct users and the medical personnel. Targets of PCARD to be implemented though the described pilot studies:

- Near zero obtrusiveness of the sensor device to the direct users.
- Intuitive operation with the sensor device and the mobile device software.
- Inclination of the direct users towards the mHealth monitoring provided by PCARD.
- Ease of use of software, both on the mobile device and on the Cloud.
- Minimal overhead for the medical personnel, even when PCARD is used in addition to the standard procedures.
- Data security and safety on the communication channels between all the actors (either users or devices) in the pilots.

Furthermore, pilot studies will help refine the future direction of development both the wireless ECG sensor and the software on all levels of PCARD.

| Requirements/performances | POAF - UKC Ljubljana | Personal doctor - HCL | Monitor - TD | MDS - Rehabilitation Izola |
|---|---|---|---|---|
| Number of concurrently monitored users | 6 | 3 | 3 | 2 |
| Measurement length per user | 6 days | 3 days | up to 5 days | up to 5 days |
| Data stream capacity per user | 200 Bytes/second | 200 Bytes/second | 200 Bytes/second | 200 Bytes/second |
| Amount of generated data per day | 0.1 GBytes/day | 52 MBytes/day | 52 MBytes/day | 34 MBytes/day |
| Number of medical experts involved | 6 | 3 | 3 | 2 |
| Study of performance | Atrial fibrillation warning sign recognition | Palpitation detection | Short or long term health state assessment | Short or long term heart condition assessment |

Table 1: PCARD platform as adapted for the follow-up of cardiac patients.

## 3 Conclusion

We have designed four pilot systems around the PCARD platform, designed for long-term monitoring of users at cardiac risk or those who wish to evaluate their health state, using mHealth solution for data acquisition and medical expert support for analysis. Appropriate medical expertise and the required form of the produced reports will be identified during the experimental period of the maintained pilot systems. It is expected that the responses of medical personnel to eventual changes in users' health state will be faster and more objective when using the PCARD pilot system; therefore the users will experience improved level of treatment and better correlation between their perceptive and actual health conditions.

The pilot studies alone demonstrate that the applicability of the PCARD system is not limited to hospitals and health care centers, where the added benefit of the system will enable “doctor-to-doctor” and “patient-to-doctor” communication. We expect also to obtain a large amount of user opinions about the level of obstructiveness of the protested system. Based on the pilot results, an improved and clinically evaluated system will be developed for the international market with a wide spectrum of opportunities for R&D companies.

The system can be also installed in non-specialized medical institutions, e.g. health centers, nursing and patients' homes for early postoperative care and similar. The patient-friendly approach can contribute to easier evidence-based health evaluation and to advantageous innovative services in wellness and health centers.

## 4 Acknowledgments

This work was partially supported by the Slovenian Research Agency under Grant P2-0095. Thanks to anonymous volunteering users and medical staff of UKC Ljubljana, Terme Dobrna, Community Health Centre Ljubljana and Društvo distrofikov Slovenije.

## 5 References

[1] Lobodzinski, S. S. (2013). ECG patch monitors for assessment of cardiac rhythm abnormalities. Progress in Cardiovascular Diseases, 56(2), 224–229. doi:10.1016/j.pcad.2013.08.006

[2] "HealthVault." 2007. Accessed at https://www.healthvault.com/ on April 14, 2015.

[3] Kališnik J M, Poplas-Ssusič T, Semeja A, Korošec T, Trobec R, Avbelj V, Depolli M, Stanič U. Mobile health monitoring pilot systems. Proceedings of the 18th International Multiconference Information

Society - IS 2015, October 9th and 12h, 2015, Ljubljana, Slovenia. volume G.

[4] Trobec R., Avbelj V., Rashkovska A., Multi-functionality of wireless body sensors. The IPSI BgD transactions on internet research. 2014;10:23-27.

[5] C.P. Subbe, M. Kruger, P. Rutherford and L. Gemmel, "Validation of a modified Early Warning Score in medical admissions", Q. J. Med. 94, 521-526, 2001.

[6] R.Miller, "Rise of the machines: Computers construct new, better biomarkers", theheart.org [Clinical Conditions > Imaging > Imaging], October 5, 2011. Accessed at http://www.theheart.org/article/1290375.do on February 3, 2012.

[7] S. Esposito et al., "Altered cardiac rhythm in infants with bronchiolitis and respiratory syncytial virus infection", BMC Infect. Dis. 10, 305, 2010.

[8] Trobec R, Rashkovska A, Avbelj V. Two proximal skin electrodes - a respiration rate body sensor. Sensors, 2012, vol.12, no. 10, pp. 13813-13828.

[9] Gjoreski H, Rashkovska A, Kozina S, Luštrek M, Gams M. Telehealth using ECG sensor and accelerometer. Proceedings of MIPRO 2014, 37th International Convention, May 26-30, 2014, Rijeka, Croatia. pp. 283-287.

[10] Zellerhoff C, Himmrich E, Nebeling D, Przibille O, Nowak B, Liebrich A., How can we identify the best implantation site for an ECG event recorder? Pacing Clin Electrophysiol 2000;23:1545–9.

[11] Tomašić I., Frljak S., Trobec R., Estimating the universal positions of wireless body electrodes for measuring cardiac electrical activity. IEEE transactions on bio-medical engineering. 2013;60:3368-3374.

[12] Rashkovska A, Tomašić I and Trobec R, "A telemedicine application: ECG data from wireless body sensors on a smartphone", Proceedings of MEET & GVS on the 34th International Convention MIPRO 2011, Opatija, Croatia, May 2011, vol. 1, 293-296.

[13] Trobec R, Avbelj V, Šterk M, Meglič B, Švigelj V. Neurological data measuring and analysis software based on object oriented design. Clinical autonomic research, 2005; 15; 173.

[14] Kaireviciute D, Aidietis A, Lip GYH. Atrial fibrillation following cardiac surgery: clinical features and preventive strategies. Eur Heart J. 2009; 30: 410-25.

[15] Kališnik J M, Avbelj V, Trobec R, et al. Effects of beating- versus arrested-heart revascularization on cardiac autonomic regulation and arrhythmias. Heart Surg Forum. 2007; 10: E279-87.

[16] Ksela J, Suwalski P, Kalisnik J M, et al. Assessment of nonlinear heart rate dynamics after beating-heart revascularization. Heart Surg Forum. 2009; 12: E10-6.

# Modular Integrated Probabilistic Model of Software Reliability Estimation

Roman Yu. Tsarev, Alexey S. Chernigovskiy, Elena N. Shtarik and Andrey V. Shtarik
Siberian Federal University, Department of Informatics, Krasnoyarsk, Russia
E-mail: tsarev.sfu@mail.ru

Mustafa S. Durmuş
Pamukkale University, Department of Electrical and Electronics Engineering, Denizli, Turkey
E-mail: msdurmus@pau.edu.tr

Ilker Üstoglu
Yildiz Technical University, Department of Control and Automation Engineering, Istanbul, Turkey
E-mail: ustoglu@yildiz.edu.tr

**Keywords:** software reliability, reliability estimation, mean time to failure, mean time to repair, availability, multiversion software

**Received:** October 22, 2015

*A modular integrated probabilistic model of software reliability estimation and an algorithm of its application for estimation of software reliability with different architecture such as multilevel, multiversion, distributed and object-oriented ones are presented in the article. The modification of this model is given there for the object-oriented multiversion software with the distributed architecture. The procedure of its estimation is perfected to improve the quality of the reliability prediction. The description of the developed program system based on the modular integrated probabilistic model of reliability estimation of the object-oriented multiversion software with the distributed architecture is presented in the article. The analysis of relation of software reliability parameters to the component count, conditional and unconditional probability of the failure appearance in components and temporary components characteristics is done there as well.*

*Povzetek: Opisan je modularni verjetnostni model za oceno zanesljivosti programske opreme.*

## 1 Introduction

The interest to the software reliability estimation has arisen at the same time as the software origin. It has been caused by the natural need to get traditional probabilistic software reliability estimation as one of the computer system components. Originally the approach to the computer system parts reliability estimation was a little different from the hardware reliability estimation and it consisted in application of well-known statistical methods of classical reliability theory in a new technological branch which laid the corner stone of the individual trend like the software reliability theory [22]. However, as far as computing machinery was developed it became obvious that software was not only the part of the computing system.

In the modern conditions of digital technology development the software discontinued to be a part of the one computing system as it used to be, it began to be used on hundreds and thousands of similar computers (basically, on personal ones) [16]. It is obvious that the problem of assurance of the stable programs functioning, identification and correcting the failures in programs sharply exists for software developers nowadays.

Over previous decades, lots of approaches, models and methods of software reliability research have been created [3], [4], [5], [19]. However, any unified approach to the solution of this problem has not been proposed yet and, apparently, it will not happen in the near future. Nevertheless, developing difficult programs systems, their creators are trying to get software reliability estimation [8], [17], [20]. One of the most effective approaches consists in sequential estimation of the programs reliability at every stage of their development [10], [19]. The main difficulty in using statistical methods is the absence of the sufficient amount of the input data. The detection of errors dynamics should be thoroughly registered and processed. Another important problem is a grain size of element's computing reliability [7], [14]. Defining all the paths of program execution during information processing as it sometimes offers is virtually unreal even for an easy program. According to this, the elements' computing reliability detailing (they are theoretically called program modules) should be limited by the completed program formations, which are connected to each other, compose more complicated unit (complex) which reliability holds our interest [6], [11], [12]. In this case it is acceptable that the computing machinery, the operating system and the programming environment are absolutely reliable. Of interest is only the reliability of functioning of special software tools which solve the main system problem [21].

As the result of the analysis of many researchers' works [2], [5], [9], [13], [15], [16] in the field of software reliability research, three basic problem groups can be distinguished. They are:

- the absence of the unified methodology of high-reliable software system development;
- the absence of the unified methodology of high-reliable software system testing;
- the absence of the unified approach in software systems reliability estimation and analysis.

One of solutions of the previous problems is the usage of the software reliability estimation models presented in this paper. The generic modular integrated probabilistic model of software reliability estimation and its modification for the multiversion software with the distributed architecture are adapted to the modern analysis and software development methods; in particular the option of application of the models for the software building following the object-oriented approach is presented there.

## 2 Methodology

### 2.1 The generic modular integrated probabilistic model of software reliability estimation

The following generic modular integrated probabilistic model of software reliability estimation has been developed to evaluate the reliability parameters of the software.

It is obligatory to satisfy the condition for this model:

$$\sum_{i=1}^{F} PU_i = 1,$$

where $F$ is a number of software architecture components; $PU_i$ is probability of using component $i$, $i = 1, \ldots, F$.

The mean time to repair is calculated as follows:

$$\begin{aligned} MTTR = \sum_{i=1}^{F} [PU_i \times PF_i \times [TA_i + TC_i + TE_i + \\ + \sum_{j=1, j\neq i}^{F} [PL_{ji} \times [TA_j + TC_j + TE_j + \\ + \sum_{l, j \in D} [PL_{lj} \times (TA_l + TC_l + TE_l)]]] + \\ + \sum_{k, i \in D} PL_{ki} \times [TA_k + TC_k + TE_k + \\ + \sum_{m=1, m\neq j}^{F} [PL_{mk} \times [TA_m + TC_m + TE_m + \\ + \sum_{l, m \in D} PL_{lm} \times (TA_l + TC_l + TE_l)]]]]]. \end{aligned} \quad (1)$$

where $M$ is a number of the software architecture levels; $PF_i$ is theoretical probability of component $i$ failure, $i = 1, \ldots, F$; $PL_{ij}$ is conditional probability of component $i$ failure under component $j$ failure, $i = 1, \ldots, F$, $j = 1, \ldots, F$; $TA_i$ is relative time of the access to component $i$, $i = 1, \ldots, F$; $TC_i$ is relative time of failure's analysis in component $i$, $i = 1, \ldots, F$; $D_{mj}$ is disjoint sets of component $j$ at level $m$, $m = 1, \ldots, M$, $j = 1, \ldots, F$; $TE_i$ is relative time of failure recovery in component $i$, $i = 1, \ldots, F$.

The mean time to failure is calculated as follows:

$$\begin{aligned} MTTF = \sum_{i=1}^{F} [PU_i \times (1 - PF_i) \times [TU_i + \\ + \sum_{j=1, j\neq i}^{F} [(1 - PL_{ji}) \times [TU_j + \sum_{l, j \in D} [(1 - PL_{li}) \times TU_l + \\ + \sum_{k, i \in D} [(1 - PL_{ki}) \times [TU_k + \sum_{m=1, m\neq i}^{F} [(1 - PL_{mk}) \times [TU_m + \\ + \sum_{l, m \in D} [(1 - PL_{lm}) \times TU_l]]]]]]]]], \end{aligned} \quad (2)$$

where $TU_i$ is relative time of using component $i$, $i = 1, \ldots, F$.

The software availability ratio is calculated as follows:

$S = MTTF / (MTTF + MTTR).$

The software reliability is computed as follows:

$$R_S = \sum_{i=1}^{F} PU_i \times R_i \text{, where } R_i = 1 - \prod_{k \in Zi} PF_{ik} . \quad (3)$$

where $R_i$ is component $i$'s reliability, $i = 1, \ldots, F$; $Z_i$ is a set of component $i$'s versions, $i = 1, \ldots, F$.

The cost of software development is calculated as follows:

$$C_s = \sum_{i=1}^{F} \sum_{j \in Zi} C_j ,$$

where $C_i$ is the cost of component $i$'s development, $i = 1, \ldots, F$.

### 2.2 The algorithm of using the generic modular integrated probabilistic model of software reliability estimation

The algorithm of software reliability parameters' evaluation with the help of the developed generic modular integrated probabilistic model is described below.

Algorithm 1: software reliability parameters' evaluation with the help of the developed generic modular integrated probabilistic model

1) Divide the estimating software into modules, define the modules' scopes, their characteristics and interaction order.
2) Define the number of architecture levels. If the architecture is multilevel, it is necessary to pass to step 4 or follow step 3 if it is not.
3) Eliminate $D_{mj}$ from the model in formulas (1) and (2). Next, pass to step 4.
4) Define the number of versions. If the architecture is multiversion, it is necessary to pass to step 6 or follow step 5 if it is not.
5) Eliminate $Z_i$ from the model in formula (3). After that, pass to step 6.

6) Define if it is possible to eliminate failures. If it is not, pass to step 8 or, if it is so, follow step 7.
7) Eliminate $TC_i$, $TE_i$ from the model in formula (1). Then, pass to step 8.
8) Get summarized expressions *R*, *MTTR*, and *MTTF*, solving formulas (1), (2), and (3).

There is a flowchart of the algorithm of the generic modular integrated probabilistic model of software reliability estimation in Figure 1.

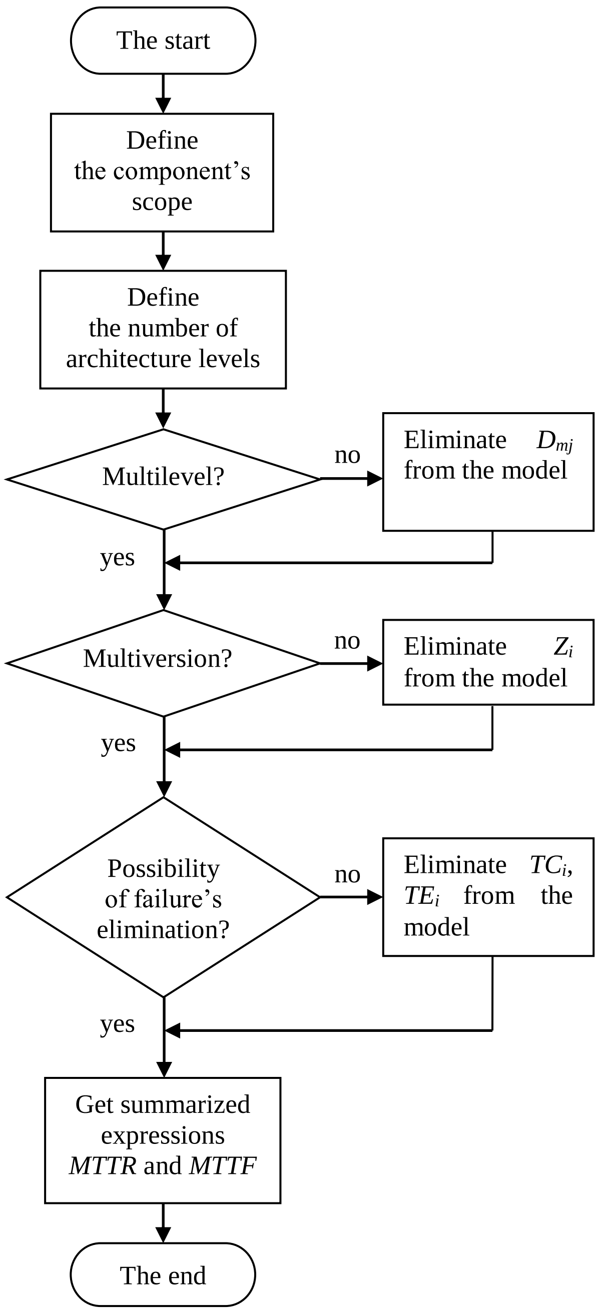

The start
Define the component's scope
Define the number of architecture levels
Multilevel?
no
Eliminate $D_{mj}$ from the model
yes
Multiversion?
no
Eliminate $Z_i$ from the model
yes
Possibility of failure's elimination?
no
Eliminate $TC_i$, $TE_i$ from the model
yes
Get summarized expressions *MTTR* and *MTTF*
The end

Figure 1: Flowchart of the algorithm of the generic modular integrated probabilistic model of software reliability estimation.

## 2.3 The modular integrated probabilistic model of reliability estimation of the object-oriented multiversion software with the distributed architecture

The difficulty in the usage of the generic modular integrated probabilistic model at the step of designing the software architecture is that all required parameters are not always known. If the component reliability is unknown beforehand, it can be estimated only at the coding stage. More exact information about reliability can be obtained at the module testing stage. The probability of using the component and component's failure can be gained after software testing. Parameters such as access, analysis and recovery component time for the distributed multiversion software can be estimated after testing, so it is not ruled out that the structure formation of the architecture of the projectable software can be at the conceptual phase. It is possible to build a class hierarchy and method's tree for the object-oriented software. In general, at this step it is necessary to set the parameters which have to be estimated at the following stages.

According to the object-oriented approach computational process is a consecutive calling sequence of class methods. The number of architecture levels equals 1 for this variant. Such parameters as access, analysis and recovery time are parts of the distributed multiversion software.

Let us examine the modification of the generic modular integrated probabilistic model for the instance of the object-oriented multiversion software with the distributed architecture in detail.

The software architecture is a set of class hierarchies for the object-oriented approach. Every class is a set of properties (variables) and methods (functions) of the object.

The process is a set of transitions from one class method to different class method [1], [9]. It is obligatory to satisfy the condition for this model:

$$\sum_{i=1}^{F} PU_i = 1,$$

where $F$ is a general component (class) count in the software architecture, $PU_i$ is a probability of component $i$'s usage, $i = 1, \ldots, F$.

The reliability of the multiversion component depends on the reliability of each version and meta-class which implements the multiversion approach:

$$R_i = (1 - \prod_{K \in Z_i} PF_{ik}) R_{mul},$$

where $R_i$ is a reliability of component $i$, $i = 1, \ldots, F$; $Z_i$ is a variety of component $i$'s versions, $i = 1, \ldots, F$; $R_{mul}$ is a reliability of the meta-class which implements the multiversion approach. Let us mention that the meta-class should not be considered as the architecture's component and it should be eliminated from computing *MTTR*, *MTTF*, and $R_s$.

The mean time to repair is calculated as follows:

$$MTTR = \sum_{i=1}^{F} [PU_i \times PF_i \times [TA_i + TC_i + TE_i +$$

$$+ \sum_{j=1, j \neq i}^{F} [PL_{ji} [TA_j + TC_j + TE_j ]]]].$$

The mean time to failure is calculated as follows:

$$MTTF = \sum_{i=1}^{F} [PU_i \times (1 - PF_i) \times$$

$$\times [TU_i + \sum_{j=1, j \neq i}^{F} [(1 - PL_{ji}) \times TU_j ]]].$$

The software availability ratio is computed in the following way:

$$S = MTTF / (MTTF + MTTR) .$$

The software reliability is calculated as follows:

$$R_s = \sum_{i=1}^{F} PU_i \times R_i \text{ , where } R_i = 1 - \prod_{k \in Zi} PF_{ik} .$$

As the suggested approach does not take into account the conditional probability of the failure in components, the following model modification was used in the implementation of the model:

$$R_s = \sum_{i=1}^{F} PU_i \times R_i \times \prod_{j=1, j \neq i}^{F} [PU_j \times (1 - R_j \times PL_{ji})] .$$

## 3 Results and discussion

Let us study the program realization of the system of the reliability estimation of the object-oriented multiversion software with the distributed architecture based on the presented model.

### 3.1 The system of the reliability estimation of the object-oriented multiversion software with the distributed architecture

The modular integrated probabilistic model of reliability estimation of the object-oriented multiversion software with the distributed architecture has been realized as the program system in C# language.

The operational system's function is:

the system user's provision of the information about the projectable software reliability parameters;

the definition of the likehood degree of the modular integrated probabilistic model of software reliability estimation in comparison with the real software.

The primary performing functions are:

the definition of the reliability parameters of the projectable software by means of the modular integrated probabilistic model;

the definition of the reliability parameters of the projectable software by means of estimation of its simulator's behaviour;

the visualization of components' behaviour of the software simulator in the time.

A great number of the system functions forms the structure from five blocks (Figure 2):

the data reduction provides data input and presentation in the form which is convenient for the user;

the modular integrated probabilistic model makes it possible to define the reliability parameters of the projectable software;

the simulator duplicates the behavior of the projectable software following the data which are obtained from the block of data reduction during the specified number of cycles;

the simulator monitoring is done for the statistics' gathering of the simulator work and definition of the reliability parameters of the projectable software following the collected data;

the output is done to lead the results of system work.

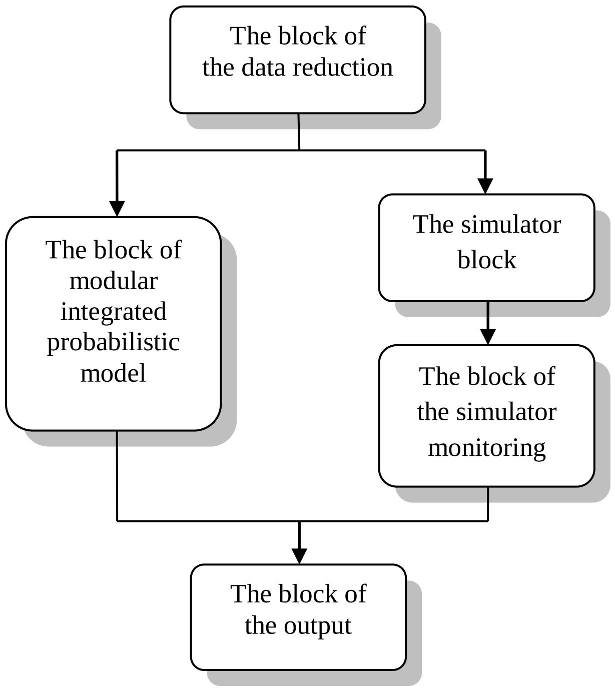

The block of the data reduction
The block of modular integrated probabilistic model
The simulator block
The block of the simulator monitoring
The block of the output

Figure 2: The structure of the system of the object-oriented multiversion software reliability estimation.

The subsystem "The block of the data reduction" serves for the solution of the following tasks:

data editing;

checkout of the correction of the posted data.

The statistical data about the structure of the projectable software are imported into the table with the clipboard or directly by the user.

The visualization of the array of software parameters and its components is performed as the table. In case of having a mistake in edited data the system user will be informed about it by means of the message "An error".

The subsystem "The block of the modular integrated probabilistic model" is basic in the system structure and serves for definition of the reliability parameters of the projectable system by means of using the modular integrated probabilistic model of estimation of object-oriented multiversion software with the distributed architecture.

The input data of the block of the modular integrated probabilistic model is a result of the subsystem "The block of the data reduction" works.

The subsystem "The simulator block" is basic in the system structure and serves for the imitation of the projectable software work in compliance with the data received from the subsystem "The block of the data reduction". The imitation of software execution continues during the time interval specified by the user. During the work of this block it is supposed that each component of the projectable software is invoked to execute the probability $PU_i$ during the time equal $TU_i$. At the same time during the execution of the component the failure will be made with the probability $PF_i$, which time equals the sum of $TA_i$ (the access time of the component $i$), $TC_i$ (the analysis time of the failure in the component $i$) and $TE_i$ (the time of failure's elimination in component $i$). Defining the failure, the probability $PL_{ij}$ (the probability of the failure in the component $i$ during the failure of the component $j$) is also considered.

The subsystem "The block of simulator monitoring" is assigned for the statistics information gathering about simulator work and defining the reliability parameters of the projectable software on basis of this statistics.

The subsystem "The block of output" serves to lead the results of the system work. It displays the operating schedule of the simulator for the user and the results in the work of the block of simulator monitoring and the block of the modular integrated probabilistic model.

## 3.2 The analysis of the modular integrated probabilistic model of reliability estimation of the object-oriented multi-versioned software with the distributed architecture

The analysis of the modular integrated probabilistic model of reliability estimation of the software includes the analysis of the model behaviour subject to the software components number, conditional and unconditional probability of the failures in the components, and also the relation of software reliability parameters to time characteristics of the components.

Let us guess that the software consists of homogeneous components with the following characteristics: the probability of using $PU_i = 1$, unconditional probability of the failure $PF_i = 0.1$, conditional probability of the failure $PL_{ij} = 0$ for all $j$, the access time $TA_i = 5$ cycles, the analysis time $TC_i = 7$ cycles, the clearing rime of the failure $TE_i = 10$ cycles, the average time of the using components $TU_i = 30$ cycles. The time of imitation is 1200 cycles.

The analysis of the software reliability relation to the component count has detected the different behavior pattern of reliability parameters in the modular integrated probabilistic model of software reliability estimation from the component count. Thus, for example, the relation of the mean time to repair $MTTR$ and of the reliability $R$ to the component count $F$ has a linear form (Figure 3 and 4). At the same time, the relation of the meaning of the mean time to failure to the component count has a nonlinear form (Figure 5).

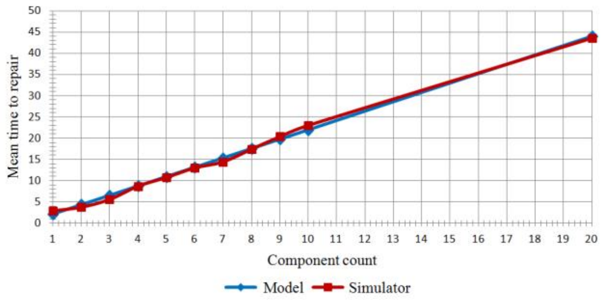


Figure 3: The relation of the mean time to repair to the component count.

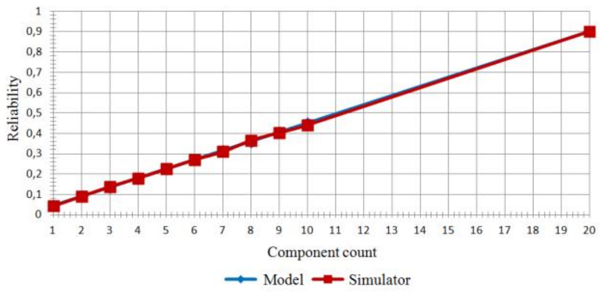


Figure 4: The relation of the software reliability to the component count.

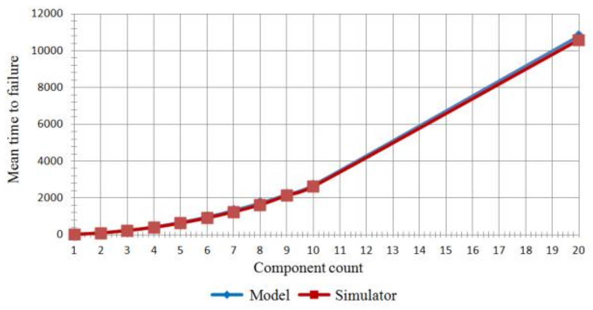


Figure 5: The relation of the mean time to failure to the component count.

Analyzing the relation of software reliability parameters to the committed component count $F = 10$ from the quantity of the unconditional probability of the failure in the software components, the linear growth of the mean time to repair time (Figure 6), the scaling-down of the mean time to failure and the reliability have been detected (Figure 7 and 8).

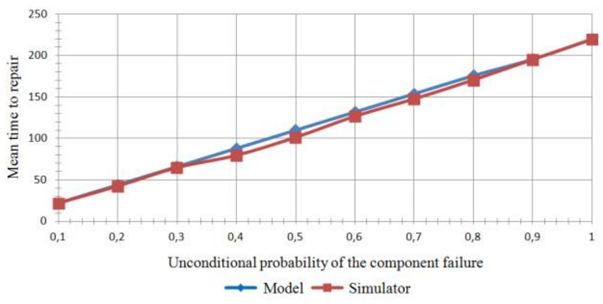


Figure 6: The relation of the mean time to repair to the quantity of unconditional probability of the failure in software components.

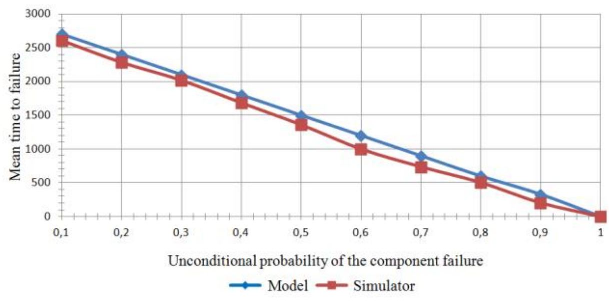


Figure 7: The relation of the mean time to failure to the quantity of the unconditional probability of the failure in the software components.

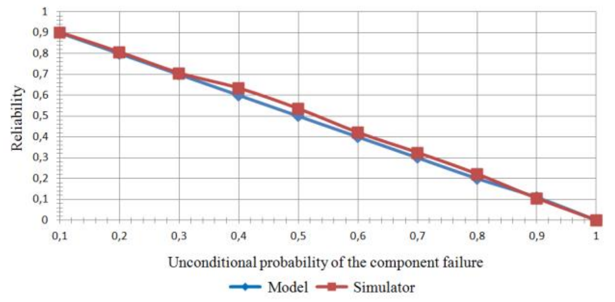


Figure 8: The relation of the reliability to the quantity of unconditional probability of the failure in software components.

Analyzing the relation of software reliability parameters to the committed component count $F = 10$ from the value of the conditional probability of the failure in the software components, the scaling-down of the mean time to failure *MTTF* is marked due to increasing of unconditional probability of the failure in the component (Figure 9). The scaling-up of the mean time to repair *MTTR* occurs during the augmenter of the unconditional probability of the failure in software components (Figure 10). At the same time, the relation of the probability point of the meaning of the software reliability *R* to the value of conditional probability of the failure in the component is absent (Figure 11).

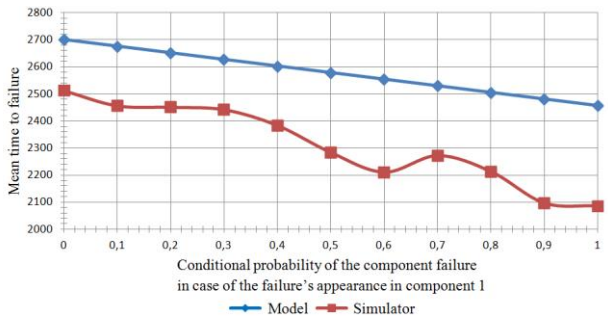


Figure 9: The relation of the mean time to failure to the conditional probability of the failure in the component in case of the failure's appearance in component 1.

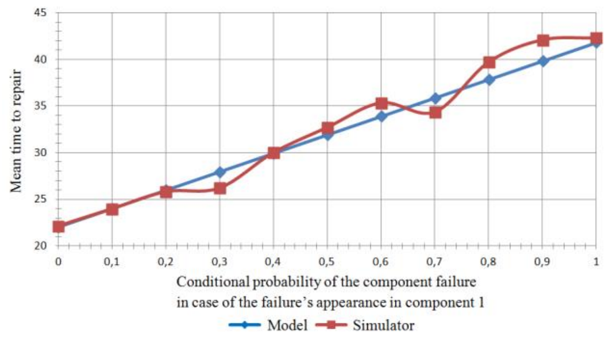


Figure 10: The relation of the mean time to repair to the conditional probability of the failure in the component in case of the failure's appearance in component 1.

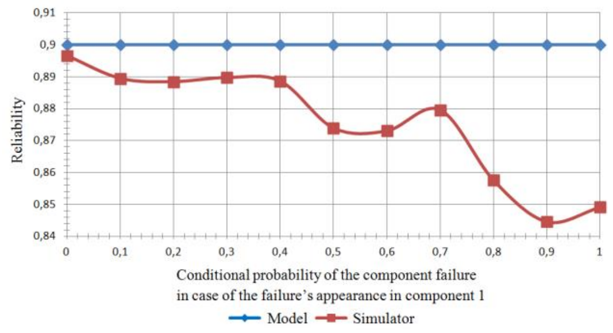


Figure 11: The relation of the software reliability to the conditional probability of the failure in the component in case of the failure's appearance in component 1.

As this exponent of the software reliability as the probability of no-failure operation does not take into account conditional probability of the failure in components, let us use its modified evaluation (10) to increase the quality of the forecast. The result is shown in Figure 12.

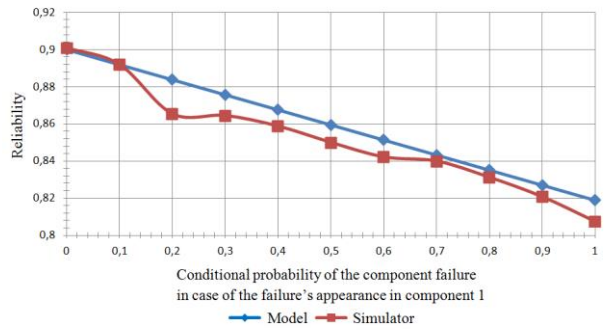


Figure 12: The relation of the software reliability to conditional probability of the failure in the component during the failure's appearance in component 1.

In Figure 13 there is a relation of the average time of usage of the component from the component count included in the software structure to the average time between the failures which equals 675 cycles.

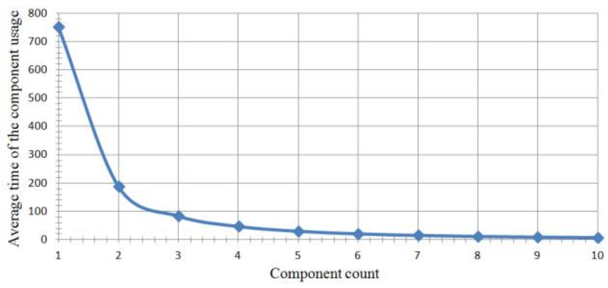


Figure 13: The relation of the average time of the usage of the component to the component count in the system.

In Figure 14 there is a relation of the mean time of the recovery of the component from the software component count to the average time of the system recovery which equals 11 cycles.

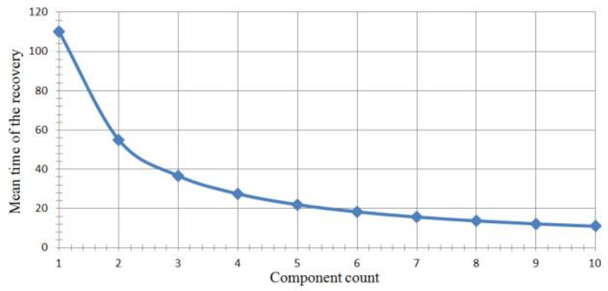


Figure 14: The relation of the mean time of the recovery after the failure of the component to the component count in the system.

During the analysis a backward exponential relation of the average time of the component recovery to the software component count has been detected.

The analysis has shown a high forecast accuracy of the meanings of the reliability parameters in the modular integrated probabilistic model of reliability estimation for the systems with a low intermodule relation. The degradation of the forecast accuracy has been detected for the systems with a high intermodule relation who is specified by a lack of attention to conditional probability of the component's failure and partial ignorance of the intermodule communications and the depth of system components integration. The presented modification of reliability calculation for the modular integrated probabilistic model permits to expand a model range of application and to improve the quality of forecasting.

## 4 Conclusion

The presented generic modular integrated probabilistic model of reliability estimation of software permits to do sums of assessment of the software reliability parameters of different architecture: multilevel, multiversion, distributed, object-oriented ones. The authors have offered the algorithm of the developed model application for the software reliability estimation with specified software architecture.

In the work the modification of generic modular integrated probabilistic model for the case of the object-oriented multiversion software with the distributed architecture has been analyzed in detail.

The developed system on the basis of the presented modification of the generic modular integrated probabilistic model for the case of the object-oriented multiversion software with the distributed architecture provides end-to-end solution of the following problems: the system user's support in reliability parameters of the projectable software and the definition of the adequacy degree of modular integrated probabilistic model of software reliability estimation towards the real software.

The research has confirmed high performance of the modular integrated probabilistic model of software reliability estimation which is characterized by the weak dependence between the modules. The nonlinear relation between the quantities of the average time of using the component, the average time of recovery after the component's failure and the number of the components in software, and also behaviour pattern of the model during the change of the quantities in conditional and unconditional probability of the failure in software components have been detected. It has been revealed experimentally that the mean time to failure and the mean time to repair linearly depend on unconditional probability of the failure in the components of software.

## 5 Acknowledgment

The reported study was funded by RFBR according to the research project №16-57-46016 CT_a and by TUBITAK according to the research project №215E196.

## References

[1] Abdallah, C., Hafida, B. (2010). A new architectural approach for dynamic adaptation of components-based software using multi agent system. *Control Engineering and Applied Informatics*, vol.12, no.4, pp. 43-50.

[2] Avizienis, A., Laprie, J.C., and Randell, B. (2001). *Fundamental Concepts of Dependability*, Research Report no. 1145, LAAS-CNRS.

[3] Avizienis, A., Laprie, J.C., Randell, B. and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, vol.1, no.1, pp. 11-33.

[4] Benso, A., Di Carlo, S. (2011). The art of fault injection. *Control Engineering and Applied Informatics*, vol.13, no.4, pp. 9-18.

[5] Boehm, B. (2011). *The Future of Software Engineering*, Springer Berlin Heidelberg.

[6] Golubev, I.M., Tsarev, R.Ju., Semenko, T.I. (2005). N-version software systems design. *11th International Scientific and Practical Conference of Students, Postgraduates and Young Scientists; "Modem Techniques and Technologies", MTT 2005 - Proceedings*, IEEE, Tomsk, Russian Federation, pp. 147-149.

[7] Huang, C.-Y., Hung, T.-Y. (2010). Software reliability analysis and assessment using queueing models with multiple change-points. *Computers and Mathematics with Applications*, vol.60, no.7, pp. 2015-2030.

[8] Huang, G., Mei, H., and Yang, F. (2006). Runtime recovery and manipulation of software architecture of component based systems. *Automated Software Engineering*, vol.13, no.2, pp. 257-281.

[9] Huang, C.-Y., Lin, C.-T. (2006). Software reliability analysis by considering fault dependency and debugging time lag. *IEEE Transactions on Reliability*, vol.55, no.3, pp. 436-450.

[10] Kang, W.-H., Kliese, A. (2014). A rapid reliability estimation method for directed acyclic lifeline networks with statistically dependent components. *Reliability Engineering and System Safety*, vol.124, pp. 81-91.

[11] Kulyagin, V.A., Tsarev, R.Y., Prokopenko, A.V., Nikiforov, A.Y., Kovalev, I.V. (2015). N-version design of fault-tolerant control software for communications satellite system. *2015 International Siberian Conference on Control and Communications, SIBCON 2015 - Proceedings*, IEEE Inc., Omsk, Russian Federation, pp. 1-5.

[12] Landon, J., Özekici, S., Soyer, R. (2013). A Markov modulated Poisson model for software reliability. *European Journal of Operational Research*, vol.229, no.2, pp. 404-410.

[13] Lee, W.S., Grosh, D.L., Tillman, F.A., Lie, C.H. (1985). Fault tree analysis, methods, and applications - a review. *IEEE Transactions on Reliability*, vol.34, no.3, pp. 194-203.

[14] Li, X., Xie, M., Ng, S.H. (2010). Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points. *Applied Mathematical Modeling*, vol.34, no.11, pp. 3560-3570.

[15] Myers, G.J., Hocker, D.G. (1981). Use of software simulators in the testing and debugging of microprogram logic. *IEEE Transactions on Computers*, vol.C-30, no.7, pp. 519-523.

[16] Okamura, H., Dohi, T., Osaki, S. (2012). Software reliability growth models with normal failure time distributions. *Reliability Engineering and System Safety*, vol.16, pp. 135-141.

[17] Park, G.-Y., Jang, S.C. (2014). A software reliability estimation method to nuclear safety software. *Nuclear Engineering and Technology*, vol.46, no.1, pp. 55-62.

[18] Rekab, K., Thompson, H., Wu, W. (2013). A multistage sequential test allocation for software reliability estimation. *IEEE Transactions on Reliability*, vol.62, no.2, pp. 424-433.

[19] Rekab, K., Thompson, H., Wu, W. (2013). An efficient test allocation for software reliability estimation. *Applied Mathematics and Computation*, vol. 220, pp. 94-103.

[20] Toader, C. (2010). Increasing reliability of web services. *Control Engineering and Applied Informatics*, vol.12, no.4, pp. 30-35.

[21] Tyagi, K., Sharma, A. (2012). A rule-based approach for estimating the reliability of component-based systems. *Advances in Engineering Software*, vol.54, pp. 24-29.

[22] Zheng, C., Liu, X., Huang, S., Yao, Y. (2011). A parameter estimation method for software reliability models. *Procedia Engineering*, vol.15, pp. 3477-3481.

# A Distributed Security Mechanism for Resource-Constrained IoT Devices

James King[1] and Ali Ismail Awad[1,2]
[1]Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology, Luleå, Sweden
[2]Faculty of Engineering, Al Azhar University, Qena, Egypt
E-mail: {jamyking@gmail.com}, {ali.awad@ltu.se}

**Keywords:** internet of things (IoT), IoT security, class-0 IoT devices , object data encryption

**Received:** October 30, 2015

*Internet of Things (IoT) devices have developed to comprise embedded systems and sensors with the ability to connect, collect, and transmit data over the Internet. Although solutions to secure IoT systems exist, Class-0 IoT devices with insufficient resources to support such solutions are considered a resource-constrained in terms of secure communication. This paper provides a distributed security mechanism that targets Class-0 IoT devices. The research goal is to secure the entire data path in two segments, device-to-gateway and gateway-to-server data communications. The main concern in the provided solution is that lighter security operations with minimal resource requirements are performed in the IoT device, while heavier tasks are performed in the gateway side. The proposed mechanism utilizes a symmetric encryption for data objects combined with the native wireless security to offer a layered security technique between the device and the gateway. In the offered solution, the IoT gateways provide additional protection by securing data using Transport Layer Security (TLS). Real-time experimental evaluations have demonstrated the applicability of the proposed mechanism pertaining to the security assurance and the consumed resources of the target Class-0 IoT devices.*

*Povzetek: V članku je analiziran mehanizem za varen prenos podatkov med napravami interneta stvari (IoT).*

## 1 Introduction

Recently, the Internet of Things (IoT), coined as such in 1999, has become an evolving paradigm in wireless communications [1]. IoT is now a hot topic in Information and Communication Technology (ICT) and has drawn the attention of many research institutions [2, 3]. The generic infrastructure of IoT is a network of devices or objects such as embedded computers, controllable and intelligent automated devices, sensors, and Radio Frequency IDentification (RFID) tags, in addition to the IoT gateway and the remote server. IoT devices have the ability to connect and exchange data with other devices and services over a network and over the global Internet [4, 5]. The deployments of IoT core technology encompass home automation, manufacturing, environmental monitoring, and medical and healthcare systems. A future mega-market is anticipated for a broad scope of applications that utilize IoT devices and technology [1].

The constrained IoT devices, Class-0 IoT devices, are devices with limited or constrained resources with respect to CPU processing power, ROM, RAM, and battery life. However, these devices still have the capability of providing their intended functionalities. The constrained IoT devices are often small in size with limited functions, such as sensors and smart devices controlling electrical appliances or services [6]. They are capable of collecting and transmitting data, such as sensor readings, across the Internet for storage and analysis. The collected and transmitted data may be personal, private, and sensitive. Figure 1 demonstrates a general architecture of an IoT system using an example of constrained IoT medical devices.

Due to a wide range of IoT applications, data security has become a major concern in IoT systems in addition to the system's scalability [7]. Information insecurity will directly impact the performance of the entire IoT system [8]. A study states that 70% of the ordinarily used IoT devices face security vulnerabilities such as insufficient authorization, lack of encryption, and insecure web interfaces [9]. In some application domains such as healthcare, data leakage can threaten the life of individuals. Therefore, developing security and privacy protection approaches is an imperative requirement [10, 11, 12]. While solutions exist to secure data from IoT devices, the majority of these solutions require support for Transport Layer Security (TLS) standards. Class-0 IoT devices fall short of the resource requirements to support most of the security approaches offered [13, 14]. Therefore, a particular security mechanism, which is designed for Class-0 IoT devices, is highly demanded.

This paper provides a distributed security mechanism that is appropriate for the Class-0 IoT devices. The philosophy behind the provided solution is that light resource-consuming object encryption is implemented on the IoT

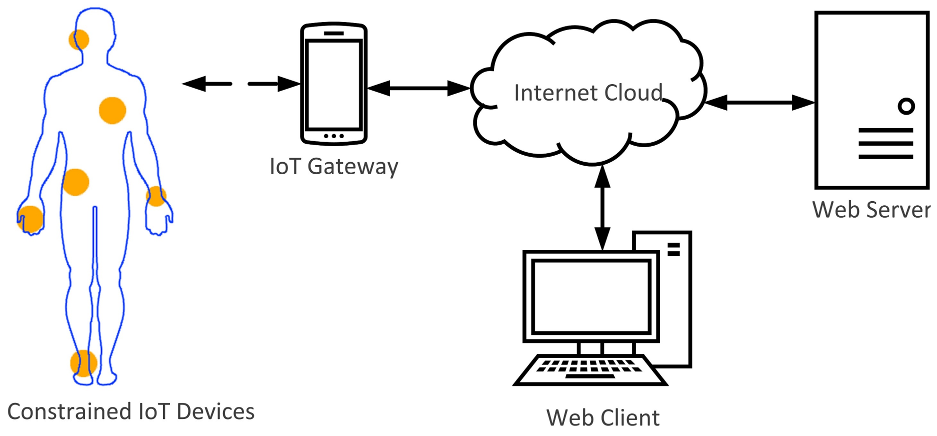


Figure 1: A generic IoT system using the example of resource-constrained IoT medical devices. The IoT system includes a network of devices, a gateway, and a web server. The IoT devices record and communicate data over the Internet.

device side, where object and protocol processing, which consumes resources heavily, is delegated to the gateway. The IoT gateway acts as an intermediary between the IoT device and the Internet [6]. The IoT gateway, shown in Figure 1, can take the form of a microcomputer, router, smart phone, or any device with ample resources to conduct TLS-based secure communication. In this research, a device-to-gateway layered security architecture has been designed and developed by implementing an Advanced Encryption Standard (AES) for the data object within the IoT device [15, 16, 17]. The extra device-to-gateway security layer has been created by employing the standard wireless security mechanism for IoT device authentication.

### 1.1 Paper contribution

The major contribution of this research is that it offers a complete security mechanism for the resource-constrained IoT devices. The security mechanism spans the IoT device, the IoT gateway, and the remote Internet server. The contribution comprises the design and implementation of a symmetric encryption of data objects at the IoT device over the native wireless security. We are thereby able to create a two-layer device-gateway security architecture. The implementation of a TLS-based security at the gateway works on standardly secure data objects before it travels over the Internet [6, 18]. The server has been configured to accept, process, and extract the data from the IoT gateway in its new format.

### 1.2 Paper structure

The rest of this paper is structured as follows: Section 2 provides background information on the IoT system, the problem description, and the related work. In section 3, the proposed security mechanism is theoretically explained in terms of the design requirements and interactions between IoT components. Section 4 is dedicated to demonstrating the implementation phase of the solution and the experimental setups. The performance evaluation of the proposed mechanism is documented in Section 5. Conclusions and future works are discussed in Section 6.

## 2 Preliminaries

IoT technology has developed in recent years to include more and more devices adopting embedded systems and communication interfaces. The future growth of IoT deployments comprises healthcare, education, manufacturing, and transportation. The main concept behind IoT devices is the possibility of collecting and sending information over the Internet [4, 11]. The architecture of the IoT components can be divided into three layers: the perception layer (physical devices), network layer (transmission layer), and application layer [7, 19]. However, each layer has its own security needs. This paper focuses on the perception layer for securing the entire data path. Figure 2 represents the layered architecture of IoT components and the data networks.

Constrained IoT devices can be grouped based on the available resources into three categories: Class-0 (C0), Class-1 (C1), and Class-2 (C2) devices. A comparison of the available resources in every category is shown in Table 1 [6]. It is apparent that the Class-0 devices have much fewer resources in terms of RAM and ROM memories. Furthermore, the available RAM size is not able to handle intensive security mechanisms.

IoT security needs to cover the entire IoT hierarchal architecture. IoT security spans the application layer, network layer, and perception layer. The basic security concerns include data confidentiality, integrity, and availability [7, 19, 20]. Constrained IoT devices have limited resources and therefore are limited to the protocols and standards they can support [21]. Efforts have been made by groups such as the Internet Engineering Task Force (IETF) to develop protocols and standards more suited for constrained environments, such as Datagram Transport Layer Security (DTLS)

Table 1: A comparison of the available resources in the categories of the constrained IoT devices [6].

| | RAM (Data size) | ROM (Code size) |
|---|---|---|
| Class-0 (C0) | $\ll$ 10KB | $\ll$ 100KB |
| Class-1 (C1) | $\sim$ 10KB | $\sim$ 100KB |
| Class-2 (C2) | $\sim$ 50KB | $\sim$ 250KB |

and Constrained Application Protocol (CoAP), by increasing the efficiency and minimizing the required computing resources [22, 23].

The security of the transport layer for Class-1 and Class-2 IoT devices can be achieved using DTLS over HTTP or CoAP. DTLS is an adaptation of the TLS protocol and has a heavy resource footprint in addition to existing application code in the device itself [14]. Like HTTP, CoAP as a stand-alone protocol does not contain security features necessary for secure data communication [24]. In order to fix this issue, a variation of TLS was developed to run under CoAP and over UDP called (DTLS) [22]. DTLS contains many features of TLS such as data encryption and authentication, with added features to deal with the unreliability of UDP [23]. Recently, CoAP over DTLS has been termed as CoAPS.

Despite the efforts of the IETF group, there is still a range of devices that fall short of the minimal resources needed to support such technologies on top of existing applications. These devices are known as "Class-0" as they fall short of the minimum threshold (10 KB of RAM and 100 KB of ROM) to support secure communication using TLS-based solutions [6].

The minimal code size and memory consumption for using DTLS were presented by Kumar et al. [14] in the DTLS implementation guide. The memory requirements outlined in the report suggest that the minimum resource requirements for DTLS (3.9 KB of RAM and 15.15 KB of ROM) would not be feasible in most Class-0 devices. It may also perform poorly on some Class-1 devices with connection times as slow as 24 seconds for a secure transmission [25]. As a conclusion, an alternative security solution is required for highly constrained IoT devices, especially for Class-0 IoT devices.

Doukas et al. [18] have attempted to secure data communication from constrained IoT medical devices by deploying IoT security in the gateway as an intermediary between the device and the Internet. This developed security solution secures data communication over the Internet by applying Public Key Encryption (PKI) and Secure Socket Layer (SSL) at the gateway. Although the solution presented in [18] focuses on the communication between the gateway and the Internet, the IoT system is still susceptible to attacks and data interception between the device and the gateway.

A solution offered by Vučinić et al. [26] was designed for more resource-heavy devices (C1 and C2). This offered solution uses DTLS-based security, and it applies a data object encryption inside a data transmission payload. The security of data objects is provided with symmetric encryption by way of an extra layer of protection for data communication. Although object layer security on its own does not offer effective security, it may be possible to add it to other security mechanisms for stronger security solutions.

Existing research addresses different challenges of secure data communication in Class-0 devices, but no single solution can be considered as a comprehensive solution that aims to secure the data path through all the IoT system components represented in Figure 2 Moreover, most of the available solutions do not target Class-0 devices. A security mechanism similar to that developed by Doukas et al. [18] provides a good base for securing IoT devices. However, it does not cover the entire data path, and it leaves a security gap between the device and the gateway. A comparison of some available solutions is presented in Table 2.

Driven by the demand for a comprehensive security solution to Class-0 IoT devices, this paper presents a complete and distributed security mechanism for these devices. Data encryption is one of the security requirements in the perception layer [21]. The novelty of the proposed solution is three-fold: the presented security mechanism focuses on the perception layer and aims to secure the entire data path from Class-0 IoT devices to the Internet; the distribution of the proposed solution over the IoT device, the gateway, and the remote server; and the provided multi-layer security between the IoT device and the gateway. By adding an extra layer of encryption at the object layer, message content can be protected inside the local network and at the gateway until it can be securely transferred over the Internet to its final destination.

## 3 A distributed security mechanism

This section focuses on the conceptual design of a distributed security mechanism. The design covers three IoT systems components: the IoT device, IoT gateway, and remote web server. Each component is discussed in detail along with a description of how the data are communicated. The design of the proposed security mechanism aims to achieve the requirements for Class-0 devices that are documented in Table 3.

The proposed solution secures data communication in Class-0-constrained devices by applying a 128-bit symmetric encryption (AES-128) to data objects, such as sensor readings, before they are transmitted between the device

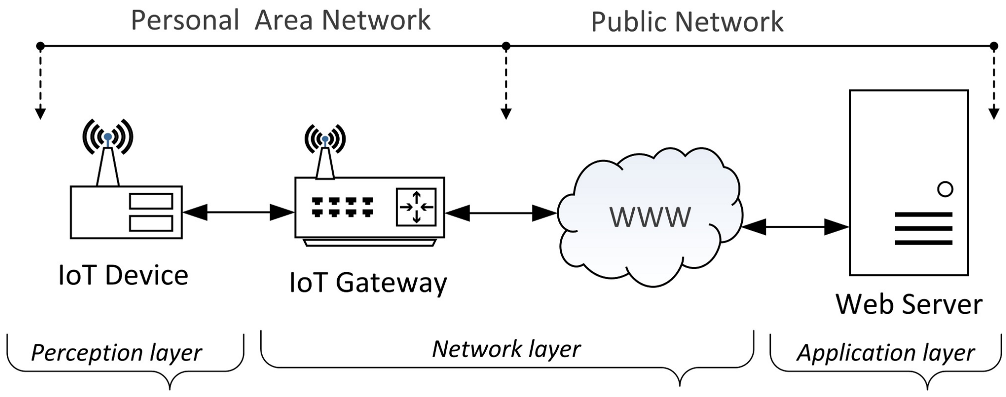


Figure 2: The three main layers of the layered architecture of IoT components. From the networking viewpoint, IoT devices and IoT gateway fall into a Personal Area Network (PAN), whereas the web server falls into a Public Network (PN).

Table 2: A comparison of some available security solutions for IoT devices.

| | The concept | The drawback for Class-0 devices |
|---|---|---|
| Doukas et al. [18] | Enabling data protection through PKI encryption | Does not secure the device to the gateway |
| Rescorla et al. [22] | Datagram Transport Layer Security V1.2 | Very heavy resource requirements |
| Vučinić et al. [26] | Object Security Architecture for the IoT | Very heavy resource requirements |

Table 3: The design requirements for the proposed distributed security mechanism.

| | Requirement |
|---|---|
| #1 | Provide data security between the Class-0 device and the IoT gateway |
| #2 | Secure data transported between the IoT gateway and the Internet |
| #3 | Perform efficiently with minimal resource consumption |

and the gateway. The data are formatted in JavaScript Object Notation (JSON) and are sent as a CoAP or HTTP POST to the gateway. The data object is encrypted using a secret key and can only be decrypted by devices with the same key. This key is shared with the destination, in this case, the web server.

Wireless transmissions in the LAN/PAN between the device and the gateway are secured at the Data Link Layer using a wireless interface module. Constrained wireless standards such as IEEE 802.15.4 and protocols such as Low power Wireless Personal Area Network (6LoWPAN) [27] are capable of supporting AES 128-bit symmetric encryption at this layer. By using an offered Pre-Shared Key (PSK) to encrypt wireless transmissions, only authorized devices connected to the network can receive traffic. By encrypting data objects at the device level (perception layer), only the device and the final destination will be able to read the encrypted data. An overview of the proposed security mechanism is represented in Figure 3. Further descriptions of the proposed distributed security mechanism on each IoT system component are provided in the following paragraphs.

## 3.1 Device-to-Gateway security

From the communication standpoint, another level of security between the IoT device and the gateway can be achieved using hardware-based symmetric encryption of the Data Link Layer (DLL) as part of the wireless protocol (e.g., IEEE 802.15.4, IEEE 802.11n). Wireless transmission can be provided using an IEEE 802.15.4 module such as a ZigBee or 6LoWPAN interface. When connecting to a network, devices are secured with a PSK, which is installed on each authorized device, and it is required for communication initiation between the gateway and the constrained devices in the network. Any unauthorized devices monitoring the traffic will not be able to decrypt data without the correct PSK. However, the built-in wireless security protects data from entities without the PSK, leaves data exposed if someone manages to compromise the wireless security, or capture the PSK from another device or from the gateway.

Confidentiality is assured between the IoT device and the destination by encrypting data at the object level. Object layer security exists at the application layer inside the payload of a transmission packet. Objects in this context

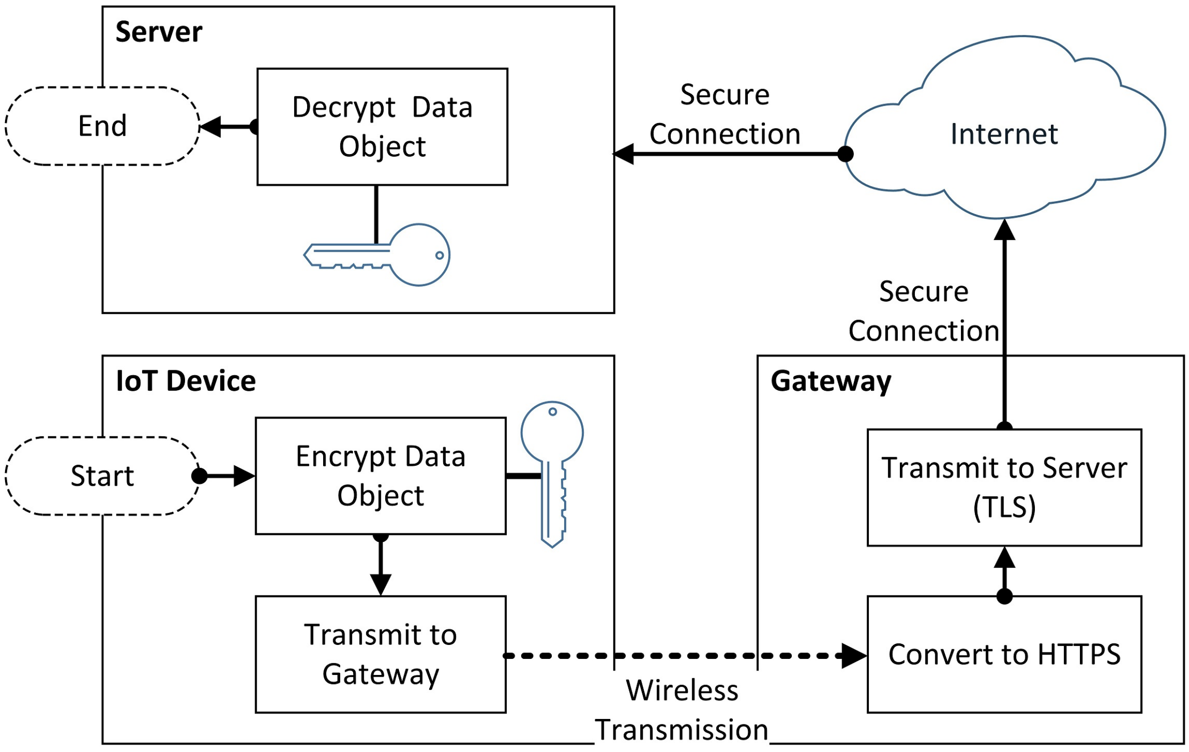


Figure 3: An overview of the proposed security mechanism shows the major processes that run on each component. The figure also represents the data connections and transmissions between the three IoT system components.

refer to a container of information, which has been formatted to be human readable. Different data formats exist for the web, including JSON, XML, and YAML. It is worth noting that object-layer security applies cryptography to a data object, but the header information such as the source and destination addresses remain exposed. The packet format and data encryption are shown in Figure 4. This level of encryption is used as a primary layer of protection, and it can be combined with the offered wireless security for stronger security between the IoT device and the IoT gateway. It works as a second defensive wall in case of a compromised wireless network.

Figure 4 depicts the two layers of security applied to data transmitted from the device to the gateway. Security is applied at the Data Link layer in the form of hardware-based AES encryption secured with a PSK. The second layer of security is applied only to the contents of the data object. Addressing and source information remain unencrypted in this layer. The data object is encrypted with a symmetric key, which has only been shared with the server so that no intermediaries will be able to decrypt the data.

## 3.2 Gateway-to-Internet security

IoT gateways are computational devices with enough resources to run operating systems and protocols necessary to securely transfer traffic across the Internet. An IoT gateway may take the form of a microcomputer with a Linux-based operating system. The gateway has sufficient resources to apply heavy security and communication protocols that cannot be supported by Class-0 devices. Once data are received by the gateway, they are processed into HTTPS and prepared for transmission to the remote server. The gateway is configured with Secure Socket Layer (SSL) tools, which are used to create a secure HTTPS connection between the gateway and the server. From the gateway point, one can forward secure communications to the server over the Internet using the configured secure socket layer.

The gateway acts as an intermediary with ample resources to support these security measures and secure data before sending it over the Internet. Data sent from the IoT device will be sent to the gateway using protocols such as CoAP and HTTP and sent across the Internet using HTTPS (HTTP over TLS) to the web server. In the proposed security mechanism, the payload of the packets is formatted as a JSON object and encrypted using AES 128-bit or 256-bit symmetric encryption. This data object will exist inside the transmission payload, while the packet header information such as source and destination address remains unencrypted, as demonstrated in Figure 4.

The JSON object is not readable by the gateway or any other intermediary entity other than the intended destination. Similarly, if the server sends a command back to the device, the data object is encrypted using the pre-shared symmetric key and is forwarded to the device for decryption. Security is applied at the Data Link Layer in the form of hardware-based AES encryption secured with a PSK. Only authorized devices should be in possession of

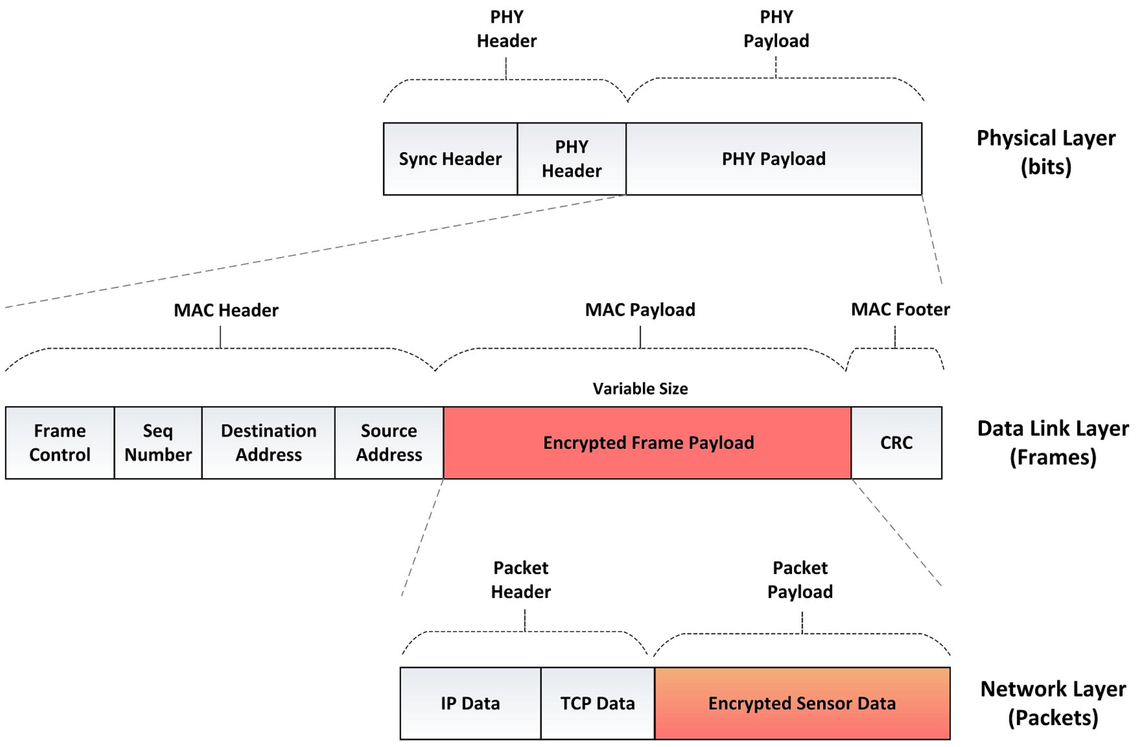


Figure 4: The utilized packet format ,which represents the packet header, the packet payload, and the encrypted part of the packet. The packet formats in the physical layer, the data link layer, and the network layer are represented.

the PSK. The second layer of security is applied only to the contents of the data object. Addressing and source information remain unencrypted in this layer. The data object is encrypted with a symmetric key, which has only been shared with the server, so that no intermediaries will be able to decrypt the data.

## 3.3 Web server security

The messages being transmitted to the server are encrypted with the server's public key, which is installed in the gateway. Only the server can decrypt messages using its corresponding private key. The private key is located on the server and is not shared with any other devices. The detailed flowchart of the proposed security mechanism with all sequential processes that are mapped to the three IoT system components is shown in Figure 5.

Once the HTTPS packets are received by the server, they are decrypted using the private key. The encrypted data object can then be decrypted using the symmetric secret key from the originating device, in this case, our class-0 IoT device. If the key is only present on one IoT device and the server, it can be used to authenticate data received from either party. If the key is shared with multiple devices, the devices are authenticated as part of a group. This scenario maintains the confidentiality of IoT data whenever it passes over a public network.

## 3.4 Advanced encryption standard

Advanced Encryption Standard (AES) is one such symmetric standard, which operates at fast speeds and requires fewer resources than DTLS, making it very suitable for Class-0 devices [14]. AES can be easily implemented and optimized on hardware. AES inputs data as 16-byte (128-bit) blocks that are then encrypted using a cryptographic key that is either 128 bits, 192 bits, or 256 bits in size [28]. The larger the key size, the greater the security and resource requirement for the device to encrypt and decrypt. Symmetric encryption can be applied at different layers of the communication stack such as the data link layer (e.g., wireless transmissions) and to specific objects of data within a message such as sensor readings. AES is suitable for the needs of Class-0 IoT devices in terms of the encryption speed and the required resources.

Symmetric cryptography involves encrypting data with a single encryption key, which is shared between multiple devices. Any device that possesses the key can decrypt data that have been encrypted with the same key. When the key is shared with other devices, there is a higher risk that it may fall into the wrong hands, and therefore, it must be kept safe.

Currently, in the proposed solution, the IoT data are encrypted in the IoT device using a symmetric key. The symmetric key is static and is installed only on the IoT device and the server. Thus, the gateway is not able to decrypt the packet payload. Messages being transmitted from the

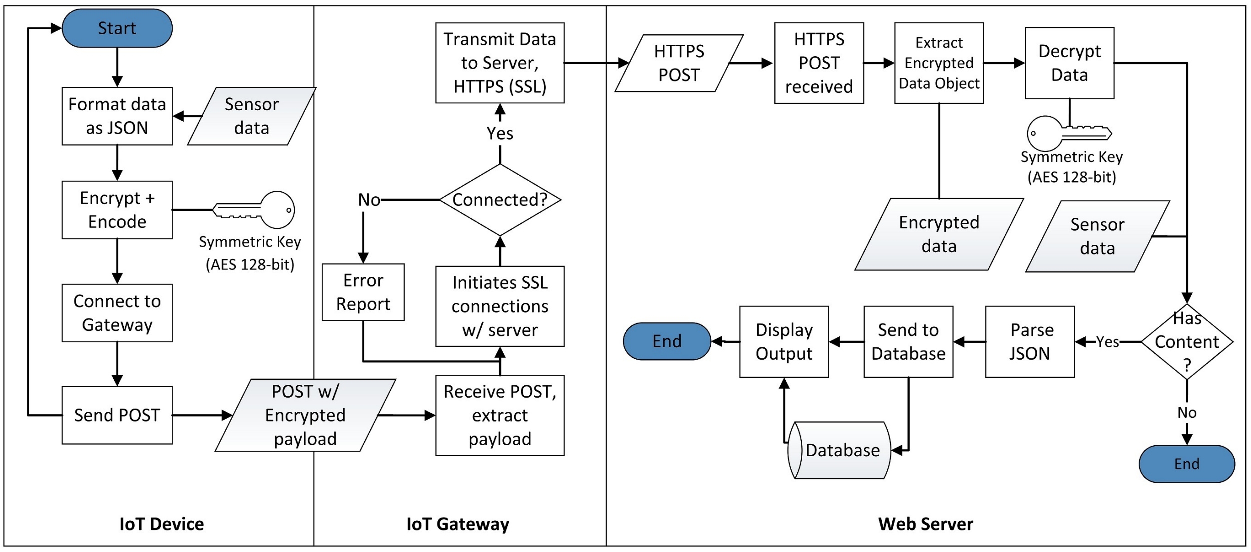


Figure 5: A full flowchart of the proposed security solution across the three IoT system components.

gateway to the server are encrypted with the server public key, which is installed in the gateway. Only the server can decrypt messages using its corresponding private key. The private key is located on the server and is not shared with any other device. An asymmetric key cryptography approach is used between the gateway and the server due to the plethora of computing capabilities.

## 4 Implementation flow

The distributed security mechanism has been implemented using real-time hardware configurations. This section describes the implementation, hardware specifications and configurations of the three IoT system components.

### 4.1 IoT device setup

For the hardware underlying the IoT device, an Arduino Uno microcontroller was used with an additional Ethernet shield added for connectivity. A wireless shield has been used as an alternative, but for the proof of concept, the Arduino wa connected directly to a wireless router via an Ethernet cable. A “DHT11” temperature and humidity sensor was connected to the Arduino. The Arduino hardware set up is shown in Figure 6 (a). The Arduino connects to and reads data from the sensor and then parses the data into the JSON format before encryption. The device automatically begins the sensor reading process when the device is connected to a power source, and it continues to repeat the process until the power is disconnected.

The temperature data are parsed as JSON and padded to 16 bytes, as this is the required block size for AES. The data are then encrypted using an AES-128 encryption library. The encrypted output may contain special characters, which are not web-friendly or human readable; therefore, it is encoded using the Base64 [29] character set so that it is easier to transmit to the remote server.

A web client has been prepared and installed on the Arduino in order to establish a connection to the IoT gateway. Once a connection is established, the Arduino uses a POST method to send data to the gateway via HTTP. The encrypted data are added to the contents of the HTTP POST before being sent. As soon as the POST message is sent, the Arduino receives a response back from the gateway confirming that the POST was received, waits for a period of time, and then restarts the processes from the beginning. Naturally, the confirmation back from the gateway to Arduino improves the reliability of the connection between the two terminals.

Through the formulation of the POST in the Arduino code, the header information is coded with the destination IP address and web service address “index.php". The encrypted sensor data are added to the contents of the post through the variable “dataEncoded”. If an error is received while attempting to connect to the gateway, the response is read when the POST reaches the gateway. If no errors are received, the connection is established and the packet is sent.

Figure 7 represents a captured TCP packet after it has been transmitted from the sensor (IoT device) and reassembled by the gateway. The packet includes the header information such as the destination and source address. It also includes the encrypted sensor data in the POST contents (i.e., “ZGioFzoApFk9CfV9XFQhxQ==").

### 4.2 IoT gateway setup

The IoT gateway was built on a Raspberry Pi (RPi) model B. The gateway hardware is shown in Figure 6 (b). The RPi is a microcomputer with ample resources to perform the heavier security processes that are too resource-intensive for the IoT device. The RPi contains a 700-MHz CPU, 512

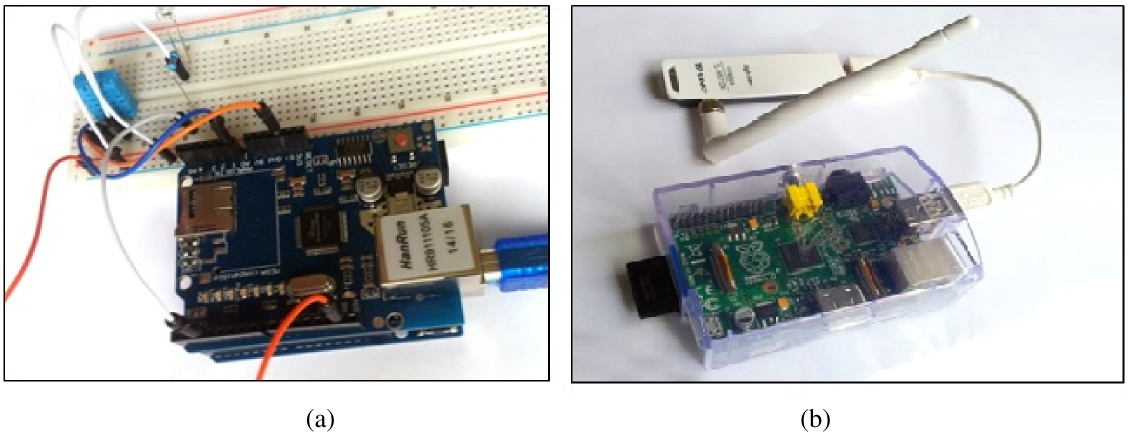


Figure 6: The hardware set up for the implementation of the proposed security mechanism. (a)– The setup of an IoT device using Arduino hardware and (b)– The setup of an IoT gateway with a wireless antenna (Raspberry Pi setup).

MB, and a SD card reader that acts as its storage memory. In this case, an 8GB SD card was used for storage. The RPi can be configured with a range of Linux-based operating systems. The RPi connects to the wireless router through a wireless USB adapter. The RPi was installed with a PSK to access the wireless network that is secured with AES 256-bit symmetric encryption.

A web application running on an Apache web server was installed on the RPi to receive and process data from the IoT device. When a POST is received from the Arduino, the encrypted payload (sensor data) is stripped. The gateway does not contain the symmetric key to decrypt data from the Arduino; however, it forwards it to the server over a secure connection.

The RPi (IoT gateway) connects to the server using a SSL connection and posts the data to the server in an HTTPS POST. For testing purposes, the security certificate was not signed by a certificate authority, and therefore, when the IoT gateway attempted to connect to the server, the verification of the certificate with a trusted third party was disabled in the code (VERIFYPEER and VERIFYHOST). In a real environment, this would be unsafe, and by disabling the verification, the gateway would not be able to ensure that the connection has not been tampered with.

### 4.3 Web server setup

For testing purposes, the server was set up on a laptop within the local area network. This server represents the online server to which data would be transmitted. An Apache web server was installed and configured on the laptop. A security certificate was created, and the server was set up to receive HTTPS connections using SSL/TLS. As soon as the connection is established by the gateway, a HTTPS POST will be sent to the remote server carrying the encrypted data.

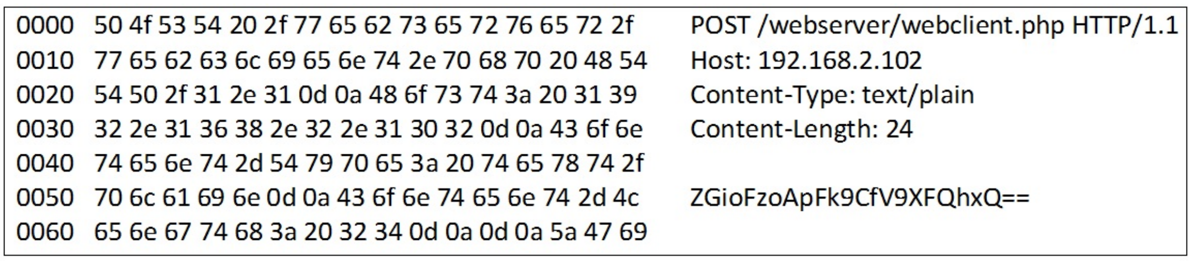
On the server side, a web service that handles the decryption process was installed. The cipher text and the symmetric key are passed to the service. The cipher text is decoded from base64 [29] into its original encrypted form. It is then processed using a "rijndael-128" cipher, which is another reference for AES-128. The final stage of the process is to parse the decrypted output and upload it to a database along with the date and the original encrypted message for reference. A web page was also created to demonstrate the working solution. The web page allows the user to view the latest sensor results, which are stored in the database.

## 5 Performance Evaluation

The proposed security mechanism has been evaluated based on its performance and ability to meet the outlined requirements in Section 3 In addition, the proposed security solution should perform in a timely manner and not be subject to an unacceptable amount of packet loss or failure. The solution is designed to support Class-0 devices with respect to resource consumption and processing time.

With AES, the data are passed to the algorithm in 16-byte blocks. If the input is larger than 16 bytes, it is divided into subsequent blocks. If a subsequent block falls short of the 16 bytes, padding is applied to increase the size of the data to 16 bytes. During the test, a small single-line JSON string was created with a temperature reading from the sensor. The result was 12 bytes in size, and 3 bytes of padding were added to the string before it was encrypted and then encoded using Base64 encoding scheme.

Using cryptography requires additional resources from the device. The performance measurements are recorded in Table 4; they satisfy the design requirements in Section 3. The requirements may change depending on the application of the device and the nature of its constraints. It is assumed that a resource overhead of 0.5 KB of RAM and 0.47 KB of ROM with an additional processing time of 0.46 seconds is an acceptable burden on most Class-0 systems. When the key size was increased to 256 bits, there was a 25% increase in processing time to 0.57 seconds and a 1% increase in ROM usage to 0.63 KB. For most applications,



```
0000  50 4f 53 54 20 2f 77 65 62 73 65 72 76 65 72 2f    POST /webserver/webclient.php HTTP/1.1
0010  77 65 62 63 6c 69 65 6e 74 2e 70 68 70 20 48 54    Host: 192.168.2.102
0020  54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 31 39    Content-Type: text/plain
0030  32 2e 31 36 38 2e 32 2e 31 30 32 0d 0a 43 6f 6e    Content-Length: 24
0040  74 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f
0050  70 6c 61 69 6e 0d 0a 43 6f 6e 74 65 6e 74 2d 4c    ZGioFzoApFk9CfV9XFQhxQ==
0060  65 6e 67 74 68 3a 20 32 34 0d 0a 0d 0a 5a 47 69
```

Figure 7: A captured packet sent from the IoT sensor. The captured packet clarifies the encrypted and not-encrypted data from the sensor. The capturing probe is installed at a point between the device and the gateway.

the increases would be acceptable and provide stronger encryption as a result.

The implemented layered security provides strong circumvention against any external attack to the IoT system. An attacker would first need to gain access to the network either through direct access to the gateway or with a PSK for the network to be able to capture the data. With the additional encryption applied to data objects, even if the attacker had access to the network or the gateway, the attacker would not be able to read the data without the cipher key.

The memories overhead with respect to RAM and ROM in Table 4 are very low compared to the solutions offered in the literature. According to the implementations of two security solutions in [24], in particular, the encryption in the Host Identity Protocol (HIP) consumes 1.7 KB of ROM, and the encryption in the DTLS imposes an overhead of 3.3 KB of ROM and 1.5 KB of RAM. This confirms the applicability of our proposed solution for the Class-0 IoT devices.

At the gateway, the data are processed into HTTPS using RSA 2048-bit and a session key and securely forwarded to the web server. Using the network protocol analyzer tool, we can analyze the traffic exchange between the server and gateway. The received data are encrypted using a session key, and hence, they are not readable by any unauthorized user eavesdropping on traffic via active or passive traffic collection mechanisms [30, 31].

The processing times in Table 4 were recorded on the device (encryption time), on the gateway (object processing time), and on the server (decryption time). Due to the constrained processing power, the encryption time varies from AES-128 to AES-256. However, the processing time is constant on the gateway because the gateway is blind to the message contents. The gateway translates a message from HTTP to HTTPS and forwards it to the server.

The reported processing times are faster than what is reported in the literature. For example, Doukas et al. [18] achieved an 0.8-second overhead on the gateway compared to 0.18 seconds for our security solution. While the processing times in [18] are slightly slower than ours, it is worth noticing that they used a larger data size of "Less than 100 KB", whereas the message size used in this research is limited to 24 bytes.

## 6 Conclusions

Internet of Things (IoT) is a promising paradigm in wireless communications that offers a capability to connect, collect, and send data over the Internet. IoT keeps expanding with broad deployment demands in many fields such as home appliance, marketing, and healthcare. Despite the research attentions that IoT has received, the security, and hence, privacy issue in Class-0 devices is still a gap. This research has presented a distributed security mechanism for constrained Class-0 IoT devices. The design principle behind the proposed solution is to delegate the low resource consuming operations to the IoT device, and keep the high resource consuming processes at the IoT gateway side. In addition to the native wireless security, a layered security scheme has been offered by performing a asymmetric encryption to the data objects at the device level. The implementation of the distributed security mechanism has included the IoT device, the IoT gateway, and the server side. A complete laboratory setup for IoT infrastructure has been developed for the implementation and the evaluation purposes. Our experimental works have proven the security level of the solution, the suitability of the security mechanism to the Class-0 devices. In the worst case, with AES-256, the encryption process consumes memory overhead of 0.5 KB of RAM, 0.63 KB of ROM, 0.57 second encryption time on the device, and 0.18 second on the gateway. The future work focuses on the distribution and management of the encryption key, bring into attention additional security aspects such as data integrity and availability for improving the overall system's performance and circumvention.

## References

[1] Kramp, T., van Kranenburg, R., Lange, S.: Introduction to the internet of things. In: Bassi, A., Bauer, M., Fiedler, M., Kramp, T., van Kranenburg, R., Lange, S., Meissner, S. (eds.) Enabling Things to Talk, pp. 1–10. Springer Berlin Heidelberg (2013)

[2] Medaglia, C.M., Serbanati, A.: An overview of privacy and security issues in the internet of things. In: Giusto, D., Iera, A., Morabito, G., Atzori, L. (eds.) The Internet of Things, pp. 389–395. Springer New York (2010)

Table 4: The memory overhead (RAM and ROM) and the processing times for the proposed distributed security mechanism.

| | Memory consumption (KB) | | Processing time (Second) | | |
|---|---|---|---|---|---|
| | RAM (Device) | ROM (Device) | IoT (Device) | IoT (Gateway) | Server |
| No encryption * | 16 | 0.5 | – | – | – |
| AES-128 ** | 0.5 | 0.47 | 0.46 | 0.18 | 0.000205 |
| AES-256 ** | 0.5 | 0.63 | 0.57 | 0.18 | 0.000404 |

*Base memory is considered. ** Overhead memory is considered.

[3] Lee, G.M., Crespi, N., Choi, J., Boussard, M.: Internet of things. In: Bertin, E., Crespi, N., Magedanz, T. (eds.) Evolution of Telecommunication Services, Lecture Notes in Computer Science, Vol. 7768, pp. 257–282. Springer Berlin Heidelberg (2013)

[4] Khan, R., Khan, S., Zaheer, R., Khan, S.: Future internet: The internet of things architecture, possible applications and key challenges. In: 10th International Conference on Frontiers of Information Technology (FIT). pp. 257–260. IEEE (2012)

[5] Höller, J., Tsiatsis, V., Mulligan, C., Karnouskos, S., Avesand, S., Boyle, D.: From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence. Elsevier, 1st edn. (2014)

[6] Bormann, C., Ersue, M., Keranen, A.: Terminology for constrained-node networks, RFC 7228, (2014), `http://www.rfc-editor.org/info/rfc7228`, last access 29.01.2016

[7] Zhao, K., Ge, L.: A survey on the internet of things security. In: 9th International Conference on Computational Intelligence and Security (CIS). pp. 663–667. IEEE (2013)

[8] Jing, Q., Vasilakos, A., Wan, J., Lu, J., Qiu, D.: Security of the internet of things: perspectives and challenges. Wireless Networks 20(8), 2481–2501 (2014)

[9] Lack of security in internet of things devices. Network Security 2014(8), 2 – (2014)

[10] Weber, R.H.: Internet of things – New security and privacy challenges. Computer Law & Security Review 26(1), 23–30 (2010)

[11] Santos, A., Macedo, J., Costa, A., Nicolau, M.J.: Internet of things and smart objects for M-health Monitoring and control. Procedia Technology 16(0), 1351–1360 (2014)

[12] Storey, A.: There's nothing 'smart' about insecure connected devices. Network Security 2014(7), 9–12 (2014)

[13] Raza, S., Trabalza, D., Voigt, T.: 6LoWPAN compressed DTLS for CoAP. In: The 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS). pp. 287–289. IEEE (2012)

[14] Kumar, S., Keoh, S., Tschofenig, H.: A hitchhiker's guide to the (datagram) transport layer security protocol for smart objects and constrained node networks (2013), `https://tools.ietf.org/html/draft-ietf-lwig-tls-minimal-00`, last access 29.01.2016

[15] Stallings, W.: Cryptography and Network Security: Principles and Practice. Pearson Education, NJ, USA (2002)

[16] Paar, C., Pelzl, J.: The advanced encryption standard (AES). In: Understanding Cryptography, pp. 87–121. Springer Berlin Heidelberg (2010)

[17] Fathy, A., Tarrad, I., Hamed, H., Awad, A.I.: Advanced encryption standard algorithm: Issues and implementation aspects. In: Hassanien, A.E., Salem, A.B.h., Ramadan, R., Kim, T.h. (eds.) Advanced Machine Learning Technologies and Applications, Communications in Computer and Information Science, Vol. 322, pp. 516–523. Springer Berlin Heidelberg (2012)

[18] Doukas, C., Maglogiannis, I., Koufi, V., Malamateniou, F., Vassilacopoulos, G.: Enabling data protection through PKI encryption in IoT m-health devices. In: The 12th IEEE International Conference on Bioinformatics Bioengineering (BIBE). pp. 25–29. IEEE (2012)

[19] Sun, X., Wang, C.: The research of security technology in the Internet of Things. In: Jin, D., Lin, S. (eds.) Advances in Computer Science, Intelligent System and Environment, Advances in Intelligent and Soft Computing, Vol. 105, pp. 113–119. Springer Berlin Heidelberg (2011)

[20] Yang, X., Li, Z., Geng, Z., Zhang, H.: A multi-layer security model for internet of things. In: Wang, Y.,

Zhang, X. (eds.) Internet of Things, Communications in Computer and Information Science, Vol. 312, pp. 388–393. Springer Berlin Heidelberg (2012)

[21] Suo, H., Wan, J., Zou, C., Liu, J.: Security in the internet of things: A review. In: IEEE 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE). Vol. 3, pp. 648–651. IEEE (2012)

[22] Rescorla, E., Modadugu, N.: Datagram transport layer security version 1.2, RFC 6347, (2012), https://tools.ietf.org/html/rfc6347, last access 29.01.2016

[23] Shelby, Z., Hartke, K., Bormann, C., Frank, B.: The constrained application protocol (CoAP), RFC7252, (2014), https://tools.ietf.org/html/rfc7252, last access 29.01.2016

[24] Garcia-Morchon, O., Keoh, S.L., Kumar, S., Moreno-Sanchez, P., Vidal-Meca, F., Ziegeldorf, J.H.: Securing the IP-based internet of things with HIP and DTLS. In: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks. pp. 119–124. WiSec '13, ACM (2013)

[25] Keoh, S., Kumar, S., Garcia-Morchon, O.: Securing the ip-based internet of things with DTLS (2013), https://tools.ietf.org/html/draft-keoh-lwig-dtls-iot-02, last access 29.01.2016

[26] Vučinić, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., Guizzetti, R.: OSCAR: Object security architecture for the Internet of Things. Ad Hoc Networks 32(0), 3–16 (2015), Internet of Things security and privacy: Design methods and optimization

[27] Kolahi, S., Li, P., Argawe, M., Safdari, M.: WPA2 security-bandwith trade-off in 802.11n peer-peer WLAN for IPv4 and IPv6 using Windows XP and Windows 7 operating systems. In: The 7th IEEE Symposium on Computers and Communications (ISCC). pp. 575–579. IEEE (2012)

[28] Elfatah, A.F.A., Tarrad, I.F., Awad, A.I., Hamed, H.F.A.: Optimized hardware implementation of the advanced encryption standard algorithm. In: 8th International Conference on Computer Engineering Systems (ICCES). pp. 197–201. IEEE (2013)

[29] Coles, M., Landrum, R.: SQL CLR Cryptography. In: Expert SQL Server 2008 Encryption, pp. 167–184. Apress (2009)

[30] Rubio-Loyola, J., Sala, D., Ali, A.: Maximizing packet loss monitoring accuracy for reliable trace collections. In: 16th IEEE Workshop on Local and Metropolitan Area Networks, LANMAN 2008. pp. 61–66. IEEE (2008)

[31] Rubio-Loyola, J., Sala, D., Ali, A.: Accurate real-time monitoring of bottlenecks and performance of packet trace collection. In: 33rd IEEE Conference on Local Computer Networks, LCN 2008. pp. 884–891. IEEE (2008)

# Drupal 8 Modules: Translation Management Tool and Paragraphs

Saša Nikolić
Faculty of Mathematics, Science and Information Technologies, University of Primorska
Glagoljaška 8, SI-6000 Koper, Slovenia
E-mail: nikolic.sasa09@gmail.com

Jurij Šilc
Computer Systems Department, Jožef Stefan Institute
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
E-mail: jurij.silc@ijs.si

**Technical paper**

**Keywords:** content management system, Internet, website, open source, module, Translation Management Tool, Paragraphs

**Received:** September 11, 2015

*As the Web has grown in the last few decades, we now have nearly one billion websites online and most of them offer rich information, that is usually difficult to manage by normal users. In order to simplify the process of creating and managing website content with relative ease and with an user friendly experience, lots of content management systems were developed. These are software applications that provide capabilities for multiple users with different permission levels to seamlessly create, edit, review and publish website content. They offer a web-based graphical user interface, enabling publishers to access the content management systems online using only a web browser. Because of good security, quality, customisability and great support by the community of developers, open source content management systems are becoming extremely popular and many of them available on the market. The most noticeable are Drupal, Joomla, Magento and Wordpress. A lot of different surveys have been done to determine which one of them is the best, but this question still remained unanswered. Because of our personal involvement into developing Drupal and helping the community, this article describes a totally new version of Drupal, named Drupal 8. Firstly, a short overview about Drupal and all its key parts is presented, followed by a chapter describing all the new features and the current status. Two subtopics of this article will include modules that we lately contributed to – Translation Management Tool and Paragraphs. Some of our main issues are described at the end of each module.*

*Povzetek: V zadnjem desetletju smo priča bliskovitemu napredku spletnih tehnologij in posledično spletnih strani ter aplikacij. Te so danes polne bogatih informacij, a jih je navadno težje upravljati. Za poenostavitev kreiranja in upravljanja s podatki na spletnih straneh so bili razviti različni sistemi za upravljanje vsebin. Danes jih je na tržišču veliko, med njimi so najbolj opazni Drupal, Joomla, Magento in Wordpress. V sestavku je govor o novi različici odprtokodnega sistema za upravljanje vsebin Drupal 8, pri razvoju katerega smo sodelovali. Najprej je na splošno predstavljen Drupal z glavnimi lastnostmi in funkcijami, nato sledi poglavje Drupal 8, kjer so opisane novosti in posodobitve. Po kratkem poglavju o trenutnem stanju sistema ter prihodnjih korakih sledi poglavje o dveh modulih, pri razvoju katerih smo sodelovali: Translation Management Tool, za lažje prevajanje vsebin, in Paragraphs, za boljše strukturiranje vsebin. Nekaj naših rešitev je opisanih na koncu vsakega od obeh modulov.*

## 1 Introduction

The Internet is probably one of the most profound achievements in human history. We became so addicted to it, that we hardly even notice it, unless it happens to be unavailable. With the help of great innovative technologies, the internet has simply dropped the barriers of time and geographical distance to turn the entire world into a local community centre. In the last decade millions of people tend to share their lives and experiences with others through their personal blog sites. Others use the web to show off their work, art or music. Still others found an opportunity to promote their companies and be more noticeable to a wider variety of people. Whatever the need is, there exists a great solution that for its installation and administration does not require any programming skills. Drupal is one of the most common solutions for building anything from simple user blogs to fully-customizable, interactive and mobile-responsive websites in several languages. Assuming that you are interested in the open source community and in learning more about Drupal, this article briefly describes what Drupal is and mainly, introduces you to the new version, Drupal 8. Last, but not least, it describes two new modules, Translation Management System and Paragraphs, their structure and my personal contribution to in their development. We have been an active Drupal member for more than 7 months now and contributed to more than 80 various issues, from core to contributed modules.

# 2 Drupal

## 2.1 What is Drupal?

Drupal was developed and released in 2001 under the open GNU General Public Licence (GPL), which means anyone is free to download it and share it with others. It is a PHP and MySQL-based system for managing websites and is used by hundreds of thousands of web developers around the world [3]. It serves as a back-end framework for more than 2% of all websites worldwide – from personal blogs to corporate, political and government sites, including the official website of the White House and various UK Government projects [2].

The default release of Drupal, known as Drupal core, contains only basic features, like user account registration and maintenance, menu management, taxonomy, page layout customization and system administration. This is enough for a simple website, a user blog or an Internet forum.

Currently, there are more than 30,000 free community-contributed add-ons, also known as contributed modules [4]. By adding different modules and features like advanced search, content translation, external text editors (WYSYWIG), many different jQuery libraries, etc. users can experience the web in totally new and different ways.

## 2.2 Why choose Drupal?

Here are some of the reasons why people may opt to use Drupal: Drupal is a CMS which allows users to update their websites without technical knowledge and ensures that it fits any organization's workflow. It is modular, extensible and scalable, which means it can grow over time as user's needs expands. Also, the website can be customized depending on the content or company features. Drupal can be installed in multiple languages, allowing users and administrators to view and administer the site in their own language. From the hardware point of view, it runs on any computing platform that supports a web server capable of running PHP (e.g., with Apache, Nginx, LiteSpeed) and a database to store content and configurations.

## 2.3 Core features

**Administer:** Drupal comes with various options for user accounts and permissions. The administrator can set up one or more roles to users, specifying with different permissions, what each user can and can not do.

**Build:** easily build websites without any programming knowledge. It also comes with pre-defined configuration, so that website building is much easier than before.

**Collaborate:** social publishing and interaction with the content on your site can be easily controlled by the administrator.

**Connect:** using aggregation, feeds, search engine connection capabilities and social media integration is widely supported to help users connect with wider audience.

**Creative Content:** Drupal's flexibility supports many content types including video, text, blog, podcasts and polls with an user-friendly web interface.

**Design & Display:** there are lots of themes created by professionals and free to use, since Drupals presentation layer allows designers to create their own interactive experiences that engage users and increase traffic.

**Extend** with more than 16,000 available modules developers can create and adapt the site to any requirements. Everyone is encouraged to contribute modules for others to use.

**Organize & Find:** many tools are available for organizing, structuring, finding and re-using website's content. Friendly path urls, custom lists, categorization with taxonomy and linking content with other content on the site are just some of the options.

## 2.4 Community

The main reason, why Drupal is more popular and secure than other open source CMS is a huge developer community. It counts more than 1 million members and provides support, constant development (and bug fixing), testing and documentation. Drupal community members make Drupal better and better every day. More than 950 people contributed code and ideas to the Drupal 7 release and even more are responsible for developing and maintaining the so-called "contrib" modules. The main website that provides a place for groups to organize, meet and work on various projects is `https://groups.drupal.org`. It is mainly based on geographic location and interest. This is a great way to get fast support, learn more by local people and to easily get involved. Drupal events and meet-ups are also very frequent, which makes it easier to exchange knowledge face to face, get ideas for new projects and making friends along the way. IRC is another fast and effective way of communication and interaction with other developers, mainly in use for support. Forums, mailing lists and social media are also available for sharing information about Drupal.

## 3 Drupal 8

Since its creation in 2001, Drupal has grown and developed year by year to meet new changing demands and needs of all its global users and to achieve that, new big, forward-looking changes needed to be made. The result is that Drupal has stayed relevant to new technologies, unlike nearly every other Open Source CMS over the years. The downside is that with every major release, Drupal developers have gone through a lot of pain adjusting to this changes. In 2006, the founder of Drupal – Dries Buytaert wrote: *"So let's capture that thought for future reference. Sweeping changes are required to make major advances in technology, and often times there is a lot of pain before the payoff."*[1]

Drupal 7 is a very popular CMS amongst users, but there are quite some big limitations, including incomplete Entity API, no separation between logic and presentation in the theme layer, and so on. Contributed modules tried to solve many of these problems, but they were mostly incomplete. With Drupal 8, these problems were solved with a head-on approach – through the Configuration Management Initiative, Twig templating layer and a new, complete Entity API.

With more than 200 new features and improvements, will definitely be the most significant update in Drupal history. Easier customizations of data structures, listings and pages will be the first thing that the users will notice at the beginning. Countless new capabilities for displaying data on mobile devices, building APIs and adapting the website to multilingual needs are also some of the other things that will make a huge impact on the usability and diversity. With a much more efficient core, easier migration from earlier versions and inline content editing tools it will become a cutting-edge platform that will set new standards for other CMS. To not forget various new modules and themes, made available by a modern Object Oriented Programming (OOP) approach on the back end side. All those new features can be summed up in different categories, but the categorisation based on what affects different types of users seems to be the most important one [7].

## 4 Drupal 8 development

One of the biggest challenges with Drupal, is that it is hard for organizations of all sizes to find Drupal talent (developers, themers, site builders, etc). Drupal 7 didn't address this problem (e.g., using procedural programming instead of object-oriented programming), and in fact made it a bit worse with the introduction of even more Drupal-specific development (e.g., excessive use of structured arrays). For most people new to Drupal, Drupal 7 could be really complex. The most effective way to address the Drupal talent issue, as well as the complexity issue, is to update Drupal with modern frameworks and platforms, so there is less Drupal-specific knowledge to learn in order to become proficient. Modern PHP concepts and standards, object-oriented programming, and the Symfony framework were adopted for that matter. While a lot of the Drupal concepts (Fields, Views, Entities, Nodes) continue to exist in Drupal 8, they are now implemented using object-oriented programming design patterns.

The advantages and disadvantages of object-oriented programming are well-known. The disadvantages are verbosity, size, slower performance and the amount of work it takes to write (including the design planning that goes into it). For people that are new to object-oriented programming the learning curve could be steep; some of the key programming techniques, such as inheritance and polymorphism, can be challenging initially. The advantages are encapsulation (both to avoid tampering with internal values and to hide implementation details), faster development thanks to re-use, extensibility, and better maintainability. Compared to procedural programs, object-oriented programs are easier to maintain, extend and refactor. So although a lot of work is spent to write the code, less work is needed to maintain it over time. For Drupal 8 this means that the code will be more abstract, more verbose, and slower, but also more maintainable, more modular, and more accessible to non-Drupal developers. The end goal is that Drupal 8 should help attract new people to Drupal in a way Drupal 7 did not. As an example, exactly the same happened with other projects like Symphony. Symphony 2 was a complete refactor and re-architecture from the previous version. People had different opinions about that. A lot of people were alienated, yet at the same time Symfony 2 was a big success.

The same thing has happened with the major releases of Drupal as well, despite how much change each one brings. Many of Drupal 8 development changes are described below based on the users.

### 4.1 End users and clients

As mentioned, Drupal 8 is a powerful platform that requires very little to no technical knowledge. It's purpose is to let content administrators use the website as they want; "getting the right content to the right people in the right language at the right time".

Comparing Drupal to its competitors, the leading Drupal-based company has set some priorities to fill in the most important gaps. The biggest shortcoming in Drupal was set to be the authoring experience. Drupal 8 expands previous functionality by allowing users to do the following:

– Easily create lists and image galleries using views.

- Using WYSYWIG editor in core to create well-structured pages and still having the option to see the rich text format.
- Inline editing is possible by default. It works with image fields, taxonomy, files, regular text, formatted text, etc.
- Use two column layout, which makes it easier to separate the essentials on the left and meta/admin data on the right.
- Preview the newly created content on the front end.

Since the mobile technology evolved so much lately, responsive designs are a must. Drupal 8 supports adding and editing content from any mobile device and comes with all built-in themes that are fully responsive. Since mobile was also a big priority, everything from the installer to the modules page has been re-designed with mobile in mind. There were made lots of big improvements:

- Responsive core themes.
- It comes with picture module that uses HTML5 picture element for responsive images.
- It is based on configurable breakpoints - courtesy of the new Breakpoint module.
- The new toolbar is mobile friendly with large tap areas, and a nice vertical sidebar or app icons at the top.
- Responsive tables make sure that the most important columns are also displayed on smaller screens.

### 4.2 Site builders

All new out-of-the-box features like views, configuration management and an improved user interface (UI) make the job of site builders a lot easier. Drupa 8 allows site builders to:

- Attach new field types (like Entity Reference) to different new types of content.
- Customize the look and feel of data entry forms on Form displays.
- Use views and customize the default front page, admin listings, sidebar content, image galleries, slide shows, etc. with 0 lines of code.
- Have a totally new administration experience, since the interface got lots of attention. The elements are redesigned for responsiveness, consistency and better accessibility.

### 4.3 Designers and themers

The most noticeable update regarding the UI is the usage of HTML5 markup for all themes and core components. The other most talked-about change is Twig, a templating engine that makes Drupal theming totally different in comparison with the older versions where template files were used for the HTML markup and PHP variables. Twig is a lot faster, more secure and makes the Drupal 8 markup a lot cleaner, while the admin UI provides a more consistent experience and is easier to use. It is also much easier to debug and view where information is coming from. It can be also seen as a development module for themers. Other than that, the front end libraries are updated. Drupal 8 offers the latest versions of jQuery, jQuery UI and Backbone, which is used as a front-end library and as in core for data modeling and state syncing.

### 4.4 Developers

The main goal was to make Drupal 8 developers do more, with fewer steps and with less knowledge about the framework itself. It was achieved with a totally new file-based configuration management system, which is a mix of some of the most modern and popular web technologies that we can use today; RESTful web services and the use of Symfony 2 framework. Developers can also benefit by the following updates:

- It uses the new Rest and serialization API, which means serialized data can be outputted as JSON and XML.
- New libraries: Assetic, Composer, PHPUnit, Guzzle, Zend Feed Component.
- Easily track changes in configurations with version control and update the production site cleanly.

## 5 Current Drupal 8 status and future releases

The Drupal community was working hard since March, 2011 on Drupal 8 and making progress constantly. The first ever stable version of Drupal 8 was released oh 19th November 2015, followed by release parties all around the globe, organized by contributors themselves. This key milestone was achieved with the work of more than 2,300 people altogether. There have been more than 11,500 committed patches to 15 alpha releases. Semantic versioning will follow after Drupal 8.0.0 with a regular release schedule. Patch-level releases will follow monthly corresponding to Drupal 7's release windows. Minor releases are planned to follow every six months. The corresponding Drupal timeline is displayed in the Figure 1.

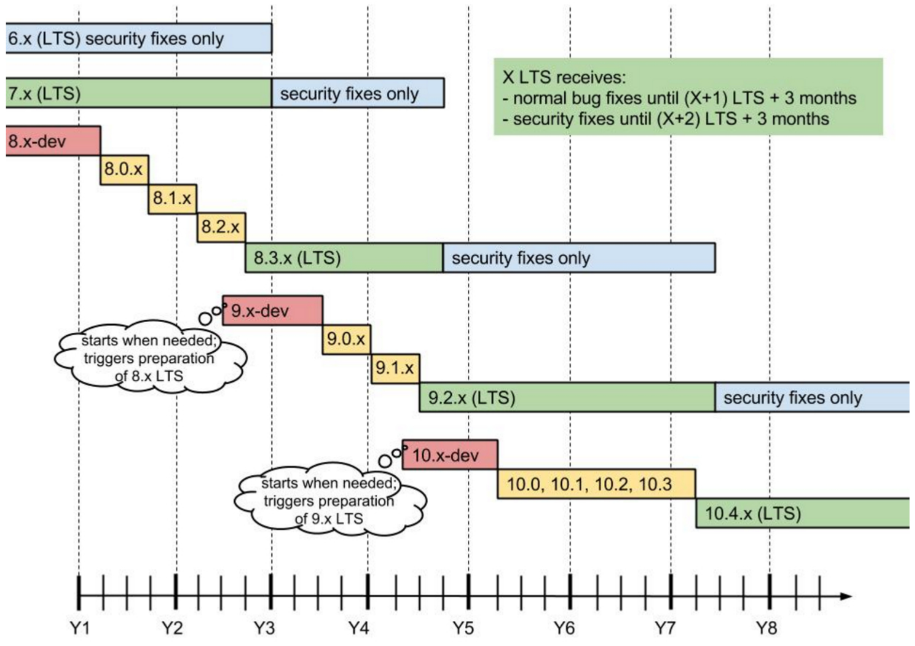

6.x (LTS) security fixes only
7.x (LTS)
security fixes only
X LTS receives:
- normal bug fixes until (X+1) LTS + 3 months
- security fixes until (X+2) LTS + 3 months
8.x-dev
8.0.x
8.1.x
8.2.x
8.3.x (LTS)
security fixes only
9.x-dev
starts when needed; triggers preparation of 8.x LTS
9.0.x
9.1.x
9.2.x (LTS)
security fixes only
10.x-dev
starts when needed; triggers preparation of 9.x LTS
10.0, 10.1, 10.2, 10.3
10.4.x (LTS)
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y8

Figure 1: Drupal timeline plan for specific version releases. As seen, the plan is well defined, although there can be many variations.

## 6 Modules

Users can extend and customize Drupal functionality with contributed modules. If a module does not quite do what is supposed to do or if there are any bugs, everyone is invited to help the module maintainer and report and, if possible, also fix the opened issue. I am lucky to be part of the Drupal community and lately we were contributing to various modules, but our main focus is on Translation Management Tool and Paragraphs, which will be shortly described below.

### 6.1 Translation management tool

The idea behind Translation Management Tool (TMGMT) module grew-up in 2011. The plan was to build a “contrib” extension to support editors, publishers, translators and project managers during their process of content translation. It uses existing language tools and data structures. The purpose is to solve all the confusion and problems that were arising while doing translation in Drupal. Let’s say there were 100 “nodes” on the page and we need to translate it to 5 different languages To do this, we would end up with 500 nodes, which all contain the same content, but in different languages. To maintain all the nodes and translations was a real struggle and it was impossible to see and manage the status of the translations. Also, there was no workflow - external services were not supported and the translator had to log into the site configuration to do his job. All this ended up in a big mess. With Translation Management Tool most of these problems are solved and the translation with Drupal is streamlined and user-friendly.

The architecture of the module is simple. It allows translation of any kind of text elements, from content, configuration and interface texts in just a few clicks. In TMGMT these sources are named source plugins and are added to the translation job, as seen in Figure 2. Each of the source plugins in a job is called job item. The translation of the job can be done by local or remote translators (also called Translation Plugins) of different kinds and the translation process can be totally automated. This means that external services can be used for creating a foreign language version of the source, but also the user himself can translate the text via the Local Translator and save it. Also automated translators are available. The module is based on a plugin architecture, that allows additional sources and translation services to be added by everyone [8].

The following services are part of the module:

– Local Translator gives the users the ability to translate the source on their own.

– File export and import via XLIFF and HTML.

– Gengo (human) as a remote translation service provider.

– Microsoft and Google Translate (machine) use their machine translation APIs.

With the installation of TMGMT the functionality of the page is extended. The user can then choose one or more languages to translate the node to and request a translation with the corresponding button. For each of the language chosen, a translation job is created. The user can

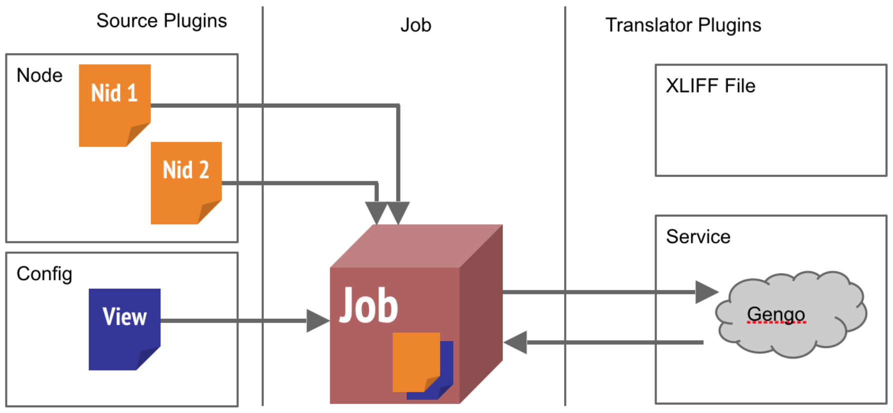


Figure 2: TMGMT architecture is made out of three basic parts; Source Plugins, Jobs (as TMGMT core functionality) and Translator Plugins.

then request a translation from the list of enabled translators. After getting the translation back, the job state can be processed, unprocessed, active, reviewed and finished. Translation overview offers a quick look at the list of all jobs with all the relevant information.

Our contribution for TMGMT is very vast. We got involved into this module at a very early phase, so our main points were discussing and implementing new functionalities, discovering and fixing bugs, extending web tests and reviewing other contributor's work – patches. We have also done a significant part of the improvements for the interface to reduce complexity and enhance the user experience.

One of our main topics was definitely the implementation of Gengo translator. It is a translator plugin for the TMGMT project, this means it extends it's functionality by allowing the submission of translation jobs to Gengo, which is an external translation service provider [5]. Because this was a port from an older version, lots of code modifications needed to be made, such as syntax changes, different remote mappings, GET, POST and PUT requests, etc. Running the tests and checking the test errors helped me with the tasks mentioned above.

Another important issue was regarding the stability of the module when deleting a translator with active translation jobs or translation items assigned. This issue involved extending the code by creating a new method called *hasTranslator*(). This simplifies the process of checking if a translator has a *target_id* and a plugin assigned. Also the translator class is simplified by adding a *hasPlugin*() method. This resulted in a lot of code refactoring through the whole module. *TranslatorTest* was extended to cover new possible cases for deleting a translator with jobs and job items, for example, for the jobs that are in the finished state - should be deleted, and active state – should not be deleted. As an addition, a success message was added after a successful removal of a translator.

## 6.2 Paragraphs

The Paragraphs module is a fairy new addition to the Drupal project. When it comes to content creation, it offers a rapid and straightforward path towards improving the quality of the website and user experience. It allows site builders to make things clearer for the end users and to give them more editing power while still have more control over their misbehavior. The old way was to put all the content in one WYSIWYG body field including images and videos. This approach had many issues:

- Inability to add rich content (galleries, accordions, parallax backgrounds etc.).
- Inadequate markup and/or undesirable inline styles.
- Users confusion.
- Bad-designed content (web pages).

With Paragraphs, the content is much more structured and easier to use. Users can choose from pre-defined paragraph types, which are independent from one another.

As a site builder, you can add an unlimited number of paragraph types to the site - each with its own fields and displays. In easy words, imagine Paragraph types as mini-content types which can be created on nodes. Since they are basic Drupal entities, it is easy to see what fields should they be composed of and how they should look like. So the biggest advantage of Paragraphs is the ability for a content editor to have total control over the flow of the content using drag-and-drop sorting.

For reference, part of my contribution to this module was providing a demo module as an example for new users which contains four different default paragraph types; a simple text field, a text with an image, an image gallery and a user, which outputs user information. These are all styled in the CSS files, so that the whole page with paragraphs is responsive on mobiles and tablets [6].

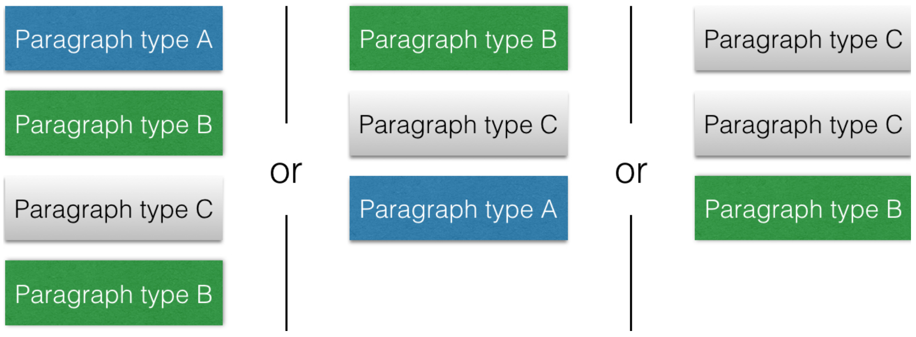


Figure 3: Options with combining paragraph types are endless. The order is easy to set with drag-and-drop functionality.

In many ways, Paragraphs module can be compared with Field Collection module, since it offers similar functionality. With Field Collection, a site builder creates a set of grouped fields and an editor can then add as many of those collections to a node, one after another. The problem here is that they all must be of the same type and there can not be any other type of content in between them. On the other side, with Paragraphs an editor can use more types together, in any order, thereby creating a flow of content without restriction. A simple example is presented in Figure 3.

Although not being "out in the wild" for a long time, this module became very popular among the users and is already being used on many web sites. One of them is also `drupal.org` (Easy Content Authoring page), which can be seen in Figure 4.

We mainly contributed to this module with implementing Paragraphs translation with TMGMT, so that users can easily translate structured data. This helps translators a lot, because they have the source in smaller bits, which are always easier to translate than bigger unstructured chunks of texts. For better history control, revisions are implemented so that users can easily spot the differences. For this we needed to extend TMGMT to depend on Paragraphs and add TMGMT specific configuration. Paragraphs needed to support translation on entity level. This was really challenging. In regular situations it is not the entity reference that should be set to translatable, because the wrapping field should maintain the same set of translated paragraphs. While translating some content, all paragraphs from the default language have to still be there and the user should have the ability to translate them into the target language. On save, the paragraph entities get updated with the proper language context to persist the translation.

## 7 Conclusions

The article describes general information about Drupal, and mainly talks about some of the most important new features in Drupal 8. In the end, two of the modules that we lately worked on are presented with a short description of our main contribution. With a combination of Drupal 8 core features, Paragraphs and Translation Management Tool modules we can build a powerful multilingual website that is very easy to manage, since the content is structured with paragraphs and good looking, because of the flexibility of styling each item separately. And thanks to Translation Management Tool content translation is easier but at the same time more powerful and extendable than ever in just a few clicks. The Drupal 8 is following latest technology trends and with this new release it should attract even more site-builders and end-users. After all, the Drupal community has been working hard to achieve all defined goals for years.

## Acknowledgment

I (S. N.) would like to thank my mentor, Assist. prof. dr. Jurij Šilc, for his support and MD Systems, as a leading contributor in open source, for initiating TMGMT, introducing and mentoring into Drupal, maintaining modules like Paragraphs and porting them to Drupal 8.

## References

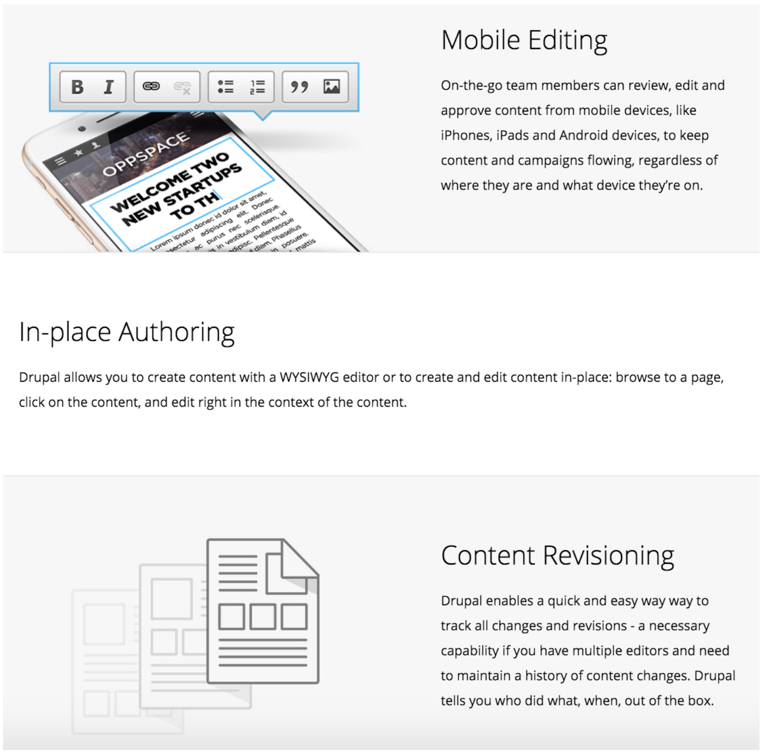
[1] Buytaert, Dries (2006), The pain before he payoff, personal blog. `http://buytaert.net/the-pain-before-the-payoff`, accessed January 14, 2016.

[2] Buytaert, Dries (2010), The State of Drupal. DrupalCon, April 19–21, 2010, San Francisco, CA. `https://archive.org/details/Css3TheFutureIsNow`, accessed January 14, 2016.

[3] Coombs, Karen (2009) Drupal Done Right, *Library Journal*, vol. 34, no. 19, pp. 30–32.

[4] Drupal homepage. `https://www.drupal.org/`, accessed July 2 2015.

[5] Professional Translation Services by Gengo. `http://gengo.com`, accessed January 14, 2016.



Mobile Editing

On-the-go team members can review, edit and approve content from mobile devices, like iPhones, iPads and Android devices, to keep content and campaigns flowing, regardless of where they are and what device they're on.

In-place Authoring

Drupal allows you to create content with a WYSIWYG editor or to create and edit content in-place: browse to a page, click on the content, and edit right in the context of the content.

Content Revisioning

Drupal enables a quick and easy way way to track all changes and revisions - a necessary capability if you have multiple editors and need to maintain a history of content changes. Drupal tells you who did what, when, out of the box.

Figure 4: Paragraphs are already used on drupal.org. Images and text can be floated left or right, full width or styled as the user prefers.

[6] Paragraphs module on Drupal.org. `https://www.drupal.org/project/paragraphs`, accessed January 14, 2016.

[7] Patel, Savan K., Rathod, V. R., Prajapati, Jinga B. (2011) Perforance Analysis of Content Management Systems - Joomla, Drupal and Wordpress, *International Journal of Computer Applications*, vol. 21, no. 4, pp. 39–43.

[8] Translation Management Tool module on Drupal.org. `http://www.drupal.org/project/tmgmt`, accessed January 14, 2016.

# Authentication and Key Agreement Protocol for Ad Hoc Networks Based on the Internet of Things Paradigm

Muhamed Turkanović
University of Maribor, Faculty of Electrical Engineering and Computer Science, 2000 Maribor, Slovenia
CEI-Systems(.eu), Valvasorjeva ulica 10, 2000 Maribor, Slovenia, www.cei-systems.eu
E-mail: muhamed.turkanovic@gmail.com, m.turkanovic@cei-systems.eu
Tel: +386 40 303 874

**Thesis summary**

**Keywords:** authentication, key agreement, ad hoc networks, wireless sensor networks, internet of things

**Received:** February 22, 2016

*The article summarizes the key findings of the doctoral thesis written by the author. The content of the thesis is based on the research fields of authentication and key agreement protocols (AKAP) for wireless sensor networks, and the internet of things (IOT). They key contribution of the thesis is a novel user AKAP for ad hoc networks, which is tailored for the IOT environment.*

*Povzetek: Prispevek predstavlja ključne rezultate doktorske disertacija avtorja. Vsebina disertacije temelji na raziskovlanima področjima protokolov za overjanje in dogovor o ključu (PODK) za brezžična senzorska omrežja in konceptu interneta stvari. Ključni prispevek disertacije je nov PODK za neinfrastrukturna omrežja, ki je prilagojen konceptu interneta stvari.*

## 1 Introduction

The domain of ad hoc networks has gained an additional boost of attention in the last decade due to the increase of interest for the Internet of Things (IOT) paradigm. In the context of application scenarios inside IOT, security and privacy play a pivotal role. The issue with providing security and privacy inside IOT is the resource-constrained architecture (i.e., limited computational and communicational capabilities) of key devices like sensor nodes. As a solution the research community proposes lightweight protocols, which represent a trade-off between efficiency and security.

This paper presents a summary of a PhD Thesis [2] which focuses on lightweight authentication and key agreement protocols (AKAP) for ad hoc networks. The first part of the thesis reviews some existing lightweight AKAP for wireless sensor networks (WSN). It then presents a classification of possible attacks on AKAP for WSN, based on the analysis of existing protocols. Secondly with the help of the classification, an analysis of two novel and prominent AKAP for WSN [6, 1] was performed and the results concluded some flaws and shortcomings. Furthermore the first part encompasses the improvement of these schemes [4, 5].

The second part of the thesis focuses on proposing a new user AKAP for ad hoc networks, tailored for the IOT environment [3].

## 2 Proposed protocol

Numerous AKAPs for WSN have been proposed but very few have addressed the challenge of establishing a shared key in a secure and lightweight manner, between a sensor node and a user outside the WSN and from the IOT environment. In order to fill the gap and solve the problem, we developed a challenge-specific AKAP, which uses a rare four-step authentication model (Fig. 1), that we believe is the most appropriate for the mentioned scenario, where a remote user from the IOT wants to directly connect to a specific sensor node from a WSN.

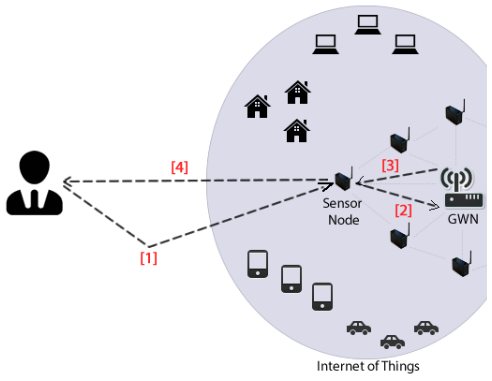


Figure 1: User authentication model of the proposed protocol [3].

Even though the protocols needs to be lightweight, because of the resource-constrained architecture of the sensor nodes, it still has to present the best possible trade-off between security and efficiency. The proposed protocol is thus based only on the use of simple mathematical computations as cryptographic hash functions and XOR. Furthermore, in order to lower the processing burden for the sensor node, the protocol uses the gateway node as a mediator for the authentication process. As a consequence, mutual authentication between all participants had to be implemented, since each participant has to be sure of the authenticity of the counterpart.

The protocol needs to be safe against all known and classified attacks against general AKAP and AKAP for WSN, thus we introduced the use of smart cards. Considering the protocol will be in use inside the IOT environment, it had also to be administrative- and user-friendly, thus enable dynamic node addition, enable the choosing and changing of user passwords, user anonymity etc.

## 3 Results and evaluation

After the development of the protocol, a security and performance analysis was performed. The security analysis was based on the ad hoc security model, which used the aforementioned classification of attacks. The results of the evaluation show that the protocol is resilient against all currently known attacks against general AKAP and AKAP for WSN.

The performance analysis consists of three separate evaluations, i.e. storage, communication and communication evaluation. The results of theses analysis show that the protocol is efficient, lightweight and thus suitable for resource constrained device like sensor nodes.

Furthermore, a comparison between the proposed protocol and other similar ones was performed. The results of the comparison show that the proposed protocol guarantees a higher level security than other protocols, while providing equal or better performance characteristics.

## 4 Conclusion

The paper summarizes the PhD Thesis [2], which addresses the problem of a user AKAP inside the IOT environment. The main contributions of the dissertation are:

- finding flaws and shortcomings in existing user AKAP for WSN;
- presenting a novel classification of attacks on user AKAP for WSN;
- development of improved versions of inadequate or deficient AKAP for WSN;
- development of a novel user AKAP for heterogeneous ad hoc WSNs based on the IOT paradigm.

The focus of further research will be the development a generalized protocol for the purpose of a more general use in the IOT. Moreover this protocol will not be based on the use of smart cards. We will also use the mathematical formal proof as a tool for the security evaluation of the proposed protocols.

## References

[1] K. Das, Ashok, P. Sharma, S. Chatterjee, and K. Sing, Jamuna, “A dynamic password-based user authentication scheme for hierarchical wireless sensor networks,” *Journal of Network and Computer Applications*, vol. 35, no. 5, p. 1646–1656, 2012.

[2] M. Turkanović, “User authentication and key agreement protocols for ad hoc networks, tailored for the internet of things environment,” Ph.D. dissertation, University of Maribor, 2016.

[3] M. Turkanović, B. Brumen, and M. Hölbl, “A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion,” *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.

[4] M. Turkanović and M. Hölbl, “An improved dynamic password-based user authentication scheme for hierarchical wireless sensor networks,” *Electronics and Electrical Engineering*, vol. 19, no. 6, pp. 109–116, 2013.

[5] M. Turkanović and M. Hölbl, “Notes on ’a temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks’,” *Wireless Personal Communications*, vol. 77, no. 2, pp. 907–922, 2014.

[6] K. Xue, C. Ma, P. Hong, and R. Ding, “A temporal-credential-based mutual authentication and key agreement scheme for wireless sensor networks,” *Journal of Network and Computer Applications*, vol. 36, no. 1, p. 316–323, 2013.

## JOŽEF STEFAN INSTITUTE

*Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.*

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 900 staff, has 700 researchers, about 250 of whom are postgraduates, around 500 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slove**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

# INFORMATICA
## AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
## INVITATION, COOPERATION

## Submissions and Refereeing

Please submit a manuscript to: http://www.informatica.si/Editors/ PaperUpload.asp. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

## QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than twentytwo years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

## ORDER FORM – INFORMATICA

Name: ...................................................

Title and Profession (optional): ..............................

...........................................................

Home Address and Telephone (optional): ....................

...........................................................

Office Address and Telephone (optional): ....................

...........................................................

E-mail Address (optional): ..................................

Signature and Date: .......................................

**Informatica WWW:**

**http://www.informatica.si/**

**Referees from 2008 on:**

A. Abraham, S. Abraham, R. Accornero, A. Adhikari, R. Ahmad, G. Alvarez, N. Anciaux, R. Arora, I. Awan, J. Azimi, C. Badica, Z. Balogh, S. Banerjee, G. Barbier, A. Baruzzo, B. Batagelj, T. Beaubouef, N. Beaulieu, M. ter Beek, P. Bellavista, K. Bilal, S. Bishop, J. Bodlaj, M. Bohanec, D. Bolme, Z. Bonikowski, B. Bošković, M. Botta, P. Brazdil, J. Brest, J. Brichau, A. Brodnik, D. Brown, I. Bruha, M. Bruynooghe, W. Buntine, D.D. Burdescu, J. Buys, X. Cai, Y. Cai, J.C. Cano, T. Cao, J.-V. Capella-Hernández, N. Carver, M. Cavazza, R. Ceylan, A. Chebotko, I. Chekalov, J. Chen, L.-M. Cheng, G. Chiola, Y.-C. Chiou, I. Chorbev, S.R. Choudhary, S.S.M. Chow, K.R. Chowdhury, V. Christlein, W. Chu, L. Chung, M. Ciglarič, J.-N. Colin, V. Cortellessa, J. Cui, P. Cui, Z. Cui, D. Cutting, A. Cuzzocrea, V. Cvjetkovic, J. Cypryjanski, L. Čehovin, D. Čerepnalkoski, I. Čosić, G. Daniele, G. Danoy, M. Dash, S. Datt, A. Datta, M.-Y. Day, F. Debili, C.J. Debono, J. Dedič, P. Degano, A. Dekdouk, H. Demirel, B. Demoen, S. Dendamrongvit, T. Deng, A. Derezinska, J. Dezert, G. Dias, I. Dimitrovski, S. Dobrišek, Q. Dou, J. Doumen, E. Dovgan, B. Dragovich, D. Drajic, O. Drbohlav, M. Drole, J. Dujmović, O. Ebers, J. Eder, S. Elaluf-Calderwood, E. Engström, U. riza Erturk, A. Farago, C. Fei, L. Feng, Y.X. Feng, B. Filipič, I. Fister, I. Fister Jr., D. Fišer, A. Flores, V.A. Fomichov, S. Forli, A. Freitas, J. Fridrich, S. Friedman, C. Fu, X. Fu, T. Fujimoto, G. Fung, S. Gabrielli, D. Galindo, A. Gambarara, M. Gams, M. Ganzha, J. Garbajosa, R. Gennari, G. Georgeson, N. Gligorić, S. Goel, G.H. Gonnet, D.S. Goodsell, S. Gordillo, J. Gore, M. Grčar, M. Grgurović, D. Grosse, Z.-H. Guan, D. Gubiani, M. Guid, C. Guo, B. Gupta, M. Gusev, M. Hahsler, Z. Haiping, A. Hameed, C. Hamzaçebi, Q.-L. Han, H. Hanping, T. Härder, J.N. Hatzopoulos, S. Hazelhurst, K. Hempstalk, J.M.G. Hidalgo, J. Hodgson, M. Holbl, M.P. Hong, G. Howells, M. Hu, J. Hyvärinen, D. Ienco, B. Ionescu, R. Irfan, N. Jaisankar, D. Jakobović, K. Jassem, I. Jawhar, Y. Jia, T. Jin, I. Jureta, Đ. Juričić, S. K, S. Kalajdziski, Y. Kalantidis, B. Kaluža, D. Kanellopoulos, R. Kapoor, D. Karapetyan, A. Kassler, D.S. Katz, A. Kaveh, S.U. Khan, M. Khattak, V. Khomenko, E.S. Khorasani, I. Kitanovski, D. Kocev, J. Kocijan, J. Kollár, A. Kontostathis, P. Korošec, A. Koschmider, D. Košir, J. Kovač, A. Krajnc, M. Krevs, J. Krogstie, P. Krsek, M. Kubat, M. Kukar, A. Kulis, A.P.S. Kumar, H. Kwaśnicka, W.K. Lai, C.-S. Laih, K.-Y. Lam, N. Landwehr, J. Lanir, A. Lavrov, M. Layouni, G. Leban, A. Lee, Y.-C. Lee, U. Legat, A. Leonardis, G. Li, G.-Z. Li, J. Li, X. Li, X. Li, Y. Li, Y. Li, S. Lian, L. Liao, C. Lim, J.-C. Lin, H. Liu, J. Liu, P. Liu, X. Liu, X. Liu, F. Logist, S. Loskovska, H. Lu, Z. Lu, X. Luo, M. Luštrek, I.V. Lyustig, S.A. Madani, M. Mahoney, S.U.R. Malik, Y. Marinakis, D. Marinčič, J. Marques-Silva, A. Martin, D. Marwede, M. Matijašević, T. Matsui, L. McMillan, A. McPherson, A. McPherson, Z. Meng, M.C. Mihaescu, V. Milea, N. Min-Allah, E. Minisci, V. Mišić, A.-H. Mogos, P. Mohapatra, D.D. Monica, A. Montanari, A. Moroni, J. Mosegaard, M. Moškon, L. de M. Mourelle, H. Moustafa, M. Možina, M. Mrak, Y. Mu, J. Mula, D. Nagamalai, M. Di Natale, A. Navarra, P. Navrat, N. Nedjah, R. Nejabati, W. Ng, Z. Ni, E.S. Nielsen, O. Nouali, F. Novak, B. Novikov, P. Nurmi, D. Obrul, B. Oliboni, X. Pan, M. Pančur, W. Pang, G. Papa, M. Paprzycki, M. Paralič, B.-K. Park, P. Patel, T.B. Pedersen, Z. Peng, R.G. Pensa, J. Perš, D. Petcu, B. Petelin, M. Petkovšek, D. Pevec, M. Pičulin, R. Piltaver, E. Pirogova, V. Podpečan, M. Polo, V. Pomponiu, E. Popescu, D. Poshyvanyk, B. Potočnik, R.J. Povinelli, S.R.M. Prasanna, K. Pripužić, G. Puppis, H. Qian, Y. Qian, L. Qiao, C. Qin, J. Que, J.-J. Quisquater, C. Rafe, S. Rahimi, V. Rajkovič, D. Raković, J. Ramaekers, J. Ramon, R. Ravnik, Y. Reddy, W. Reimche, H. Rezankova, D. Rispoli, B. Ristevski, B. Robič, J.A. Rodriguez-Aguilar, P. Rohatgi, W. Rossak, I. Rožanc, J. Rupnik, S.B. Sadkhan, K. Saeed, M. Saeki, K.S.M. Sahari, C. Sakharwade, E. Sakkopoulos, P. Sala, M.H. Samadzadeh, J.S. Sandhu, P. Scaglioso, V. Schau, W. Schempp, J. Seberry, A. Senanayake, M. Senobari, T.C. Seong, S. Shamala, c. shi, Z. Shi, L. Shiguo, N. Shilov, Z.-E.H. Slimane, F. Smith, H. Sneed, P. Sokolowski, T. Song, A. Soppera, A. Sorniotti, M. Stajdohar, L. Stanescu, D. Strnad, X. Sun, L. Šajn, R. Šenkeřík, M.R. Šikonja, J. Šilc, I. Škrjanc, T. Štajner, B. Šter, V. Štruc, H. Takizawa, C. Talcott, N. Tomasev, D. Torkar, S. Torrente, M. Trampuš, C. Tranoris, K. Trojacanec, M. Tschierschke, F. De Turck, J. Twycross, N. Tziritas, W. Vanhoof, P. Vateekul, L.A. Vese, A. Visconti, B. Vlaovič, V. Vojisavljević, M. Vozalis, P. Vračar, V. Vranić, C.-H. Wang, H. Wang, H. Wang, H. Wang, S. Wang, X.-F. Wang, X. Wang, Y. Wang, A. Wasilewska, S. Wenzel, V. Wickramasinghe, J. Wong, S. Wrobel, K. Wrona, B. Wu, L. Xiang, Y. Xiang, D. Xiao, F. Xie, L. Xie, Z. Xing, H. Yang, X. Yang, N.Y. Yen, C. Yong-Sheng, J.J. You, G. Yu, X. Zabulis, A. Zainal, A. Zamuda, M. Zand, Z. Zhang, Z. Zhao, D. Zheng, J. Zheng, X. Zheng, Z.-H. Zhou, F. Zhuang, A. Zimmermann, M.J. Zuo, B. Zupan, M. Zuqiang, B. Žalik, J. Žižka,

# *Informatica*

## An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

**Subscription Information** Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Litostrojska cesta 54, 1000 Ljubljana, Slovenia.
The subscription rate for 2016 (Volume 40) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printing: ABO grafika d.o.o., Ob železnici 16, 1000 Ljubljana.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Janez Perš)
Slovenian Artificial Intelligence Society (Dunja Mladenić)
Cognitive Science Society (Urban Kordeš)
Slovenian Society of Mathematicians, Physicists and Astronomers (Andrej Likar)
Automatic Control Society of Slovenia (Sašo Blažič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Vojteh Leskovšek)
ACM Slovenia (Andrej Brodnik)

Informatica is financially supported by the Slovenian research agency from the Call for co-financing of scientific periodical publications.

Informatica is surveyed by: ACM Digital Library, Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

**Volume 40 Number 1 March 2016** **ISSN 0350-5596**

# *Informatica*

## An International Journal of Computing and Informatics

Informatica **40** (2016) Number 1, pp. 1–155