Volume 35 Number 1 March 2011 ISSN 0350-5596

Informatica

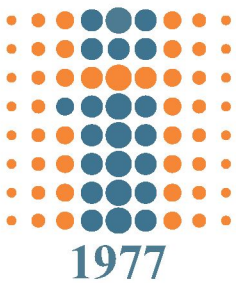
An International Journal of Computing and Informatics

Special Issue:

Autonomic and Self-Adaptive Systems

Guest Editor:

Javier Cámara
Carlos Cuesta
Miguel Ángel Pérez-Toledano



EDITORIAL BOARDS, PUBLISHING COUNCIL

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 Autonomic and Self-Adaptive Systems

Traditionally, handling changing requirements, faults, or upgrades on different kinds of software-based systems have been tasks performed as a maintenance activity conducted by human operators at design or development time. However, factors such as uncertainty in the operational environment, resource variability, or the critical nature of some systems which cannot be halted in order to be changed, have lead to the development of systems able to reconfigure their structure and behaviour at run time in order to improve their operation without any human intervention.

This kind of systems, which typically operate using an *explicit representation* of their structure and goals, has been studied within different research areas of software engineering (e.g., component-based development, requirements engineering, software architectures, etc.) and described with different names, which put their emphasis on different aspects. From those different names (self-healing, self-managed systems, etc.) we have selected two which are among the most popular, namely *autonomic* and *self-adaptive* systems, for the title of this special issue.

Though there are many definitions for them - and sometimes they are even considered equivalent - we will provide here a brief distinction between both terms, which still refer to closely related approaches. A system will be considered *autonomic* when it automatically provides a number of essential features and system-wide properties, without the need of explicit human intervention - sometimes, not even from the programmer. On the other hand, a system is *self-adaptive* when it provides the means to adapt to either external or internal changes by using its own resources. Obviously, many of the autonomic functions provide adaptivity, and most of the self-adaptive features must be automatically provided.

Another promising approach to systems able to dynamically adapt themselves is that of *self-organizing* systems. These are typically composed by a large number of constituent components which operate according to a set of local rules, rather than with an explicit representation of its structure and goals – i.e. providing a decentralized setting. In this case, the *emergent* behaviour derived from component interaction stabilizes the system in the event of faults or changes in the environment which need to be handled.

Although most research efforts in both approaches have been isolated and lacked specific forums for discussion until recently, there is a thriving international community currently involved in the study of these, sometimes known as *self-* systems*, laying out the foundations that will enable their systematic development. There are already several workshops, conferences and symposia devoted to the study and discussion of this topic; and this special issue follows also from the efforts of this growing community.

This special issue gathers several selected, extended and enhanced versions for papers from the First and Second editions of the Workshop on Autonomic and Self-Adaptive Systems (WASELF), which were respectively celebrated in 2008 and 2009 in Spain, emphasizing the participation of both authors and reviewers from the international research community. Additionally, several internationally recognized experts in the area have been invited to contribute to this issue, providing a baseline for the high quality of the material included in this issue. Every published article has followed a rigorous reviewing process, and had to be approved by at least three reviewers with a high expertise in the selected topics.

After the selection and approval of the reviewing committee, this special issue gathers five valuable contributions. The first two of them present frameworks to support full-fledged self-adaptation; while the next one discusses reconfiguration for behavioural adaptation. The latter two discuss the topic in the context of services, considering their choreographies and model-driven composition. We will briefly summarize all of them in the following.

The first paper, entitled *"A Framework for Automatic Generation of Processes for Self-Adaptive Software Systems"* has been authored by Carlos Eduardo da Silva and Rogério de Lemos, and presents a framework for run-time generation of self-adaptation processes in software systems, explaining the need for the automatic generation of these processes at run-time, and how to base this approach on AI planning and workflows.

The second paper, entitled *"An Aspect-Oriented Approach for Supporting Autonomic Reconfiguration of Software Architectures"*, and authored by Cristóbal Costa, Jennifer Pérez, and José A. Carsí presents a proposal to support the *autonomic reconfiguration* of hierarchical software architectures, taking a semi-decentralized approach to tackle the problems of maintainability present in *self-organizing* systems and scalability in *self-adaptive* ones.

Next, the third paper, entitled *"Component Reconfiguration in Presence of Mismatch"* and authored by Carlos Canal and Antonio Cansado, discusses how to reconfigure systems in which components present mismatch and are *not* designed with reconfiguration capabilities. In particular, this work identifies the requirements for achieving run-time component substitution and defines interchangeability notions under *behavioural* adaptation.

The fourth paper, entitled *"Realizability and Dynamic Reconfiguration of* Chor *Specifications"* has been authored by Nima Roohi and Gwen Salaün. This paper presents solutions to check if a certain service choreography is realizable and if a specific *reconfiguration* can be applied dynamically to the software system. The paper uses Chor as choreography specification language and proposes an encoding of Chor into the FSP process algebra in order to

check the realizability of the choreography.

Finally, the paper entitled *"Model-Based Dependable Composition and Monitoring of Self-Adaptive Services"*, authored by Javier Cubo, Carlos Canal, and Ernesto Pimentel, presents an approach based on self-adaptive and error recovery techniques. The proposal uses a *model-based* mechanism to formalise Service-Oriented Architectures in order to decrease the cost of their maintenance and evolution.

The guest editors wish to thank Professor Matjaz Gams for providing us with the opportunity to edit this special issue on Autonomic and Self-Adaptive Systems. Finally, the editors would also like to thank both the authors of the papers for their contributions and all the referees for their critical and valuable comments. All their efforts helped to ensure the high quality of the material presented in this special issue.

Javier Cámara
Carlos Cuesta
Miguel Ángel Pérez-Toledano

A Framework for Automatic Generation of Processes for Self-Adaptive Software Systems

Carlos Eduardo da Silva and Rogério de Lemos
School of Computing, University of Kent
Canterbury, Kent, CT2 7NF, UK
E-mail: {ces26, r.delemos}@kent.ac.uk

Keywords: self-adaptive systems, workflow generation, planning, process, architectural reconfiguration

Received: May 10, 2010

The self-adaptation of software systems is a complex process that depends on several factors that can change during the system operational lifetime. Hence, it is necessary to define mechanisms for providing a self-adaptive system the capability of generating during run-time the process that controls its adaptation. This paper presents a framework for the automatic generation of processes for self-adaptive software systems based on the use of workflows, model-based and artificial intelligence planning techniques. Our approach can be applied to different application domains, improves the scalability associated with the generation of adaptation plans, and enables the usage of different planning techniques. For evaluating the approach, we have developed a prototype for generating during run-time the workflows that coordinate the architectural reconfiguration of a web-based application.

Povzetek: Opisano je okolje za generiranje procesov za prilagodljive sisteme.

1 Introduction

It is commonly agreed that self-adaptive software systems should have the capability of modifying their own structure and/or behaviour at run-time due to changes in the system, its requirements, or the environment. In order to determine the actions to be taken to adapt itself, a self-adaptive software system observes and analyses itself and its environment, and if an adaptation is deemed to be necessary, a plan is generated for altering the system in a controlled manner. These systems are usually implemented in terms of a feedback control loop, and one of its key activities is the generation of adaptation plans [6].

The self-adaptation of a software system is a complex process that depends on several factors that may change during the system operational lifetime. In this context, an adaptation plan should respect the relationships and dependencies among the elements that compose the system, taking into account the actual state of the system and its environment. Thus, it is expected that self-adaptive software systems should be able to generate adaptation plans during run-time, in order to deal effectively with the variability and uncertainty involved in the adaptation. Otherwise, the self-adaptation process would cease to be consistent with the actual system that it controls, since the adaptation process would not be able to consider the actual system state, thus restricting what and how can be adapted. Some existing approaches for software self-adaptation (such as [7] [15]) explore mechanisms for the selection of adaptation plans, where each available plan and the conditions associated with the selection of that plan are defined at design-time through the use of adaptation policies. However, it is difficult to anticipate, at design-time, all the possible contexts of adaptation for some types of systems. For example, resources considered during the design of the adaptation policies may not be available during the system execution, or new adaptation possibilities that have not been considered at design-time may become feasible due the availability of new resources. To deal with this problem, it is necessary the definition of mechanisms for providing a self-adaptive software system the capability of generating, during run-time, the process that controls and coordinates software adaptation. In a previous work [8], we have outlined an approach based on the use of dynamic workflows for coordinating the self-adaptation of software systems. In this paper, we detail that approach by defining a framework that enables the dynamic generation of processes during run-time. This framework consists of a process and collection of languages, mechanisms and techniques, and its supporting computational infrastructure, consisting of open source and in-house tools,

Currently, there is a wide range of techniques for generating processes in different application domains, such as, web service composition [13], grid computing [11], software management [3], and architectural reconfiguration [4], [20]. These approaches present different solutions that are very specific to their respective domains. Moreover, they are difficult to be reused in other domains [11], and in some cases, in other applications inside the same domain [4]. Compared with existing similar approaches, our main contribution is a generic framework for the automatic generation of processes for self-adaptive software systems. The objective is to define a framework that can be

applied to different application domains, and that can handle the scalability associated with the generation of adaptation plans. In order to achieve this, our framework is based on a combination of techniques, such as, workflows, artificial intelligence (AI) planning and model transformation. Workflows are used as a means to implement the processes, AI planning is used to dynamically generate the processes, while model transformation is employed in the translation between domain specific models into planning problems. A more specific contribution, which is an integral part of the framework, is the use of model-based technology for the generation of adaptation plans, thus enabling the usage of different planning techniques according to the needs of the application domain. In order to increase the scalability of the framework, the generation of processes is split into two levels of abstraction, namely, strategic and tactical, and to reduce the complexity of the overall interaction between these levels, the framework depends on the explicit representation of feedback loops [6], [18]. In order to evaluate the proposed approach, we have implemented a prototype where the proposed framework is being applied for generating adaptation plans that manage the architectural reconfiguration of a web-based self-adaptive application.

The rest of this paper is organised as follows. Section 2 presents some preliminary information that constitute the basis of the defined framework. Section 3 describes the proposed framework. Section 4 presents the application of the framework in the context of architectural reconfiguration of self-adaptive software systems, and evaluates the overall effectiveness of the proposed framework. Section 5 discuss some related work, while the conclusions and future directions of research are presented on Section 6.

2 Background

This section presents a brief overview of some of the key technologies on which the proposed framework for dynamic process generation is based.

We have adopted as a basis for generating processes the three-layer reference model for architecture-based self-managed systems adopted by Kramer and Magee [18]. In their model, the component control layer (the bottom layer) consists of the components that accomplish the system function, and includes facilities for supporting the manipulation of components (e.g., status query, creation and interconnection). The change management layer modifies the component architecture in response to changes that occur at the bottom layer, or in response to new goals from the layer above. The goal management layer (top layer) deals with high level goals of the system by changing management plans according to requests from the layer below and in response to the introduction of new goals.

The techniques for the generation of workflows can be divided into static and dynamic [13]. Static techniques for generating workflows are employed during design or compile-time, when workflows are not expected to change during run-time. On the other hand, dynamic techniques support the generation or modification of workflows at run-time in order to handle changes that may occur. Static techniques can receive as input an abstract model of the tasks that should be carried out, without identifying the resources to be used during execution, and workflow generation consists of selecting the actual resources for the abstract model. While dynamic techniques are able to automatically create the abstract model and select the appropriate resources for its execution. Our approach is classified as a dynamic generation technique where we apply AI planning for create the abstract model.

AI planning endeavours to find from a particular initial state a sequence of actions that are able to achieve an objective. In order to use AI planning, it is necessary to define a domain representation, which identifies among other things the available actions (or tasks) that can be used for generating a plan, and a problem representation, which includes the initial state and the desired goal. Currently, there is a wide variety of planners available that employ different algorithms and techniques, and support different heuristics. These planners allow tasks to be represented in different ways, including, pre- and post-conditions, temporal information (time for executing the task), and hierarchical task networks (tasks that can be decomposed in further tasks). For supporting this wide range of planning systems, there is a standard language called the Planning Domain Definition Language (PDDL), which is used to specify domain and problem representations [14].

Model-based technology explores the use of models at different levels of abstraction, and transformations between levels to manage complexity. These models conform to specific meta-models, which define the relationships among the concepts that can be used in a particular domain. Transformations allow the generation of different artefacts (e.g., source code) based on high level models, ensuring consistency between the models and the generated artefacts [19].

3 Framework for process generation

In our approach, processes are represented through workflows that are dynamically generated. Our framework divides the generation and execution of workflows in three phases: strategic, tactical and operational. At the *strategic phase*, AI planning is used to generate abstract workflows. An abstract workflow describes the set of tasks and the data dependencies among them, but without identifying the actual resources that will be used during the workflow execution. At the *tactical phase*, an abstract workflow is mapped into a concrete workflow which identifies the actual resources associated with the tasks. It is important to note that an abstract workflow can be mapped into different concrete workflows by using different combinations of resources. At the *operational phase*, the concrete workflow is executed. Figure 1 presents a simplified view of our

approach for the dynamic generation of workflows.

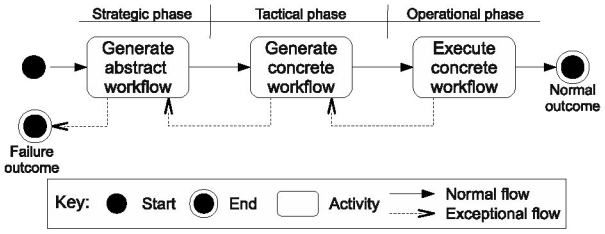


Figure 1: Overview of workflow generation.

The proposed approach can be placed in the context of the three layers reference model for self-managed system adopted by Kramer and Magee [18]. At the goals management layer, corresponding to our strategic phase, abstract workflows are generated according to a particular objective (or goal). These workflows are used as a basis for generating the concrete workflows at the change management layer, corresponding to our tactical phase. Once a concrete workflow has been generated, it is executed at the component control layer, corresponding to the operational phase. In our approach, similar to the three layers reference model, if errors occur at a particular phase and that phase is not able to handle the error, these are propagated to the previous phase in which they should be dealt with. In case of a problem occurs during the execution of the concrete workflow, a new concrete workflow is generated at the tactical phase, without the need to generate a new abstract workflow. If it is not possible to generate a concrete workflow (e.g., there are not enough resources), the generation goes back to the strategic phase, where a new abstract workflow is generated. In the eventuality it is not possible to generate an abstract workflow, the generation finishes with an error.

In the rest of this section, we provide more details concerning key aspects of the proposed framework. The first step, though, is to elaborate on the type of workflows supported by our approach, then we describe the task templates that are used for generating workflows, and finally, we present the actual process that enables the dynamic generation of workflows.

3.1 Types of workflows

In the context of the proposed framework for the dynamic generation of processes during run-time, it is important to distinguish how workflows are related to the system itself. Workflows can either be an integral part of the system or they can be peripheral to the system.

A workflow is an integral part of a system if its execution constitutes the system itself. This is the case of workflows that implement business processes, where each executed task contributes to the service to be delivered by the business (the goal of the workflow). In these workflows, the non-functional properties associated with the workflow may influence the non-functional properties of the system itself. For example, the execution time associated with a particular task should influence the time it takes for the workflow to execute. Moreover, the criteria for selecting which tasks should be part of a workflow and how they should be composed should be dictated by the requirements associated with the system. For example, in the case of web service orchestration, depending on the expected overall execution time of the workflow, the appropriate combination of tasks should be identified to compose the workflow.

A workflow is a peripheral part of a system if its execution has as an outcome the system that is expected to deliver the actual services. This is the case of workflows, for example, that coordinate architectural reconfiguration of software systems, where the goal of the workflow is to generate a system that can deliver the actual services. In these workflows, the non-functional properties associated with the workflow have almost no relation with the properties of the system being generated by the workflow. Thus, it is expected that the criteria for generating the workflow, and the criteria for generating the system from the execution of the workflow to be distinct. For example, in the architectural configuration of a software system, reliability could be a key property for that system, and it may not be considered when generating the workflow that would produce the actual system.

3.2 Defining task templates

In order to use AI planning, it is necessary to define task templates to be used by the planner in terms of their pre- and post-conditions. In our work we have used Planning Domain Definition Language (PDDL) to define task templates, and these templates are implemented as workflows. These implementations are structured in terms of atomic actions [10], as a means for incorporating fault tolerance, based on exception handling, into the workflow execution. Figure 2 presents the base structure defined for implementing task templates. Following the task templates specification, task template implementations include pre- and post-conditions checks, respectively, before and after the execution of the associated task.

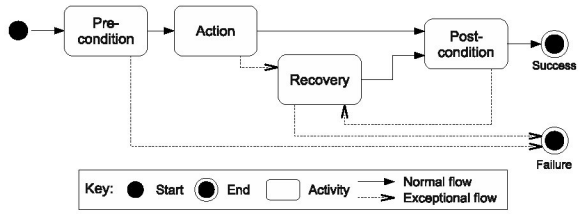


Figure 2: General structure for task templates implementation.

Task templates can have two possible outcomes: Success, representing the successful execution of the task, and Failure, representing failure in the execution of the task. A task template implementation may incorporate a recovery path, which is activated when there is a failure during execution of the task, or a violation of its post-condition. The

recovery can be implemented in different ways (forward or backward) according to the application domain, and the task only finishes successfully when the post-condition is fully satisfied. In our approach, it is the system developer who decides how to implement the recovery. For now, we are restricting the implementation of our approach to backward error recovery, where the effects of a task are undone before finishing with a failure.

3.3 Workflow generation process

As previously mentioned, our framework is partitioned into three different phases, where the two first phases are associated with the generation of workflows, and the last phase is related to the workflow execution. Each phase of the generation process is composed by several activities[1], where some of these activities are dependent on the application domain in which the generation framework is being used, while others are completely independent of the application domain. In the following, we detail the activities associated with the strategic and tactical phases, while focusing on the domain independent activities.

3.3.1 Strategic phase

The main objective at this phase is to find the sequence of tasks that will compose an abstract workflow. This is achieved by means of AI planning techniques. In order to generate a workflow, an AI planner receives as input the goal to be achieved, the initial state, and a set of available task templates (with associated pre- and post-conditions).

Figure 3 presents an overview of how a workflow is generated at the strategy phase. In order to use AI planning, it is necessary, first, to obtain the initial state and the goals associated with the workflow, which are represented by the Obtain current state and Obtain workflow goals activity. These activities are dependent of the application domain in which the generation framework is being used.

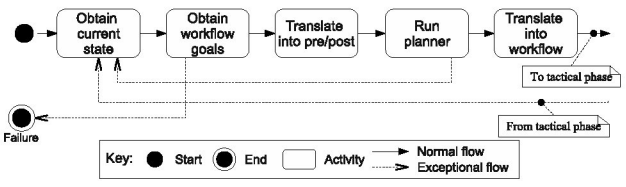


Figure 3: Overview of the activities of the strategic phase.

The Translate into pre/post represents the activity of translating the goal and the initial state from the notation (or representation format) used by the application domain to the notation employed by the planning technique. This translation receives as input one or more models and produces a planning problem representation in PDDL. These input models describe the goal and the initial state associated with the workflow in a domain dependent format. For example, goal-oriented models can be used to represent the objective associated with business processes [16], while architectural models can be used when considering software reconfiguration [20].

Once the pre- and post-conditions have been defined, the next step is the execution of the planner being used (Run planner activity). Depending on the planning technique used, there are several possibilities for generating workflows, and one workflow must be selected based on certain criteria. This selection can be incorporated in the planning technique through the use of different heuristics (such as the number of tasks in the generated workflow, or the time/-cost for executing the generated workflow), or can be a second step after the several possible workflows have been identified. In our approach, we have decided to represent it as a single activity, since it is affected by the planning technique used. Since most planners support PDDL, and receive as input pre- and post-conditions in this language, we can easily change the planner used with no, or very small impact in the task templates specification and the other activities of the strategy level. In case it is not possible to find a plan for a given set of pre- and post-conditions, the Run planner activity finishes with an error, and the strategic phase is restarted.

The Translate into workflow activity receives the output of the planner and translates it into an abstract workflow in the format used for specifying workflows in the execution platform, in our case, the YAWL workflow modelling language [23]. This translation involves the instantiation of each action identified in the PDDL plan as a YAWL workflow task, where the names identified in the action are used for populating the associated workflow task.

It is important to mention that the tasks that compose an abstract workflow are referred to as *strategic tasks*, and the resources associated with these tasks are referred to using logical names, which are sufficient to identify the actual resources at the next phase. In case of a problem at the tactical phase, or during the Run planner activity, the current state is updated (Obtain current state activity) and the Obtain workflow goals activity is activated to decide whether a new goal should be tried, or if the generation finishes with an error.

3.3.2 Tactical phase

The main concern at this phase is to allocate the appropriated resources to the tasks of the abstract workflow. The way in which the resources are identified is dependent on the application domain, and on the type of workflow being considered. When dealing with workflows that are an integral part of the system, the resources of the tactical phase are called *tactical tasks*, which are used for replacing the *strategic tasks* of the abstract workflow. In case of workflows that are a peripheral part of the system, we consider the *strategic tasks* as *parametrised tasks*, where logical names are used as the task parameters. In this case, the resources of the tactical phase are called *concrete pa-*

[1] Activities refer to the steps of the generation, while tasks are associated with the generated workflow.

rameters. Considering those two types of workflows, we have structured the activities of this phase in a way that allows the use of any of the cases (or both at the same time) according to the application domain. Figure 4 presents an overview of the activities at this phase.

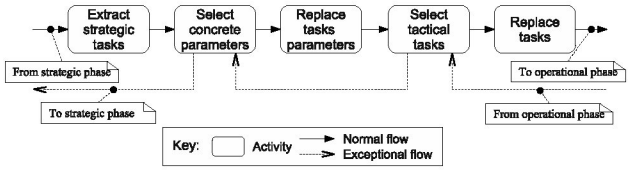


Figure 4: Activities of the tactics level.

The first activity at the tactical phase consists in extracting the strategic tasks that compose the received abstract workflow (Extract strategic tasks). The next activity (Select concrete parameters) is responsible for identifying the concrete parameters of a peripheral workflow. These concrete parameters are then used as replacement for the logical parameters in the extracted tasks (Replace tasks parameters). These tasks, with the concrete parameters identified, are then used in the next activity (Select tactical tasks) for selecting the tactical tasks of an integral workflow. Once the tactical tasks have been selected, they are used as replacement in the tasks of the abstract workflow (Replace tasks), resulting in a concrete workflow that can be executed at the next phase.

In case of failure during the execution of the generated workflow, new tactical tasks are selected for generating a new concrete workflow. If no tactical tasks are selected based on the concrete parameters, new concrete parameters are selected and the process repeated. If there are no available resources for the concrete parameters, the tactical phase finishes with a failure, and the process goes back to the strategical phase. It is important to note that it is necessary to control the relationship between the selection of concrete parameters, and the selection of tactical tasks. For example, in case there are no tactical tasks for a determined set of parameters, this set must be identified, and eliminated from the possible parameters in order to avoid an infinite loop. The same thing happens in the interaction between the execution of a concrete workflow and the selection of new tactical tasks, and between the strategic and tactical phase.

Apart from the changes in the task template implementation, the tactical phase is also customised according with the considered type of workflow. In this way, it is possible to deal with *peripheral workflows* by eliminating the Select tactical tasks activity, only with *integral workflows* by eliminating the Select concrete parameters and the Replace tasks parameters activities, or with both types by not eliminating any of the activities. This decision must be made during the instantiation of the framework into a particular domain. For example, different web service instances could be captured as tactical tasks of an integral workflow.

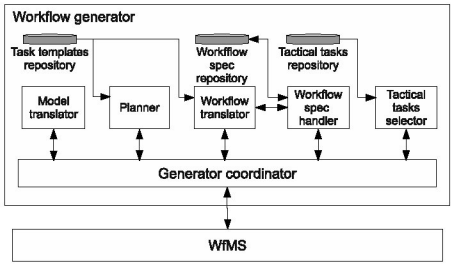


Figure 5: Overview of the infrastructure for supporting process generation.

3.4 Infrastructure

In the sequence, we describe the tools and techniques of the infrastructure that provides the basis for dynamically generating workflows. We present the infrastructure's main components, and identifying those components that must be modified according with the application domain.

Figure 5 presents the infrastructure that underpins the proposed framework. The generation process is managed through a workflow. Workflow specifications are modelled using the YAWL workflow modelling language [23], and are executed in the YAWL Workflow Management System (WfMS). All elements have been implemented using the Java programming language, and the communication among them uses web service technology. The Model translator component must be customised based on the application domain by providing the transformation rules that translates domain specific models into planning problems. We have successfully used different planners, implemented in C and Python (Satplan, Sgplan6, LPG-td, LAMA)[2] by building Java wrappers for each one of them. When dealing with planning based only on pre- and post-conditions, the replacement of the planner used has no impact in the other components of the framework. We also used temporal based planning, requiring the modification of the task template specification for including the appropriated non-functional property (e.g., the cost of each action), and the transformation rules of the Model translator component for including the metric used by the planner (e.g., minimize the total execution time of the generated plan) in the problem representation. The Workflow translator component is responsible for translating a PDDL plan into an YAWL workflow specification. It does that by parsing a PDDL plan and recovering the task implementation associated with each task of the plan. These task implementations are them composed together as the tasks of a new workflow specification using the services of the Workflow spec handler component. The Workflow spec handler provides different services for manipulating workflow specifications, and has been implemented based on the API of the YAWL WfMS. All repositories have been implemented as Java classes. The Workflow spec repository interacts with the YAWL WfMS and its worklets service [1] for load-

[2]Participants of different editions of the International Planning Competition: http://ipc.icaps-conference.org/

ing concrete workflows into the WfMS engine. The Tactical tasks selector is responsible for implementing the decision making associated with the selection of tasks in the tactical phase, while the Tactical tasks repository is used for storing all available tactical tasks.

4 Process generation for architectural reconfiguration

This section describes how the proposed framework is being used for generating the workflows that manage the architectural reconfiguration of a self-adaptive application, presenting the case study application used for its evaluation, its prototype implementation and a brief discussion about some experiments results.

4.1 The reconfiguration process

In the context of the Collect/Analyse/Decide/Act (CADA) feedback control loop [12], our reconfiguration process is related to the decide and act phases, while we assume the existence of mechanisms responsible for the collect and analyse phases. The decide phase of the feedback loop is responsible for identifying what to adapt (the selection of a configuration), and how to adapt (the generation of a workflow for connecting this configuration), while the act phase is responsible for executing the generated workflow. In this scenario, it is important to mention that we are not concerned with the selection of an architectural configuration for the system. This is outside the scope of our work, and there are several existing approaches that can be used to implement this (e.g., [2]). Instead, our focus is in demonstrating how adaptation plans can be generated based on the selected configuration.

In our approach, we divided a configuration model in two different levels of concerns, reflecting the division between abstract and concrete workflows. In this way, an *abstract configuration* describes a system configuration in terms of its components, identified by a logical names, their types and connections, identifying the structure of the system, but abstracting away from the actual component instances. A *concrete configuration*, on the other hand, describes a system configuration in terms of actual component instances, and their respective attributes. Similar to an abstract workflow, an abstract configuration can be instantiated into different concrete configurations depending on the availability of actual component instances.

Figure 6 presents an overview of the reconfiguration process. At the strategic phase, the initial state and goal correspond, respectively, to the current and selected configurations. The selected configuration is an abstract configuration, where the affected elements are treated as abstract elements. For example, when dealing with the replacement of a component, the new component can be treated as an abstract component, allowing the reuse of the abstract configuration by replacing this abstract component. If the process is being applied to establish a new configuration, all elements are treated as abstract elements. The Translate into Pre/post activity generates the planner inputs from architectural models using a translation algorithm based on model comparison techniques, where the architectural models are compared, and the results of this comparison are translated into a problem specification in PDDL using a set of model transformation rules. This algorithm follows the idea of model transformation employed in model-driven engineering, where a model (the comparison results) is transformed into another model (pre- and post-conditions expressed in PDDL) [19]. In case the planner can not find a plan for a given set of pre- and post-conditions, a new configuration is selected, or the reconfiguration finishes with a failure.

At the tactical phase, the Obtain concrete configuration is responsible for finding a concrete configuration for the system, which is used for generating a concrete workflow. This activity corresponds to the Select concrete parameters activity presented in Section 3.3.2. It is important to mention that in the present context, the generated workflows are a peripheral part of the system, where abstract workflows are characterised by the use of logical names as tasks parameters. In this way, we do not consider the Selection of tactical tasks activity at the tactics phase.

The generated workflow is then executed changing the target system. In case of a failure during its execution (e.g., a failure while connecting two components), a new concrete configuration is selected, and a new concrete workflow is generated and executed. For now, we are assuming the existence of exception handling mechanisms capable of undoing the effects of the failed workflow before returning to the tactical phase. In case there are not enough resources for establishing a new concrete configuration based on the selected abstract configuration, a new abstract configuration is selected, and a new abstract workflow is generated. If it is not possible to find a new configuration for the system, the process finishes with an error.

4.2 Case study

In the sequence, we present an example scenario of a distributed system that was used to evaluate our work.

The developed prototype application provides stock quotes portfolio reports with suggestions of investments based on historical and current information about the client, and the actual stock quotes values. In this scenario, we consider the existence of different resources that are captured by different component types, which can be combined in different configurations for the provision of the mentioned service. For each of these component types, there are several component instances that can be used. New instances and resource types can become available, resulting in configurations not envisioned at design-time.

Figure 7 presents an example of a configuration for the provision of this service. The Front end component represents the user access point to the service. Report ser-

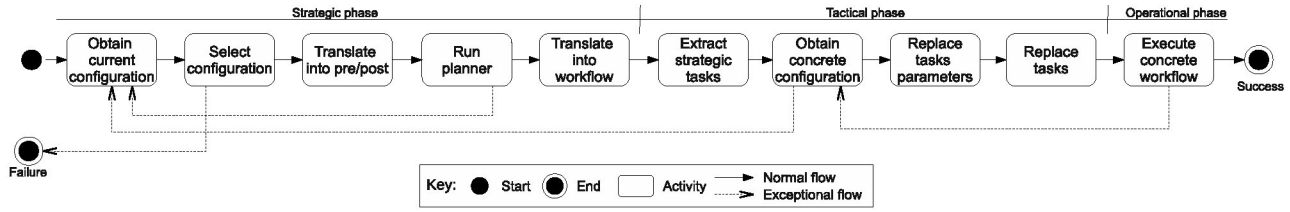


Figure 6: Overview of the process for generating reconfiguration plans.

vice logic represents the application logic of the offered service (stock quote portfolio report), and requires services from internal and external providers. External providers are used for obtaining stock quote values, which can be obtained from different sources. These are captured through Bridge components. A bridge component handles architectural mismatches between the service providers and the system, providing an uniform interface for the different online providers. The Client info logic component, an internal service, provides information about the client, and requires a Database component.

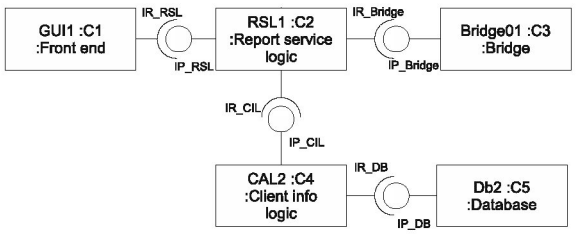


Figure 7: Example of a configuration.

Follow the division between abstract and concrete configurations, each component instance of a concrete configuration is associated to a logical name and type. In this way, GUI1 :C1 :Front End indicates that the component instance GUI1 is associated with logical name C1 with functional requirements associated with components of type Front End.

4.3 Prototype implementation

The infrastructure for supporting the defined reconfiguration process has been implemented based on the architecture presented in Figure 5, and is presented in Figure 8.

In this prototype, architectural models are represented using Eclipse Modelling Framework (EMF) models based on xADL 2.0 [9]. In order to run our experiments, we have implemented a simplified execution platform in which components must be blocked (a kind of quiescent state [17]) before being involved in a reconfiguration, blocked components are not considered when selecting a new configuration, and all architectural elements provide two different types of interfaces, application and configuration services interfaces, as explained in [10]. The Model translator component has been implemented based on EMF

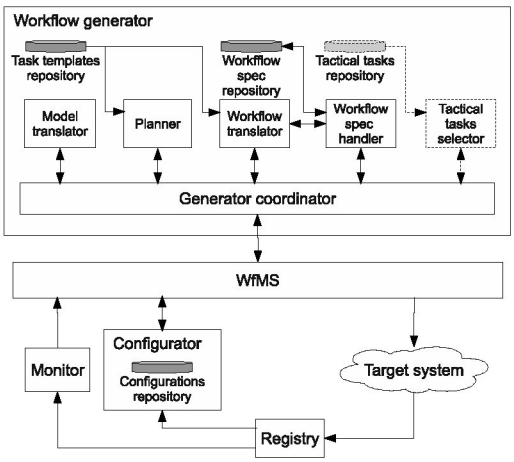


Figure 8: Overview of the overall system architecture.

and Atlas Transformation Language[3]. This component receives architectural models as input, and applies a set of ATL transformation rules for generating a problem description in PDDL. Figure 9 presents an example of a PDDL problem description generated by the Model translator component. This example considers the establishment of the configuration presented in Figure 7. Thus, the inputs used are an empty xADL description, corresponding to the current configuration, and a xADL description of the configuration of Figure 7, corresponding to the selected configuration. The header has an identifier for the problem (reconfigurationProblem) and a reference to the domain representation where the actions and predicates are described. The list of objects involved includes all components (identified by logical names) and their provided and required interfaces. In this example, all involved components are unblocked, and all connections are not established.

Since we consider the generation of peripheral workflows, the Tactical tasks selector component is deactivated by removing the respective activity from the workflow that controls the reconfiguration. The generated workflow changes the configuration of the Target system. The Registry is where all available components, their respective attributes, and a current model of the target system are stored. We assume that the Registry is responsible for monitoring the available resources, and for providing an accurate view of the Target system. Any changes in

[3]http://www.eclipse.org/atl

```
(define (problem reconfProblem)
(:domain reconfigurationDomain)
(:objects C1 C2 C3 C4 C5 - component
  IR_RSL IR_Bridge IR_CIL IR_DB - requiredInterface
  IP_RSL IP_Bridge IP_CIL IP_DB - providedinterface
)
(:init
  (not (blocked C1)) (not (blocked C2)) (not (blocked C3))
  (not (blocked C4)) (not (blocked C5))
  (not (connected C1 IR_RSL C2 IP_RSL))
  (not (connected C2 IR_Bridge C3 IP_Bridge))
  (not (connected C2 IR_CIL C4 IP_CIL))
  (not (connected C4 IR_DB C5 IP_DB))
)
(:goal (and
  (not (blocked C1)) (not (blocked C2)) (not (blocked C3))
  (not (blocked C4)) (not (blocked C5))
  (connected C1 IR_RSL C2 IP_RSL)
  (connected C2 IR_Bridge C3 IP_Bridge)
  (connected C2 IR_CIL C4 IP_CIL)
  (connected C4 IR_DB C5 IP_DB)
  )
))
```

Figure 9: Example of pre- and post-conditions in PDDL.

the resources' availability should be reflected at the Registry. The Monitor component represents the mechanisms used for the collect and analysis phases of the feedback control loop. This component observes the components of the established configuration, checking the values of the component attributes (e.g., response time) against a defined threshold. In case of violation, it starts a reconfiguration by starting the execution of the reconfiguration workflow. The Configurator is responsible for selecting a configuration for the system. It has been implemented based on the use of a utility function that is a linear combination of the utilities of the different elements. The Configurations repository stores the configuration models used during the reconfiguration.

4.4 Discussion

In order to demonstrate and evaluate the proposed approach, we have conducted some experiments involving the architectural reconfiguration of a case study application.

Among the experiments, we have considered the establishment of a particular configuration, which for the planner, is the toughest scenario, since it involves all components of the configuration. In this experiment, we have observed the search space (number of possible actions) of the LPG-td planner, changing the number of resources available and the size of the generated workflow.

The size of the generated workflow depends on the number of components and connections in the selected configuration. Every component of a configuration must be blocked before being connected, and unblocked at the end of the reconfiguration. In this way, a configuration with *n* components and *n-1* connections will have *3n-1* tasks. For example, the configuration of Figure 7, with five components and four connections, requires a workflow with 14 tasks (blocking the five components, establishing the four connections, and unblocking the five components). In these experiments we have considered three different abstract configurations, which require three, four and five components, and contains respectively two, three and four connections.

We have implemented two variations of our approach.

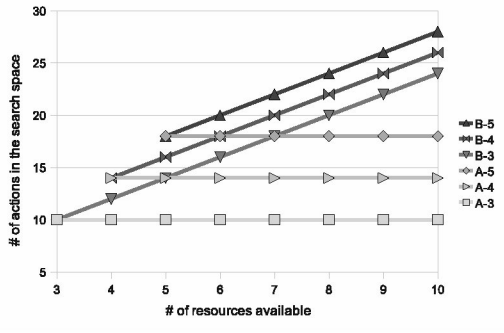


Figure 10: Search space variation changing the resources available.

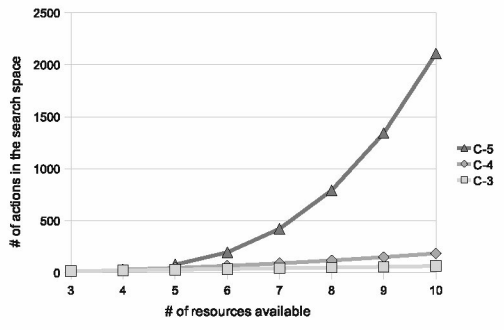


Figure 11: Search space variation considering the selection of a configuration by the planner.

The main difference between these variations is the amount of information passed to the planner. Our approach (identified by A in the graphs) explores the division between strategy and tactics using logical names for representing the components of a configuration. The second approach (identified by B) separates the selection of a configuration from the generation of the plan, but includes all available resources in the planner. While the third approach (identified by C), based on [4], uses the planner for selecting a configuration for the system, including all possible configurations and all available resources in the planner.

Figure 10 shows the variation in the search space as we vary the number of resources available. In this graph, A-3 means the numbers obtained by our approach (A) for a workflow related with a configuration that requires three different components. Since we use logical names to represent the resources required by a configuration, and do not consider all available resources in the planner, the number of resources available does not affect the search space. We notice a linear progression of the search space when we include all available resources in the planner (B-3, B-4 and B-5). The starting points of each curve represents the minimum number of resources required by the correspondent configuration.

Figure 11 presents the change in the search space when the selection of a configuration is combined with the generation of the plan. Based on this experiment, it is clear the overhead caused by mixing selection and plan generation.

As expected, the search space increases with number of resources known by the planner, and the selection of a configuration by the planner further aggravate its scalability. The division between strategy and tactics helps to reduce the search space since the planner does not need to know about all available resources, but only about those involved in the reconfiguration, which are represented by logical names.

5 Related work

The focus of this section is to review how existing approaches generate adaptation plans for self-adaptive software, and we start with those approaches that specify adaptation plans at design-time. Cheng et al. [7] capture adaptation plans as a set repair strategies, consisting of condition-action rules, where conditions are evaluated based on utility functions. Georgas and Taylor [15] propose the use of adaptation policies, also captured as condition-action rules, that indicate what actions should be taken in response to events. Both approaches are focused on the definition of the mechanisms for selecting one adaptation plan from the set of available plans, where each available plan, identifying what and how to adapt, is defined at design-time. A major limitation with these approaches is the difficulty in anticipating all possible contexts in which system adaptation may take place. This has a major impact on identifying the appropriate adaptation plans because of the combinatorial nature between conditions and actions, which also affects the management of the available plans. Different from these approaches, our approach provides the means for defining adaptation plans during run-time.

There are other approaches (such as [5]) that provide support for selecting what to adapt at run-time. However, there is a cost to be paid in these approaches, since the adaptation is enacted by deactivating and reactivating the whole software system, even for the case when a single element needs to be replaced. Moreover, the main focus of these approaches is on the selection of what to adapt based on the state of the environment and the resources required for enabling it, and not on how to enact the selected adaptation. Differently from these approaches, our focus is on the dynamic generation of the process that enacts the adaptation in a way that does not require the complete deactivation of the system.

Some approaches use model comparison techniques, in the context of architectural reconfiguration, for generating adaptation scripts that impact only those elements affected by the adaptation. Alia et al [2] compare the models of the current and the selected system configuration for identifying the actions of the adaptation script. Morin et al. [20] build on top of model comparison by applying priority rules for ordering the identified adaptation actions. However, model comparison and priority rules are not enough to generate plans with complex relationships among the adaptation actions and the involved system components. For it, it is necessary to consider pre- and post-conditions associated with the actions, as demonstrated in [22]. Moreover, the focus of these approaches is not the generation of adaptation scripts, but the selection of the system configuration [2], and coping with the exponential growth in the number of possible configurations [20]. Thus, they do not consider the possible issues associated with the generation of the adaptation scripts. Our approach also applies model comparison for generating adaptation plans, but the results of the comparison are used for identifying the inputs for an AI planner, supporting the generation of plans that can deal with complex relationships between its constituent tasks.

Other approaches have applied AI planning for selecting a configuration and deciding how to change the system at the same time [3], [4]. Both these approaches require the inclusion of all available resources in the planner, which, together with the mix of configuration selection and plan generation, affects the scalability of the planer. Moreover, both approaches require the specification of the current system state and the target system state using PDDL. In our approach, we are not restricted to a fixed set of resources for generating adaptation plans, and we employ model-based technology for translating domain specific models into planning problems. The partition of our approach into different levels of abstraction provides support for dealing with variations in the resources availability and reduces the search space considered by the planner, increasing its scalability, while more specific and scalable techniques can be used for selecting an adaptation for the system (such as [2]).

Concerning workflow generation, several approaches apply AI planning techniques in different domains [21], such as, grid computing [11] and web service composition [13]; however, these approaches are very specific to their respective domains. Our approach is based on ideas from [11] [13], in which the generation is partitioned into strategical and tactical for increasing scalability. Differently from these approaches, we are focused on providing an framework that can be applied to different domains, and consider two types of workflows, integral and peripheral, supporting different decision making and planning mechanisms according with the application domain.

6 Conclusions and future work

This paper has presented a framework for the automatic generation of processes for self-adaptive software systems based on workflows, AI planning and model transformation. The framework can be applied to different application domains by supporting the use of the most suitable generation techniques according to the application domain. Moreover, our approach reduces the search space considered by a planner by splitting the generation in two levels of abstraction, and provides support for generating different types of workflows. In order to evaluate the proposed framework and its respective computational infrastructure,

a prototype was developed for experimenting our approach, and comparing it with similar approaches. The effectiveness of the whole approach was evaluated in the context of a web-based self-adaptive application for obtaining stock quotes reports, where the processes generated at run-time are responsible for coordinating the architectural reconfiguration of the system.

Although the proposed approach for the dynamic generation of process for self-adaptive software systems has produced quite promising results, we have identified a couple of limitations that if properly handled could enhance overall effectiveness of the approach. First, the type of workflows being generated are simple sequential workflows, however, the intent is to incorporate non-determinism and other planning techniques to capture uncertainty into the generation of workflows, and to support different control flow constructs, such as, conditional and parallel execution branches. Second, although the task templates for our framework are structured using atomic actions, one issue that was not yet fully investigated is how to exploit this for the provision of fault tolerance. The intent is to incorporate exception handling mechanisms for tolerating faults that might occur during the execution of generated workflows.

As future work, we would like to simplify the reuse of the framework, and this could be achieved by using meta-transformation languages for translating domain specific models into pre- and post-conditions. Also in this direction, the intent is to incorporate ideas from software product lines into our framework for dealing with the variability of processes. Another future work would be the application of the proposed framework into other application areas to evaluate its overall effectiveness since our initial idea was to see this framework applied to support software self-adaptation whenever processes need to generated during run-time. Concerning the domain of reconfiguration, we intent to consider more complex scenarios in which, for example, there exists transfer of state between components).

Acknowledgement

Carlos Eduardo da Silva is supported by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07D401107BR.

References

[1] M. Adams et al. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In *Proc. of the OTM CoopIS'06*, pages 291–308, 2006.

[2] M. Alia et al. A component-based planning framework for adaptive systems. In *Proc. of the OTM DOA'06*, pages 1686–1704, 2006.

[3] A. Andrzejak et al. Feedbackflow-an adaptive workflow generator for systems management. In *Proc. of the ICAC'05*, pages 335–336, 2005.

[4] N. Arshad et al. Deployment and dynamic reconfiguration planning for distributed software systems. *Software Quality J.*, 15(3):265–281, 2007.

[5] M. Autili et al. Towards self-evolving context-aware services. In *Proc. of the DisCoTec CAMPUS'08*, 2008.

[6] Y. Brun et al. Engineering self-adaptive systems through feedback loops. In *Software Engineering for Self-Adaptive Systems*, pages 48–70. 2009.

[7] S.-W. Cheng et al. Architecture-based self-adaptation in the presence of multiple objectives. In *Proc. of the ICSE SEAMS'06*, pages 2–8, 2006.

[8] C. E. da Silva and R. de Lemos. Using dynamic workflows for coordinating self-adaptation of software systems. In *Proc. of the ICSE SEAMS 2009*, pages 86–95, 2009.

[9] E. M. Dashofy et al. A comprehensive approach for the development of modular software architecture description languages. *ACM Trans. Softw. Eng. Methodol.*, 14(2):199–245, 2005.

[10] R. de Lemos. Architectural reconfiguration using coordinated atomic actions. In *Proc. of the ICSE SEAMS'06*, pages 44–50, 2006.

[11] E. Deelman et al. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1):25–39, 2003.

[12] S. Dobson et al. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1(2):223 – 259, 2006.

[13] S. Dustdar and W. Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30, 2005.

[14] M. Fox and D. Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *J. of AI Research*, 20:61–124, 2003.

[15] J. C. Georgas and R. N. Taylor. Towards a knowledge-based approach to architectural adaptation management. In *Proc. of the WOSS'04*, pages 59–63, 2004.

[16] D. Greenwood and G. Rimassa. Autonomic goal-oriented business process management. In *Proc. the ICAS'07*, page 43, 2007.

[17] J. Kramer and J. Magee. The evolving philosophers problem: Dynamic change management. *IEEE Trans. Softw. Eng.*, 16(11):1293–1306, 1990.

[18] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In *Proc. of the FOSE'07*, pages 259–268, 2007.

[19] J. Miller and J. Mukerji. MDA guide version 1.0.1. Technical report, OMG, 2003.

[20] B. Morin et al. An aspect-oriented and model-driven approach for managing dynamic variability. In *Proc. of the MoDELS'08*, pages 782–796, 2008.

[21] D. Nau. Current trends in automated planning. *AI Magazine*, 28(4):43–58, 2007.

[22] C. Shankar and R. Campbell. Ordering management actions in pervasive systems using specification-enhanced policies. In *Proc. of the PERCOM'06*, pages 234–238, 2006.

[23] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: Yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.

An Aspect-Oriented Approach for Supporting Autonomic Reconfiguration of Software Architectures

Cristóbal Costa-Soria
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
E-mail: cricosso@upv.es

Jennifer Pérez
E.U. Informática, Technical University of Madrid (UPM)
Carretera de Valencia km.7, E-28031 Madrid, Spain
E-mail: jenifer.perez@eui.upm.es

Jose Ángel Carsí
Dept. of Information Systems and Computation, Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
E-mail: pcarsi@dsic.upv.es

Keywords: dynamic reconfiguration, AOSD, autonomic computing, software architecture

Received: February 6, 2010

The increasing complexity of current software systems is encouraging the development of self-managed software architectures, i.e. systems capable of reconfiguring their structure at runtime to fulfil a set of goals. Several approaches have covered different aspects of their development, but some issues remain open, such as the maintainability or the scalability of self-management subsystems. Centralized approaches, like self-adaptive architectures, offer good maintenance properties but do not scale well for large systems. On the contrary, decentralized approaches, like self-organising architectures, offer good scalability but are not maintainable: reconfiguration specifications are spread and often tangled with functional specifications. In order to address these issues, this paper presents an aspect-oriented autonomic reconfiguration approach where: (1) each subsystem is provided with self-management properties so it can evolve itself and the components that it is composed of; (2) self-management concerns are isolated and encapsulated into aspects, thus improving its reuse and maintenance.

Povzetek: Predstavljen je pristop s samo-preoblikovanjem programske arhitekture.

1 Introduction

The increasing complexity of current software systems is becoming unmanageable: large complex systems are more and more difficult to develop and maintain [35]. One of the most promising techniques to deal with the design of large, complex software systems is Software Architectures [39]. Software Architectures[1] provide techniques for describing the structure of complex software systems (i.e. the key system elements and their organization). Its aim is to hide low-level details and help to understand the system. The structure of a software system is described in terms of architectural elements (components and connectors) and their interactions with each other. This structure can be formally described using an Architecture Description Language (ADL), which is used later to build the executable code of the software system. In addition, most ADLs generally support hierarchical composition (i.e. a composition hiding technique for defining systems of systems), which may be helpful for modelling large-scale complex systems in a scalable way. However, although software architecture helps in the description and development of complex systems, this is not enough: the management and maintenance of these systems still requires a great effort. To minimize such effort, self-managed software architectures were proposed [28]. According to the definition of Kramer and Magee [23], a self-managed software architecture is one in which components automatically configure their interaction in a way that: (i) is compatible with an overall architectural specification, and (ii) achieves the goals of the system. However, the development of self-managed architectures still remains a challenge [23]. Although several works have been

[1] This work has been partially supported by the Spanish Department of Science and Technology under the National Program for Research, Development and Innovation project MULTIPLE (TIN2009-13838), and by the Conselleria d'Educació i Ciència (Generalitat Valenciana) under the contract BFPI06/227.

proposed [4], they generally do not scale well for large systems or do not explicitly consider the maintainability of the self-management subsystem itself.

On designing a self-management infrastructure, also its maintenance, scalability and flexibility must be taken into account. First, maintenance should be improved by isolating dynamic change concerns from functional concerns, as it has been stated from other works [8, 10, 26]. Second, scalability can be improved by providing a decentralized self-management infrastructure. Finally, flexibility should be a fundamental property of the self-management subsystem. It should deal with not only goal-oriented proactive changes (i.e. driven autonomously), but also reactive changes (i.e. driven externally) to cope with unanticipated situations, such as the addition of new functionality.

Our work is focused on the design, construction and maintenance of systems with self-management features, from a Model-Driven Development (MDD) perspective [36]. This paper takes a step forward from a previous work [10], which proposed to isolate the dynamic reconfiguration concern from the rest of the system and its decomposition into reconfiguration specifications and reconfiguration mechanisms. In this paper we detail these ideas, addressing the description and design of composite components (i.e. a component composed of other components) capable of reconfiguring its architecture without depending on a unique centralized entity in charge of reconfiguration. In addition, the reconfiguration of composite components is managed without tangling evolution and functional concerns. We have called this feature **aspect-oriented autonomic reconfiguration**, since local autonomy for dynamic reconfiguration is provided for each composite component, and separation of concerns is provided by means of Aspect-Oriented Software Development techniques [22]. Moreover, dynamic reconfiguration is platform-independent, by identifying the high-level features that a reconfigurable technology should provide. Our approach has been applied to PRISMA [32], which provides a platform-independent Aspect-Oriented ADL and is supported by a MDD framework.

This paper is structured as follows. Section 2 presents the design decisions that guided our approach. Section 3 introduces PRISMA, where this approach has been applied to. Section 4 presents a case study, which is used to illustrate the key ideas of this work. Section 5 describes our approach for supporting autonomic reconfigurations. Section 6 discusses the related works addressing dynamic reconfiguration. Finally, section 7 presents the conclusions and further works.

2 Dynamic reconfiguration of software architectures

Our work defines a design approach to build reconfigurable software architectures, a key issue in the development of self-managed software architectures. *Dynamic reconfiguration* of software architectures [16] is a term that is used to refer, generally, to those changes that are produced in the topology of a composite system at runtime, by preserving the system state and consistency. Those dynamic changes may involve: (i) addition of new functionality (i.e. new components), (ii) replacement and/or removal of existing functionality, and (iii) modification of connections between architectural elements.

A dynamically reconfigurable system is characterized by different dimensions or attributes (e.g. change type, granularity, activeness, impact, management, etc.) [4, 5]. We state here the attributes that we have considered the most important to include in our approach and the reasons that guided such decisions:

Abstraction level. Several works have addressed the support for dynamic change, although at different levels of abstraction. On the one hand, a lot of works focus on the technical feasibility of dynamic updating [25, 33, 34]. These works are generally tied to a specific technology: their reconfigurations are defined at a low abstraction level (e.g. in Java). On the other hand, other works focus on the specification of dynamic reconfigurations at a high abstraction level (i.e. by means of ADLs [4, 7, 12, 16]). However, generally these works have not addressed how to support the execution of such high level reconfigurations. Since the dynamic reconfiguration of software systems is highly related with the management of running software artefacts, we should consider not only the specification of how a system should be reconfigured, but also the mechanisms that support the execution of this reconfiguration process. One of the major contributions of this paper is the definition of a model that bridges the existing gap among high-level reconfigurations and low-level supporting mechanisms.

Activeness of changes. Dynamic reconfigurations can be reactive or proactive. On the one hand, *reactive reconfigurations* are dynamic changes that are driven by an external agent (usually the system architect or developer) and through a user interface. Endler defined them as *ad-hoc reconfigurations* [16]. An example of their utility is to perform component updates: to correct bugs or introduce new unanticipated behaviours. On the other hand, *proactive reconfigurations* are dynamic changes that are autonomously driven by the system when some specific conditions or events apply. Proactive reconfigurations are usually described by means of reconfiguration specifications. A reconfiguration specification describes *when* the architecture should change (e.g. in response to certain events or state changes) and *what* kind of changes must be performed on the architecture for each situation. Proactive reconfigurations can be described at design-time (called *programmed reconfigurations* [16]) or synthesized at run-time, according to high-level goals [38]. Both programmed reconfigurations and high-level goals are defined by the architect. An example of their utility is to provide system dependability: if a component instance does not adequately respond, the system might change its connections to another suitable component instance or recreate the instance again. Reactive and proactive reconfigurations should be considered as complementary. Both must be supported to allow a system: to reconfigure itself autonomously (i.e. using programmed

reconfigurations), and to introduce unforeseen changes or updates at runtime (i.e. using ad-hoc reconfigurations). Since both kinds of reconfigurations rely on the same mechanisms to carry out the runtime changes, a way to support reactive and proactive reconfigurations is by explicitly modelling these mechanisms. Thus, the system architect can specify which kinds of reconfigurations are provided, by appropriately enabling or disabling reconfiguration mechanisms and providing proactive behaviour. This provides the architect with a high level of flexibility for defining reconfigurable systems. Our proposal provides support for both reactive and proactive reconfigurations.

Management of dynamic reconfigurations. Due to the growing size of software systems, the scalability of the reconfiguration subsystem is also an important issue [4, 23]. The management of reconfigurations can be addressed either in a centralized or in a decentralized way. On the one hand, centralized approaches (e.g. self-adaptive systems [14, 17, 28]) provide a single, global entity (the *Configuror*) that contains (or generates) both the reconfiguration specifications and mechanisms that will change the overall software system. The main disadvantage is a poor scalability: the larger the system, the more complex and less maintainable the configuror is, since the scope that it must supervise increases proportionally. In addition, a centralized reconfiguration manager turns into a single point of failure: if it fails, the overall system would also lose the ability to reconfigure. On the other hand, decentralized approaches (such as self-organised architectures [18, 37]) distribute reconfiguration management across the elements of the architecture, which are capable of reconfiguring the architecture to which they belong. These approaches have better scalability, since all components can perform reconfigurations. However, a disadvantage is that reconfiguration specifications are spread among different components, thus decreasing maintenance of such specifications. Another disadvantage is that system-wide properties are more difficult to control.

Our proposal follows a *hierarchical decentralized approach*. It is decentralized because each composite component of the architecture has autonomy to reconfigure its internal composition, independently of other components. It is hierarchical because each composite component reconfigures not only its composition, but also drives and coordinates the internal reconfiguration of the composite components it is composed of. That is, a composite component can reconfigure itself autonomously, but in these cases where changes could impact other components of its upper level, the reconfigurations are coordinated by its upper level self-management subsystem, to ensure the architectural consistency.

Separation of concerns. In the context of software evolution, the separation of concerns is important to separate those parts of the software that exhibit different rates of change [26]. This should be considered to appropriately avoid the entanglement of functional and reconfiguration concerns [8, 10], and improve their design and maintainability. Aspect-Oriented Software Development (AOSD) [22] proposes the separation of the crosscutting concerns of software systems into separate entities called aspects. This separation avoids the tangled concerns of software, allowing the reuse of the same aspect in different entities of the software system as well as its maintenance. Although several proposals have addressed the integration of aspects in software architectures, very few of them have considered the encapsulation of the reconfiguration concern into aspects [3, 10, 15]. We consider that the separation among the functional and reconfiguration concerns is a first step to build adaptive systems easier to maintain. Thus, the reconfiguration code will be able to change the functional code without being affected. Our proposal takes advantage of AOSD techniques to improve the reconfiguration management.

This paper provides four contributions to the design of autonomous dynamically reconfigurable systems. First, it defines a model to bridge the gap among high-level reconfiguration specifications and low-level supporting mechanisms. Second, it considers the support for both reactive and proactive reconfigurations, to achieve a better level of flexibility. Third, it describes a hierarchical decentralized approach to tackle the problems of scalability present in self-adaptive approaches and maintainability in self-organizing ones. Fourth, it explicitly separates reconfiguration concerns to improve their maintainability and reuse. These ideas have been integrated in the PRISMA software architecture model, which is briefly introduced next.

3 Background: the PRISMA model

PRISMA provides a model and a language for the definition of complex software systems [30, 32]. Its main contributions are the way in which it integrates elements from aspect-oriented software development and software architecture approaches, as well as the advantages that this integration provides to software development.

Among the different Architecture Description Languages (ADLs) from the literature, the PRISMA ADL was selected because of the benefits it provides for supporting the dynamic evolution of software architectures. First, PRISMA allows modelling the functional decomposition of a system and its crosscutting concerns by using architectural elements and aspects, respectively. Thus, we can easily isolate functional and reconfiguration concerns. Second, PRISMA does not only allow modelling the structure (i.e. the architecture) of a system, but also allows describing precisely the internal behaviour of each architectural element. The behaviour is specified by using a modal logic of actions and a dialect of the polyadic π-calculus. π-calculus is used to specify and formalize the processes of the PRISMA model and mobility capabilities [2], and the modal logic of actions is used to formalize how the execution of these processes affects the internal state of aspects. Thus, since the internal behaviour is formally described, this allows us to automatically interleave the actions required to perform the runtime evolution of its instances. Lastly, the PRISMA ADL is supported by a

Model-Driven Development framework [36], which allows the automatic generation of executable code from PRISMA models/specifications [32]. This also benefits the support for dynamic reconfiguration: the code generation templates will only include reconfiguration mechanisms in the final code when needed. Next, the main concepts of the PRISMA ADL are introduced.

PRISMA introduces aspects as a new concept of software architectures rather than simulating them using other existing architectural terms (components, connectors, views, etc). Aspects are first-order citizens of software architectures and represent a specific behaviour of a concern (safety, coordination, distribution, *reconfiguration*, etc.) that crosscuts the software architecture. PRISMA has three kinds of architectural elements: components, connectors, and composites. Each architectural element encapsulates its functionality as a black box and publishes a set of services that they offer to other architectural elements through their ports. However, the internal view of these architectural elements differs between simple and composite ones.

The internal view of **components and connectors** (which are *simple* architectural elements) is an invasive composition [1] of aspects, which can be shown as a prism (see Figure 1). Each side of the prism is an aspect that the architectural element imports. A component differs from a connector in that it imports a functional aspect, whereas a connector imports a coordination aspect. Aspects are synchronized among them by means of weavings, which indicate how the execution of an aspect service can trigger the execution of services in other aspects (see Figure 6).

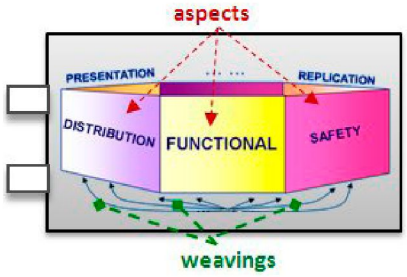


Figure 1: Internal view of *simple* PRISMA elements

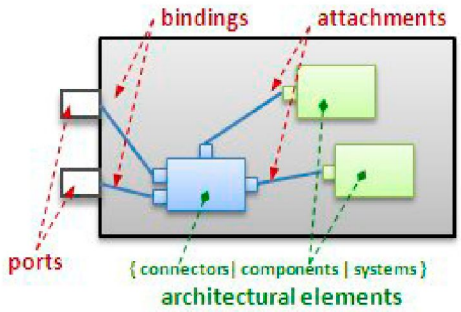


Figure 2: Internal view of *composite* elements

The internal view of **composite components** (called in the PRISMA ADL as *Systems*) consists of a set of architectural elements (components, connectors and other composites) and the links among them (see Figure 2). A link can be of two kinds: an attachment, if it links a component and a connector; or a binding, if it links an (internal) architectural element with one of the ports of the composite (i.e. allowing the communication with external architectural elements). Further details about the semantics of the PRISMA ADL can be found in [30, 32].

4 Case study: Agrobot

To illustrate our approach, we present in this section the software architecture of the *Agrobot*, an autonomous agricultural robot for plague control. Its objective is to patrol -at periodical intervals- a small field or delimited area, looking for pests or disease attacks over a set of growing crops. When a threat is detected, a pesticide is applied to, as a first counter-attack measure, and a real-time alarm is sent to the manager in order to take further specialized actions. The *Agrobot* architecture is hierarchically defined, i.e. as a system of systems. The top level, shown in Figure 3, describes the set of subsystems the robot is composed of and their interactions with each other. Each subsystem is depicted as a component, which provides and requires a set of services through its ports. Each component not only depicts the name of the instance (e.g. *LeftCamera*, see Figure 3, bottom-left), but also the name of its architectural type (e.g. *VisionSystem*), which defines the structure, behaviour and constraints of the component. The interaction among components is coordinated by different connector types (represented as blue small components), which are not detailed in this paper due to space reasons.

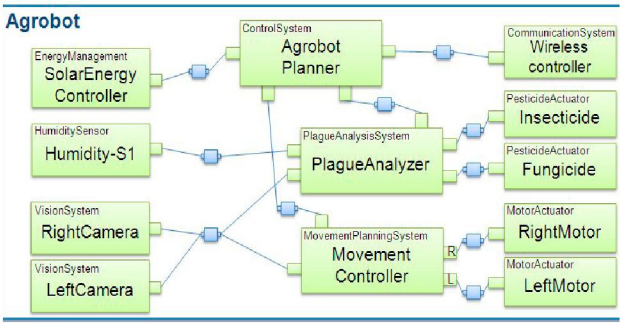


Figure 3: Software architecture of the Agrobot

Each component of the *Agrobot* architecture provides a different set of actions (e.g. image capturing, pattern analysis, movement, sensing, communication, pesticide activation, etc.), which are combined appropriately to fulfil a task (e.g. to supervise a growing crop, go to another crop, recharge energy, etc.). For instance, task planning and selection is performed by the *AgrobotPlanner* component, the energy management is performed by the *SolarEnergyController*, the communication is performed by the *WirelessController*, etc. Among the different components, we will focus on the image capturing subsystem, provided by the *RightCamera* and *LeftCamera* components. Both components are instances of the same architectural type, *VisionSystem*, but are parameterized to use a right camera

or a left camera, respectively. These components capture and pre-filter real-time images from the environment, which are used by other components to look for crop diseases (i.e. the *PlagueAnalyzer* component) or to guide the movement (i.e. the *MovementController* component). The *RightCamera* and *LeftCamera* components are composite components, i.e. their behaviour is provided by a composition of other architectural elements. They are mainly composed of a video capture component, *VideoCaptureCard*, a hardware device which captures images from the environment at a constant frame rate; and an image processing component, *ImageProcCard*, a hardware device which pre-processes the images captured. For instance, Figure 4 shows the internal structure of the *RightCamera* component: an instance of a *VideoCaptureCard* component, *Right-VCapt*, sends the captured images to an instance of an *ImageProcCard* component, *ImgProc-1*. These components are coordinated by a connector, *VCC-Conn*. The pre-processed images are sent to other subsystems by means of another connector, the *IPC-Conn* connector.

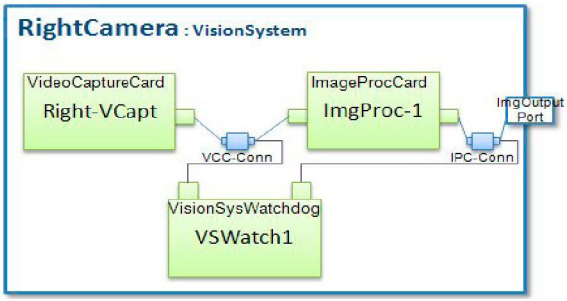


Figure 4: Architecture of the RightCamera component

Self-management is used in the *RightCamera* and *LeftCamera* components to reconfigure the internal architecture when a fault is detected in one of its components. Fault detection is performed by a watchdog component, *VisionSysWatchdog*, which periodically checks if images are being correctly captured and processed. In case misbehaviour is detected, this component sends an event to notify a failure. For instance, if the image processing component does not correctly process images or has a negative performance, then the *VisionSysWatchdog* component sends an event, which contains the name of the failing architectural type:

```
faultyOutput!(output `ImageProcCard`)
```

The events raised by the *watchdog* component will be captured by the self-management mechanisms, reacting appropriately for each kind of event. For instance, in case of an occurrence of the previously described event (i.e. *faultyOutput*), the failing component instance must be removed (i.e. the hardware image-processing device is deactivated) and another, different, component must be used instead: the *ImageProcSoftware* component. This component implements another (compatible) image processing algorithm, but with less performance than the removed one. Thus, the image capturing subsystem can continue seamlessly working.

In the next section, the self-management mechanisms that support the dynamic reconfiguration of a composite component are described in detail.

5 Autonomic reconfiguration of composite components

One of our previous works was the study and identification of the active concerns in evolution processes. As other authors also stated [4, 17, 20, 23, 28], we observed that self-managed architectures usually follow a closed control loop that periodically supervises the architecture, plans if any (corrective) change needs to be performed, and effects them. Similar control loops have been proposed to develop autonomous systems (e.g. robots), being the most extended the autonomic control loop [21], which is usually referred to as the MAPE loop (Monitor, Analyse, Plan, Execute). This loop performs control operations on a managed resource to achieve a set of predefined high-level goals, which are part of the knowledge of an autonomic (i.e. self-controlled) element. The autonomic control loop has the advantage that clearly isolates the main concerns commonly present in every process of (self-)change. Other architecture-based proposals for self-management generally merge analysis and planning, or planning and execution, or do not explicitly model the knowledge required to perform the changes.

Our proposal uses the autonomic control loop as a reference model to define how a system reconfigures itself, bridging the gap among high-level specifications (i.e. ADLs) and technology-specific (dynamic updating) mechanisms. We have adapted the original MAPE loop for this purpose: the managed resource is the architecture of a system, and the control operations performed on this resource are mainly introspection operations (for monitoring the architecture) and reconfiguration operations (for changing the architecture). Another adaptation that has been done to the original MAPE loop is its implementation by means of aspects: each one of the different controlling components (i.e. Monitor, Analyse, Plan, etc.) has been encapsulated in a different aspect. Next subsections describe the details of the approach.

5.1 Aspects for reconfiguration

Our approach defines four aspects to encapsulate the reconfiguration concerns. They are the following (see Figure 5): (i) *Monitoring*, the concern that captures the events that take place in the architecture of a composite component (i.e. the managed resource); (ii) *Reconfiguration Analysis*, the concern that analyses the different events to detect if a reconfiguration must be done, and that defines the set of reconfigurations to be performed on the architecture; (iii) *Reconfiguration Coordination*, the concern that plans/ coordinates how the reconfigurations must be applied safely to the architecture without interrupting current transactions, and (iv) *Reconfiguration Effector*, the concern that applies atomic reconfiguration operations on the running system.

Each one of these aspects will be described in the different subsections.

The reason of using aspects instead of modules for encapsulating dynamic reconfiguration behaviour is because of the advantages that AOSD provides [22], i.e. better reuse and maintenance of the different concerns. Although modules can be used to separate concerns, the invocations among different modules (e.g. procedure calls) are explicitly defined inside each module, thus making each module dependent of the other. However, in PRISMA aspects are, by definition, independent of each other: there are not invocations among aspects, but there are synchronizations among aspects. An aspect defines provided and required services, and each service is treated as a *hook* which can be intercepted. These interceptions are performed by weavings, which are defined outside the aspects and define how two aspects are bound together (i.e. synchronized). Thus, aspects are completely independent of each other: modifying an aspect will only impact the weavings that are specifically related to this aspect, but not other aspects.

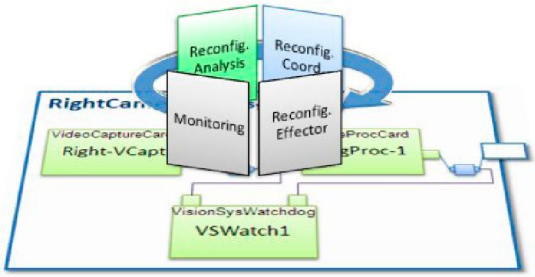


Figure 5: Aspects for autonomic reconfiguration

For instance, Figure 6 shows some of the weavings that have been defined in the VisionSystem architectural type. The first weaving intercepts the execution of the service *create-ImageProcSoftware* (provided by the Reconfiguration Analysis aspect), and replaces it with the execution of the service *createArchElement* (provided by the Reconfiguration Coordination aspect). In other words, this weaving binds the execution of a domain-specific reconfiguration service (i.e. create-ImageProcSoftware) to a generic reconfiguration service (i.e. createArchElement). This weaving will be invalidated if any of these services has its signature changed. For instance, if a parameter is removed, the weaving definition can be modified to provide a default value to the other service (or the result of applying a function). In both cases, the modification of an aspect does not necessarily impact to the other aspects. The analysis of this impact is outside the scope of this work; however other authors have conveniently addressed this problem, such as Pérez-Toledano et al. [31].

Another reason for using aspects is to avoid that changes (i.e. maintenance operations) on technology-specific reconfiguration mechanisms may have an impact on the technology-independent reconfiguration specifications, and vice versa. Each aspect has a different role in the MDD process. On the one hand, the *Reconfiguration Analysis* aspect is domain-specific: it is defined by the architect and contains the high-level specific reconfiguration policies (in terms of PRISMA concepts) for the composite component it is weaved to. On the other hand, the aspects *Monitoring* and *Reconfiguration Effector* (depicted in Figure 5 in dark grey) implement the technology-specific mechanisms that provide support for supervising/changing the architecture. They model the low-level services that are provided by the infrastructure and allow us to combine them to perform high-level reconfiguration operations. This is performed by the aspect *Reconfiguration Coordination*: it encapsulates the mappings from high-level PRISMA concepts to low-level technological services. Thus, code that has different rates of change [26] is explicitly separated: dynamic updating mechanisms (i.e. Monitoring and Effector aspects), reconfiguration specifications (i.e. Analysis aspect), and the mappings among them (i.e. Coordination aspect).

```
Weavings
  ...
  ReconfCoord.createArchElement("ImgProcSW",
       params, newID)
     insteadOf
  VisionSysRecAnalysis.create-ImageProcSoftware(
       params, newID);

  Monitoring.beforeServiceRequest(*, eventName,
       eventParams)
     insteadOf
  VisionSysRecAnalysis.beforeEvent(eventName,
       eventParams);
  ...
End_Weavings;
```

Figure 6: Example of weavings among aspects

5.1.1 The monitoring aspect

This aspect *monitorizes* the architecture of the composite component where it has been imported to. It provides a set of services for collecting information about: (i) the *events/messages* that take place in the architecture, (ii) the current *configuration* of the architecture, and (iii) the *runtime status* of the different elements of the architecture. These services are shown in Figure 7. Next, they are explained in detail.

Services 1 to 3 are provided to intercept any event triggered in the composite component and act *before*, *instead of*, or *after* the event. Since this aspect supervises the architecture of a composite component, the events that it can capture are only those that take place at the level of interactions, i.e. service requests among internal components (i.e. through attachments), or coming from/to external ports (i.e. through bindings). Thus, internal components/connectors remain unaffected by interception mechanisms (i.e. encapsulation is preserved). For instance, the service 2 (see Figure 7) intercepts a service request just before it is delivered to the service provider. The parameter *serviceName* defines the request to intercept, whereas *elemID* defines which element sent the request (an external port, an architectural instance, or a connection). As it will be described in the following section, event capturing is used to trigger reconfiguration processes.

Services 4 to 8 provide information about the current configuration of the managed architecture. Since the architecture of a dynamic reconfigurable system can change substantially over time, information about the configuration at any given moment is essential. For instance, the service 4 returns a PRISMA specification with the current configuration, so it can be analysed at runtime. The other services are auxiliary and allow us to get the instances of a particular type, the connections to a given instance, etc. In this way, a composite component can be aware of its internal configuration and use this knowledge to decide if a reconfiguration is necessary. Moreover, this information also allows us to verify whether or not a set of reconfiguration actions has been successfully executed.

Finally, service 9 provides information about the runtime status of the elements the composite component is made of: if the elements are idle, processing services or stopped. This information allows a composite component to be aware of whether its elements are ready to be reconfigured or not.

```
Monitoring Aspect
  ...
  Services
(1)  afterServiceRequest(elemID, serviceName,
        output paramList);
(2)  beforeServiceRequest(elemID, serviceName,
        output paramList);
(3)  insteadOfServRequest(elemID, serviceName,
        replacingService, output paramList);
(4)  getConfigSpecification(output PRISMASpec);
(5)  getArchElementInstances(typeName,
        output instanceList);
(6)  getConnections(archElemId,
        output connectionList);
(7)  getArchElementProperties(archElemID,
        output propertiesList, output portList);
(8)  getConnectionProperties(connID,
        output archElem1, output archElem2);
(9)  getStatus(elemID, output status);
  ...
End_Aspect;
```

Figure 7: Services of the Monitoring aspect

For instance, to be subscribed to (i.e. intercept) the event *faultyOutput* when it is triggered by the *VisionSysWatchdog* component (i.e. before the event is processed), the following code must be executed:

```
beforeServiceRequest!("VisionSysWatchdog",
        "faultyOutput", output paramList)
```

5.1.2 The reconfiguration analysis aspect

This aspect describes the *proactive* reconfiguration behaviour of a composite component. This aspect is application-specific: it is defined for a specific composite component, and contains the policies that will drive the reconfiguration of this component. The *Reconfiguration Analysis* aspect defines *when* to perform a reconfiguration, and *how* the different architectural elements must be reconfigured. We have used the PRISMA AOADL to define event-condition-action (ECA) policies. These policies are expressive enough to describe how a composite component should react in presence of certain events or conditions. In our approach, these policies are described at design-time, although they can be changed at runtime by using reflective dynamic evolution mechanisms, as described in a previous work [11]. The system architect defines ECA policies by means of *configuration transactions* (i.e. Actions) and *reconfiguration triggers* (i.e. Events and Conditions). An example of this aspect is shown in Figure 8: it shows the *Reconfiguration Analysis* aspect of the *VisionSystem* architectural type.

A **configuration transaction** is a specification that describes an ordered set of domain-specific reconfiguration operations to be executed transactionally (all or none), in order to achieve a new type-conformant configuration. Thus, reconfiguration operations will be executed, and if anything fails, the reconfiguration will be rollbacked. For instance, the transaction *RepairImageProcessUnit* (see Figure 8, transactions section) describes how the component *ImageProcCard* must be replaced by the component *ImageProcSoftware* in case of malfunction. The transaction consists of two processes. The first one (see *BEGIN* process) obtains the references to the instances that are going to be affected by the reconfiguration process. Then, the second process (see *RECONF* process) performs a set of configuration actions: creates a new instance of the *ImageProcSoftware* component, attaches this instance to the instance of the *VCC-Conn* connector, detaches the failing *ImageProcCard* instance from the *VCC-Conn* connector instance, etc.

```
ReconfigurationAnalysis aspect VisionSysRecAnalysis
  ...
  Triggers
    RepairImageProcessingUnit() when
      {eventParams==["ImageProcCard"]}
      beforeEvent!("faultyOutput", out eventParams);
    ... [more reconfiguration triggers]

  Transactions
    in RepairImageProcessingUnit():
      BEGIN::=
      // Get IDs of instances subject to changes
      oldImProcCardID=imageProcCard-list[0] →
      VCCConnID=VCC-Conn-list[0] →
      IPCConnID=IPC-Conn-list[0] → RECONF;
      RECONF::=
        create-ImageProcSoftware!(cameraPos,
           output newImProcID) →
        attach-Att_VCCConn_IPCSW!(VCCConnID,
           newImProcID, output newAttID) →
        attach-Att_IPCSW_IPCConn!(newImProcID,
           IPCConnID, output newAttID) →
        detach-Att_VCCConn_IPC!(VCCConnID,
           oldImProcCardID) →
        detach-Att_IPC_IPCConn!(oldImProcCardID,
           IPCConnID) →
        destroy-ImageProcCard!(oldImProcCardID) →
        END;

    ... [more transactions]
End_Aspect VisionSysRecAnalysis;
```

Figure 8: Example of a Reconfiguration Analysis aspect

A **reconfiguration trigger** is a condition which, if true, activates a configuration transaction. This condition may evaluate user-defined attributes (e.g. performance), or be true when a certain event is intercepted (e.g. an exception, a service request, the creation or destruction

of connections, etc.). For instance, the reconfiguration trigger shown in Figure 8 (see *triggers* section) activates the configuration transaction *RepairImageProcessUnit* when a certain event is intercepted in the architecture *and* a certain condition is fulfilled. The event to intercept is the service request *faultyOutput*, and the condition is that one of the parameters of this service is "ImageProcCard". This denotes that an instance of ImageProcCard is failing (see section 4).

Note that this aspect does not directly invoke services from other aspects. For instance, the reconfiguration trigger intercepts services by means of the service *BeforeEvent*. This service is really a hook that is bound to the Monitoring aspect by means of a weaving relationship (see the second weaving in Figure 6). Without this weaving, the service *BeforeEvent* does nothing.

5.1.3 The reconfiguration coordination aspect

This aspect is in charge of driving the successful execution of the reconfiguration plans that have been triggered by the *Reconfiguration Analysis* aspect. It ensures that these plans are transactionally performed (all or none), and that the current state of the architecture is preserved. When a reconfiguration transaction is triggered, the service *beginConfigurationTransaction* is implicitly executed. The execution of this service prepares the architecture of the composite component to be reconfigured. Then, the execution of each configuration action belonging to a configuration transaction implicitly triggers the execution of one of the generic reconfiguration services provided by the *Reconfiguration Coordination* aspect. The headers of these services are shown in Figure 9. Their behaviour is defined using the PRISMA AOADL syntax. For illustration purposes only these services for creating and destroying architectural elements are completely shown.

These generic reconfiguration services describe the set of low-level actions to perform for each different kind of reconfiguration action (i.e. creating instances, disconnecting instances, replacing instances, etc.). Each generic reconfiguration service performs three steps. First, the running transactions of the elements affected by a reconfiguration action are finished in a consistent way. For instance, the affected elements when performing the *destroyArchitecturalElement* operation are the instance to destroy, its connections, and its adjacent architectural element instances. Second, the set of required low-level changes are applied. For instance, the destruction of an instance and its connections. These low-level changes are performed by the Reconfiguration Effector aspect (see section 5.1.4). Third, when the reconfiguration has been realized, it is verified whether or not the desired configuration has been achieved, by querying to the Monitoring aspect about the configuration information (see section 5.1.1). Each generic reconfiguration service successfully executed is registered in a data structure, in order to undo the operation if anything fails.

Finally, if a reconfiguration transaction ends successfully, the service *EndConfigurationTransaction* is implicitly executed. Then, all the elements that were stopped are restarted. It only makes sense to start reconfigured elements when all the reconfiguration operations have been performed successfully. If any of the reconfiguration services fails, the configuration transaction is rollbacked.

```
ReconfigurationCoordination Aspect
  BeginConfigurationTransaction():
     ... // Initialisation of auxiliary structures
  EndConfigurationTransaction():
     CHECK::=  |transState=valid|COMMIT  +
        |transState=fail|ROLLBACK.
     COMMIT::=
        Destroy!(DestructionStack_popElement())→
     ... // [Commit and Rollback processes]
  CreateArchitecturalElement(AEType, params,
    output newID):
    CREATE::= CreateInstance!( typeof(AEType),
          params, output newID) → CHECK;
    CHECK::= CheckConsistence!(output transState) →
     |transState=fail|EndConfigurationTransaction!()
          + CONTINUE;
    CONTINUE::= ElementCreated(newID) →
          Start!(newID).
  DestroyArchitecturalElement?(id):
     STOP::= CheckConnections!(id) →
        Stop!(id) → Status!(id,status) →
        |status="Blocked"|DESTROY + STOP;
     DESTROY::= DestructionStack_pushElement(id).
  CreateAttachment(sourceArchElemID, srcPort,
     targetArchElemID, trgPort,output attID):
        [... body ommitted for space reasons ]
  DestroyAttachment(attachmentID):  [...]
  CreateBinding(sysPortName, archElemID,
     archElemPortName, out bindingID):  [...]
  DestroyBinding(bindingID):  [...]
  ReplaceArchitecturalElement(IDToBeReplaced,
     newAEType, [initializationValues], out newID):
     [...]
End_Aspect;
```

Figure 9: Services of the Reconfig. Coordination aspect

5.1.4 The reconfiguration effector aspect

This aspect *effects*, or performs, changes on the architecture it manages. It provides a set of atomic, simple reconfiguration services to interact with the other high-level aspects. These services are simple because they do not take into account the status (i.e. whether the element has been previously stopped or not) and/or the relations with the adjacent architectural elements. They must be correctly coordinated to carry out a safe reconfiguration: this is performed by the Reconfiguration Coordination aspect (see section 5.1.3). The most relevant services are shown in Figure 10.

The implementation of each reconfiguration service is technology-dependent: depending on the technology selected and how the component execution model has been implemented, the dynamic updating mechanisms to use will be different. For instance, the current implementation of the PRISMA model, PRISMANET, has been done using .NET technology and a concurrent, event-based, aspect-oriented execution model [29]. The management of connections at runtime has been done by the use of indirections and publish-subscribe mechanisms, which are implemented in ports. Among the available strategies for implementing the quiescence of

running, stateful components [19, 24, 40], finally a variation of the tranquillity approach was implemented. The support for instance replacement requires the implementation of three features: type replacement, state mapping and interface adaptation. Our current implementation only provides type replacement and state mapping, in a similar way as described by Ritzau et al. [33], but adapted for event-based, aspect-oriented components. An example of how interface adaptation can be provided is described in Cámara et al. [6].

```
ReconfigurationEffector Aspect
  ...
  Services
    StartElement(elemID);// Reach an Active status
    StopElement(elemID); // Reach a Quiescent status
    CreateInstance(componentType, initParams,
       out componentID);
    DestroyInstance(componentID);
    Connect(componentID1, port1, componentID2,
       port2, out connectionID);
    Disconnect(connectionID);
    ReplaceArchitecturalElement(ID,type,[params]);
  ...
End_Aspect;
```

Figure 10: Services of the Reconfig. Effector aspect

5.2 The evolver component: weaving the reconfiguration aspects

The previously described aspects provide autonomic reconfiguration capabilities to those composite components that import them. However, the infrastructure for supporting dynamic reconfiguration is not costless: it may introduce a performance overhead of 2% [41]. Since not all the components of a system require this degree of flexibility, and to optimize performance and system resources, the decision of which composite components will support dynamic reconfiguration or not is left to the architect. This decision is reflected by importing the reconfiguration aspects in those composite components that may undergo dynamic changes. Only when the specification of a composite component imports these aspects, the PRISMA Model Compiler [32] includes the reconfiguration mechanisms in the generated code of the composite component.

To synchronize appropriately the aspects for autonomic reconfiguration and ease their maintenance, these aspects have been encapsulated into a component called *Evolver*[2]. This component provides autonomic reconfiguration capabilities to the composite component that it has been imported to. It is integrated in the architecture of a composite component like another component, but it provides services that belong to the meta-level. That is, it offers services that introspect and change the architecture within the *Evolver* resides (i.e. a composite component).

By default, the *Evolver* only imports the aspects that support dynamic reconfigurations, i.e. *Monitoring*, *Reconfiguration Coordination* and *Reconfiguration Effector*. The activeness of change (i.e. proactive, reactive or both) is specified by the architect, depending on its needs. On the one hand, to introduce proactive reconfigurations, a Reconfiguration Analysis aspect must be defined. This is done by completing an automatically generated, empty Reconfiguration Analysis aspect with the reconfiguration policies needed. On the other hand, to allow reactive reconfigurations, two ports must be added to the Evolver: one for introspection and another for changing the architecture. The former publishes the introspection services provided by the *Monitoring* aspect (i.e. the services 4 to 8 shown in Figure 7). The latter publishes the generic reconfiguration services provided by the *Reconfiguration Coordination* aspect. These ports allow performing unanticipated reconfigurations on a composite component. These reconfigurations could be requested by another component (such as another Evolver, which would act as a *configuror* of other elements), or by the architect itself (e.g. by connecting these ports to a component that provides a user interface).

Thus, a reconfigurable composite component will have a fixed part, i.e. the Evolver, and a variable part where the Evolver will act upon, i.e. all the other components and connections of the composite component. However, this does not mean that the reconfiguration process is unconstrained. A reconfiguration is limited by the constraints defined in the type of a composite component [9]. This type defines which components can be used in the architecture and how they can be interconnected. Thus, although different instances of the same composite component reconfigure its architecture, they will always maintain type conformance, so that the overall composition is preserved.

5.3 Hierarchically decentralized evolvers

The Evolver provides a composite component with dynamic reconfiguration capabilities, which can be initiated both proactively (i.e. autonomously-driven) and reactively (i.e. externally-driven). These kinds of activeness are combined to build a *hierarchical decentralized approach* for self-management.

Each reconfigurable composite component is provided with an Evolver that proactively manages its architecture. This proactivity makes a composite component autonomous, and allows us to distribute (and *decentralize*) reconfiguration policies among the different composite components that build a system. In addition, the decentralization we propose is *hierarchical*. Since not all the reconfiguration policies are confined to a single composite component, but can span different composites, a coordination structure among different Evolvers is needed. This coordination is performed hierarchically: the Evolver of a composite component coordinates the reconfigurations of lower-level Evolvers, i.e. those that manage the reconfigurations of composite components integrated in the architecture of the upper-

[2] This name has been chosen because this component also imports other aspects, related to the dynamic evolution of architectural types. See [11] for further details.

level Evolver. For instance, the *Agrobot* system has an Evolver that manages not only the reconfiguration of the *Agrobot* architecture, but that also coordinates the reconfigurations of the composite components that compose this architecture (see Figure 11): e.g. *RightCamera, LeftCamera,* the *MovementController*, etc.

This hierarchical decentralized reconfiguration is supported by means of reconfiguration goals, reactive reconfiguration ports (i.e. introspection and reconfiguration ports), and reconfiguration events. The details of this approach are described below.

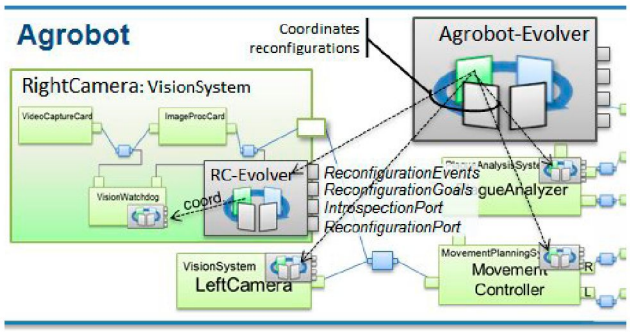


Figure 11: Agrobot and the coordination of Evolvers

5.3.1 Hierarchical change coordination

Although reconfigurable composite components may proactively reconfigure themselves, in certain cases these changes cannot be only performed locally. This is the case when reconfigurations impact several composite components simultaneously. For instance, the introduction of a new image encoding algorithm in the *Agrobot* will not only impact the image capturing subsystem (i.e. the *VisionSystem*), but also those components that decode and analyse the images captured (e.g. the *PlagueAnalyzer*). In these cases, changes must be done in a coordinated manner among the different composite components to preserve the architecture consistency. Otherwise, a *VisionSystem* component may produce images that other subsystems would be unable to decode. In our approach, this coordination of changes is performed hierarchically: the Evolver of a composite component (i.e. the upper-level Evolver) drives the reconfiguration of other composite components, through their respective Evolvers. This can be done in two ways: non-intrusively or intrusively.

Non-intrusive reconfigurations are driven by changing the **reconfiguration goals** of reconfigurable composite components. These goals are provided by the Evolver of a composite component to allow its upper-level Evolver to set reconfiguration preferences or to initiate internal proactive reconfigurations. A reconfiguration goal is an attribute defined by the architect in the *Reconfiguration Analysis* aspect that: (1) is externally visible and modifiable, and (2) is evaluated in either: (i) a *reconfiguration trigger*, to determine if a configuration transaction must be initiated; or (ii) a *configuration transaction*, to decide how a reconfiguration must be performed.

For instance, the Evolver of a *VisionSystem* composite component provides a reconfiguration goal to define the minimum performance that the *VisionSystem* must provide. Depending on the value of this goal, certain reconfigurations will be done or not. This goal is set by means of an attribute called *min_frame_rate*, which defines the minimum rate for producing images. This attribute is evaluated to decide whether a reconfiguration should be initiated to increase performance or, by the contrary, to release resources. To increase performance, the Evolver instantiates additional image processing components, whereas to release resources removes them and disables the watchdog component (thus decreasing reliability). This way, the upper-level Evolver, i.e. the *Agrobot* Evolver, can drive how the reconfiguration of the *VisionSystem* should be performed: preserving performance or reliability.

The advantage of using reconfiguration goals is that they allow us to drive the reconfiguration of a composite component without breaking its encapsulation, i.e. without directly accessing its internal composition. However, the disadvantage is that only anticipated reconfigurations can be done (i.e. those defined by reconfiguration goals). Unanticipated changes, such as the addition of a new component to a composite component, must be done intrusively. This is done through **reactive reconfiguration ports**. These ports are provided by the Evolver of a composite component to allow externally-driven reconfigurations (see section 5.2). In this context, these ports are used to allow an upper-level Evolver to explicitly introspect and change the internal composition of a composite component. Moreover, since reconfiguration services are internally provided by the *Reconfiguration Coordination* aspect, transactional reactive reconfiguration support is also provided: even all the changes externally requested are successfully executed, or all the changes are undone.

This way, an upper-level Evolver can reconfigure in a coordinated way the internal composition of different reconfigurable composite components. A coordinated change can be also transactionally performed. Each reconfiguration transaction initiated in a reactive port is considered as a subtransaction of the coordinated change transaction. If a subtransaction fails (i.e. a set of reconfigurations cannot be performed inside a composite component), then the coordinated change transaction can be entirely aborted, by deferring the commits of each subtransaction until the end of the coordinated change process.

Note that both non-intrusive and intrusive reconfigurations can be performed in a composite component if and only if its Evolver has enabled them (i.e. by exporting reconfiguration goals or reactive reconfiguration ports, respectively). This way, the architect of a reconfigurable composite component has a great level of flexibility to determine whether a composite component can be managed from outside or not, and how it can be managed.

5.3.2 Bottom-up change notifications

Another functionality that is provided by an Evolver component is the notification of changes to its upper-level Evolver. This is needed when the Evolver of a composite component has initiated changes that may impact the upper-level, i.e. the architecture where the composite component is located. For instance, consider the removal of an internal component whose functionality was being exported to other elements (e.g. the removal of the *ImageProcCard* component in the *VisionSystem*, due to a failure). These changes must be notified to its upper-level Evolver, so it can initiate additional actions to preserve architecture consistency: disabling the *VisionSystem* instance that has reduced its functionality. These changes are notified by means of *reconfiguration events*.

A **reconfiguration event** is used to communicate internal changes to outside, and has the following signature: `ReconfigurationEvent(type, message)`. The *message* parameter gives a descriptive code about the reconfiguration performed. The *type* parameter describes the impact of change, i.e. what kind of change is going to be performed: (i) *local*, an internal change: the existing interfaces remain unchanged; (ii) *medium*, a conservative change: new interfaces are added, or existing interfaces are extended with new services (i.e. existing interactions are still valid, but additional functionality is provided); and (iii) *system-wide*, a potentially disruptive change: existing interfaces are deleted, or some services removed.

Reconfiguration events can be triggered by the Reconfiguration Coordination or the Reconfiguration Analysis aspects. The Reconfiguration Coordination aspect triggers a reconfiguration event automatically when an external port or a binding to an internal component are added, changed, or removed. The reason is that external ports and binding are the means by which a composite component interacts with its environment. If an internal change impacts a port or a binding, this change will also impact the environment, so it must be notified. The Reconfiguration Analysis aspect may also trigger reconfiguration events to notify about a situation or reconfiguration performed. This is specified by the architect in proactive specifications. For instance, in a *VisionSystem* composite component, the *VideoCaptureCard* component is a critical element. If this element fails, and since the *VisionSystem* cannot perform its functionality, then the environment (i.e. the Agrobot architecture) must be notified about. This is specified in the Reconfiguration Analysis: when the event `faultyOutput?(`VideoCaptureCard`)` is intercepted, then the following event is triggered: `ReconfigurationEvent!( "system-wide", "VIDEOCARD FAILURE")`. This event will be captured by the upper-level Evolver, which will disable the composite component that has triggered this event to avoid processing its results. Thus, although one *VisionSystem* composite component failed, the robot would be able to continue working, because it is provided with two replicas of this component.

6 Related work

In the last years, a lot of research efforts have been done to address the dynamic evolution of software systems [5, 25, 34] and the reconfiguration of software architectures [4,16,19,24]. Some works have addressed the integration of AOSD techniques in software architectures [13, 32], although most of them have been mainly focused on modelling the separation of concerns at the architectural level. Only a few proposals have explicitly addressed the use of aspects to separate the evolution concerns in software architectures. AO-Plastik [3] isolates the reconfiguration concern by using aspectualized components and connectors to encapsulate the reconfiguration specifications. SAFRAN [15] has extended the FRACTAL component model to introduce adaptation aspects, which decouple reconfiguration from functional concerns. However, these approaches do not take into account all the concerns involved in the autonomous control loop, such as monitoring and effecting changes. Greenwood and Blair [20] proposed the use of dynamic aspects for monitoring and effecting changes. However, this work is focused on a particular technology whereas our approach is based at the architecture level in a MDD context.

There are many ADLs that provide dynamic reconfiguration support through specific language primitives, such as Gerel [16], Darwin [24], LEDA [7] or PiLaR [12]. These primitives are used in component specifications to describe when and how the architecture should be reconfigured. However, these works only focus on reconfiguration specifications but do not address how these specifications are finally applied on the architecture. In addition, their functional specifications are tangled with reconfiguration specifications. Several architecture-based approaches that provide self-adaptation capabilities have emerged [28]. Dashofy et al. [14] and the Rainbow framework [17] describe an architecture-based approach to provide self-healing and self-adaptation of running systems, respectively. However, both approaches use external and centralized reconfiguration mechanisms instead of using localised mechanisms to each composite component.

Morrison et al. [27] describe a conceptual framework where evolvable systems are structured in Evolver-Producer pairs (E-P). A *Producer* is a process that carries out productive functionality. An *Evolver* is a process that monitors the Producer and/or environmental stimulus, and uses this information to generate a new version of the Producer or even the *locus* (i.e. the context) where the E-P pair is located. These concepts are recursively applied to build composite systems: both an Evolver and a Producer may be internally composed of an E-P pair. Our approach shares several ideas with this conceptual framework: (i) a composite component is the locus where an E-P pair is located; (ii) the architectural elements composing a composite component represent a Producer process; and (iii) the Evolver component of a composite component behaves as an Evolver process (i.e. it can change the entire locus or generate a new version of the Producer). Another similarity with our work is that each

locus is provided with localised reconfiguration capabilities, explicitly isolating functionality from evolution. However, the framework is only conceptual, the high-level mechanisms for change are not described, and coordination issues among evolvers are not addressed.

7 Conclusion and future work

This paper has described an approach for supporting the autonomic reconfiguration of hierarchical software architectures. Instead of using a centralized self-management infrastructure to supervise the entire system and its subsystems, a hierarchical decentralized approach is proposed. Each subsystem (i.e. a composite component): (i) manages its internal reconfiguration independently of other subsystems, and (ii) provides reconfiguration events and goals to its upper level (i.e. the architecture within which it is used), to allow its integration and management. The upper level then: (i) uses these events to be informed about changes which may affect other elements, and (ii) according to the new situation, it reconfigures its architecture and/or changes the reconfiguration goals of components to fit the new needs. This approach can be recursively applied, because the same set of aspects is used at each level (i.e. *Monitoring*, *Reconfiguration Coordination* and *Reconfiguration Effector* aspects). Only the architecture-specific aspect (i.e. the *Reconfiguration Analysis* aspect) changes at each level, because the context to manage (i.e. the architecture) is different. Thus, this approach provides a software architecture with the following properties: (i) flexibility, due to the use of dynamic reconfiguration mechanisms; (ii) maintainability, because aspect-oriented techniques are used to separate reconfiguration concerns from other concerns, and (iii) scalability, because management is decentralized.

Further works remain, as the dynamic generation of reconfiguration plans from high-level goals. We have used the PRISMA AOADL to define simple ECA policies, although other kind of approaches may be used, such as those related to the synthesis of tasks from high-level goals [38]. Our contribution is not the definition of the reconfiguration specification, but the explicit separation between the reconfiguration specifications and the mechanisms that support them. This way, business logic, reconfiguration specifications, and reconfiguration mechanisms can be maintained separately. The business logic can be dynamically changed by reconfiguration specifications, by means of reconfiguration mechanisms. And reconfiguration specifications can also be dynamically changed by using the reconfiguration mechanisms, treating them as any other concern of the system, as we stated in [11].

References

1. Aßmann, U.: *Invasive Software Composition*. Springer, 2003.
2. Ali, N., Ramos, I., Solís, C.: *Ambient-PRISMA: Ambients in mobile aspect-oriented software architecture*. Journal of Systems and Software 83(6): 937-958, 2010.
3. Batista, T., Tadeu, A., Coulson, G., et al.: On the Interplay of Aspects and Dynamic Reconfiguration in a Specification to Deployment Environment. In: *2nd European Conf. on Software Architecture*. LNCS, vol. 5292. Springer, 2008.
4. Bradbury, J.S., Cordy, J.R., Dingel, J., Wermelinger, M.: A Survey of Self-Management in Dynamic Software Architecture Specifications. In: *Workshop on Self-Managed Systems*. Newport Beach, CA, 2004.
5. Buckley, J., Mens, T., Zenger, M., Rashid, A., Kniesel, G.: Towards a taxonomy of software change. *Journal of Software Maintenance and Evolution*, 17(5). Wiley, 2005.
6. Cámara, J., Salaün, G., Canal, C.: Composition and Run-time Adaptation of Mismatching Behavioural Interfaces. *J. of Universal Computer Science*, 14(13), Springer, 2008.
7. Canal, C., Pimentel, E., Troya, J.M.: Specification and Refinement of Dynamic Software Architectures. In: *Working IFIP Conference on Software Architecture (WICSA'99)*. San Antonio, Texas, USA, 1999.
8. Cazzola, W., Chiba, S., Saake, G.: Guest Editors' Introduction: Aspects and Software Evolution. *Transactions on Aspect-Oriented Software Development*, 4: 114-116. Springer, 2007.
9. Costa-Soria, C., Heckel R.: Modelling the Asynchronous Dynamic Evolution of Architectural Types. In: *Self-Organizing Architectures*. LNCS, vol. 6090, pp. 198-229. Springer-Verlag, Berlin Heidelberg, July 2010.
10. Costa-Soria, C., Pérez, J., Carsí, J.A.: Handling the Dynamic Reconfiguration of Software Architectures Using Aspects. In: *13th European Conf. on Software Maintenance and Reengineering*. Kaiserslautern, Germany, 2009.
11. Costa-Soria, C., Hervás-Muñoz, D., Pérez, J., Carsí, J.A.: A Reflective Approach for Supporting the Dynamic Evolution of Component Types. In: *14th Int. Conf. on Engineering of Complex Computer Systems (ICECCS'09)*. 2-4 June 2009.
12. Cuesta, C.E., Romay, P., Fuente, P., Barrio-Solórzano, M: Reflection-Based Aspect-Oriented Software Architecture. In: *European Workshop on Software Architecture (EWSA'04)*. LNCS, vol. 3047. Springer, 2004.
13. Cuesta, C.E., Romay, P., Fuente, P.d.l., Barrio-Solárzano, M.: Architectural aspects of architectural aspects. In proc. of: *2nd European Workshop on Software Architecture (EWSA'05)*. LNCS, vol. 3527. Springer, 2005.
14. Dashofy, E.M., van der Hoek, A., Taylor, R.N.: Towards Architecture-Based Self-Healing Systems. In: *Workshop on Self-Healing Systems*. Charleston, South Carolina, 2002.
15. David, P., Ledoux, T.: An Aspect-Oriented Approach for Developing Self-Adaptive Fractal Components. *5th Symp. on Software Composition (SC'06)*. Vienna, Austria, 2006.
16. Endler, M., Wei, J.: Programming Generic Dynamic Reconfigurations for Distributed Applications. In: *First International Workshop on Configurable Distributed Systems*. London, UK, 1992.
17. Garlan, D., Cheng, S., Huang, A., et al. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. *Computer*, 37:46-54. IEEE, 2004.
18. Georgiadis, I., Magee, J., Kramer, J.: Self-organising software architectures for distributed systems. In: *Workshop on Self-Healing Systems*. Charleston, South Carolina, 2002.
19. Gomaa, H., Hussein, M.: Software reconfiguration patterns for dynamic evolution of software architectures. *4th Int. Conf on Software Architecture (WICSA'04)*. IEEE, 2004.
20. Greenwood, P., Blair, L.: A Framework for Policy Driven Auto-adaptive Systems Using Dynamic Framed Aspects. *Transactions on AOSD II*. LNCS, vol. 4242, pp. 30-65. Springer, 2006.
21. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. *Computer*, 36(1):41-50. IEEE, 2003.

22. Kiczales, G., Lamping, J., Mendhekar, A., et al.: Aspect-Oriented Programming. In *11th'ECOOP'97*.
23. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: *ICSE - Future of Software Engineering (FOSE'07)*, pp. 259–268. IEEE, 2007.
24. Kramer, J., Magee, J.: The Evolving Philosophers Problem: Dynamic Change Management. *Transactions on Software Engineering*, 16(11):1293-1306. IEEE, 1990.
25. McKinley, P.K., Sadjadi, S., Kasten, E., et al.: Composing Adaptive Software. *Computer*, 37(7). IEEE, 2004.
26. Mens, T., Wermelinger, M.: Separation of concerns for software evolution. *Journal of Software Maintenance and Evolution*, 14(5):311-315. Wiley, 2002.
27. Morrison, R., Balasubramaniam, D., Kirby, G., et al.: A Framework for Supporting Dynamic Systems Co-Evolution. *Automated Software Engineering*, 14(3):261-292. Springer, 2007.
28. Oreizy, P., Gorlick, M., Taylor, R.N. et al: An Architecture-Based Approach to Self-Adaptive Software. *Intelligent Systems*, 14:54-62. IEEE, 1999.
29. Pérez, J., Ali, N., Costa, C., et al.: Executing Aspect-Oriented Component-Based Software Architectures on .NET Technology. In: *3rd International Conference on .NET Technologies*. Pilsen, Czech Republic, June 2005.
30. Pérez, J: *PRISMA: Aspect-Oriented Software Architectures*. PhD Thesis, Universidad Politécnica de Valencia, 2006.
31. Perez-Toledano, M.A., Navasa, A., Murillo, J.M., Canal, C.: TITAN: a Framework for Aspect Oriented System Evolution. In: *International Conference on Software Engineering Advances (ICSEA'07)*. IEEE, 2007.
32. Pérez, J., Ali, N., Carsí, J.A., et al.: Integrating aspects in software architectures: PRISMA applied to robotic tele-operated systems. *Information & Software Technology*, 50(9-10):969-990. Elsevier, 2008.
33. Ritzau, T., Andersson, J.: Dynamic Deployment of Java Applications. In: *Java for Embedded Systems Workshop*. London, 2000.
34. Segal, M.E., Frieder, O: On-the-Fly Program Modification: Systems for Dynamic Updating.*IEEE Software*, 10(2) 1993.
35. Software Engineering Institute: *Ultra-Large-Scale Systems: Software Challenge of the Future*. Technical Report. Carnegie Mellon University, Pittsburgh, USA, 2006.
36. Selic, B.: The pragmatics of model-driven development. *Software*, 20(5). IEEE, 2003.
37. Serugendo, G.D.M., Gleizes, M.P., Karageorgos, A.: Self-organisation and emergence in MAS: An Overview. *Informatica (Slovenia)*, 30(1):45-54. 2006.
38. Sykes, D., Heaven, W., Magee, J. et al.: From goals to components: a combined approach to self-management. *Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'08)*. Germany, 2008.
39. Taylor, R.N., Medvidovic, N., et al.: *Software Architecture: Foundations, Theory and Practice*. Wiley, 2009.
40. Vandewoude, Y., Ebraert, P. et al.: Tranquillity: A low Disruptive Alternative to Quiescence for Ensuring Safe Dynamic Updates. *Transactions on Software Engineering*, 33(12):856-868. IEEE, 2007.
41. Wang, Q., Shen, J., Wang, X., Mei, H.: A Component-Based Approach to Online Software Evolution. *J. of Software Maintenance and Evolution*, 18(3). Wiley 2006.

Component Reconfiguration in Presence of Mismatch

Carlos Canal and Antonio Cansado
Department of Computer Science, University of Málaga, Spain
E-mail: canal@lcc.uma.es

Keywords: component substitution, dynamic reconfiguration, software adaptation

Received: February 20, 2010

This paper discusses how to reconfigure systems in which components present mismatch in both their signature and behavioural interfaces. We are interested in performing component substitution without stopping the system, though we assume components are not designed with reconfiguration capabilities in mind. We also consider that components may need to be adapted before interacting with the system. In this work we identify the basic requirements for achieving runtime component substitution, and define several different interchangeability notions that are adequate to component substitution under behavioural adaptation. Our approach is illustrated with a case-study of a client/server system where the server needs to be substituted by a new one. Classic equivalence and compatibility notions fail to find a new server because the only one available implements a different interface. We show how our interchangeability notions could be used in order to let the system keep on working.

Povzetek: Opisano je preoblikovanje sistemov, ko se zgodi neskladje.

1 Introduction

Software reuse is of great interest because it reduces costs and speeds up development time. Indeed, a vast number of software components are already available through the Internet, and many research and development efforts are being invested in devising techniques for combining them safely and efficiently. In particular, Software Adaptation promotes the use of adaptors in order to compensate mismatch among component interfaces. In fact, this is the only known way to adapt off-the-shelf components since designers usually only have access to their public interfaces. Without adaptation, components could not be put together or their execution could lead to deadlocking scenarios [2, 9].

Still, one of the most challenging issues in Software Adaptation is that systems need to adapt to environmental changes, server upgrades or failures, or even the availability of a new component more suitable to be used in the system. Indeed, the need for finding a new component to be integrated in the system may be either reactive or proactive. The reactive case is caused by the system itself. For instance as a consequence of connection loss or failure of one its components, thus creating a hole in the system that must be filled for its correct functioning. The proactive case would be caused by the availability of a new component that is suspected to be a good candidate for being integrated in the system, replacing some of its current components. In both cases, we have first to detect the need for reconfiguration by using runtime monitoring techniques both on the system and on its environment. Then, the interface of the candidate components —and its compatibility with the rest of the system— must be evaluated, attending not only to its signature interface (names of services, operations, messages, etc.), but also to its behavioural interface (the order in which the elements in the signature interface are expected to be used) and the QoS features provided/expected by the component and the system.

When dealing with this kind of dynamic reconfiguration [14], component substitution must be applied without stopping the complete system, and trying to affect minimally its performance, in particular the functioning of those of its parts that are not directly involved in the reconfiguration. That means that components must collaborate to support reconfiguration capabilities. In fact, it is important to determine when the system can be reconfigured and which kind of properties the system holds after reconfiguration.

Few works have studied the interplay of behavioural adaptation and reconfiguration so far. In most approaches to reconfiguration, substituting a component by another one requires the new component to implement the same functionality as the former one. This means that substitution is usually limited to instances (or subtypes) of a given component. However, it is possible that a component cannot substitute another one, but an adapted version can.

This paper identifies some basic requirements for runtime component substitution and we describe the operations required to achieve this reconfiguration. We also define several different interchangeability notions that are well fitted for component substitution under behavioural adaptation. The paper is structured as follows: Firstly, Section 2 provides some background on behavioural interfaces and adaptation. Then, Section 3 introduces a client/server system that is used as running example through all this document. Section 4 presents our reconfiguration model, for describing systems as a collection of static ar-

chitectural views (configurations), and reconfiguration operations for moving from one configuration to another one; it also shows how reconfiguration states can be defined at certain points of system execution, and how new components must be initialised for arriving to these states. Next, Section 5 defines different notions of substitutability that we believe are adequate for component replacement under behavioural adaptation. Then, Section 6 outlines the platform that we plan to implement for validating our results. Finally, Section 7 presents related works on reconfiguration and behavioural adaptation, and Section 8 concludes the paper.

This paper builds on our previous work in the field. It is an extension of our position paper [10], developing the ideas presented there, adding many explanations and more detailed examples. In presents also our model for dynamic reconfiguration, and the notions of substitutability discussed in [10] are formally defined here.

2 Background

We assume that component interfaces are equipped both with a signature (set of required and provided operations), and a protocol. For the protocol, we model the behaviour of a component as a Labelled Transition System (LTS). The LTS transitions encode the actions that a component can perform in a given state. For reasons of space we omit the signature interface when it can be easily inferred from the corresponding protocol.

Definition 1. [LTS]. *A Labelled Transition System (LTS) is a tuple* $\langle S, s_0, L, \rightarrow \rangle$ *where* S *is the set of states,* $s_0 \in S$ *is the initial state,* L *is the set of labels or alphabet,* $\rightarrow$ *is the set of transitions :* $\rightarrow \subseteq S \times L \times S$. *We write* $s \xrightarrow{\alpha} s'$ *for* $(s, \alpha, s') \in \rightarrow$.

Communication between components are represented using *actions* relative to the emission and reception of messages corresponding to operation calls, or internal actions performed by a component. Therefore, in our model, a *label* is either the internal action τ or a tuple (M, D) where M is the message name and D stands for the communication direction ($!$ for emission, and $?$ for reception).

LTSs are adequate as far as user-friendliness and development of formal algorithms are concerned. However, higher-level behavioural languages such as process algebras can be used to define behavioural interfaces in a more concise way. We can use for that purpose the part of the CCS notation restricted to sequential processes, which can be translated into LTS models: $P ::= 0 | a?P | a!P | \tau.P | P1 + P2 | P/L | A$, where 0 denotes a do-nothing process; $a?P$ a process which receives a and then behaves as P; $a!P$ a process which sends a and then behaves as P; $\tau.P$ a process which performs an internal action τ and then becomes P; $P1 + P2$ a process which may act either as $P1$ or $P2$; P/L is the process P after hiding the names in L, preventing any communication on those names; and A denotes the call to a process defined by an agent definition equation $A = P$. Additionally, we will use the parallel operator $||$ for representing the composition of components —represented by CCS processes— allowing the synchronisation of their input and output actions.

In this paper we will use LTSs or CCS expressions indistinctly for representing components and adaptors. Both could be easily obtained for standard notations such as WS-BPEL or WWF.

2.1 Specification of adaptation contracts

Adaptors can be automatically generated based on an abstract description of how mismatch can be solved. This is given by an *adaptation contract* ($\mathscr{AC}$). In this paper, the adaptation contract between components is specified using *vectors* [8]. Each action appearing in a vector is executed by one of the components, and the overall result corresponds to a loose synchronisation between all of them. A vector may involve any number of components and does not require interactions to occur on the same names of actions. For distinguishing between actions with the same name occurring on different components, we prefix actions with component names.

For example, $\langle C1.on!, C2.activate? \rangle$ is a vector denoting that the action $on!$ performed by component $C1$ corresponds to action $activate?$ performed by component $C2$. This does not mean that both actions have to take place simultaneously, nor one action just after the other; for the transmission of $C1$'s action $on!$ to $C2$ as $activate?$, the adaptor will take into account the behaviour of these components as specified in their LTS, accommodating the reception and sending of actions to the points in which the components are able to perform them (Fig. 1).

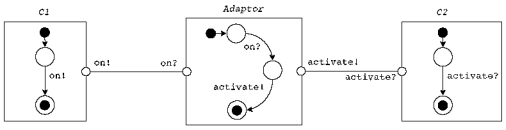


Figure 1: Components $C1$ and $C2$ connected through an adaptor.

2.2 Adaptor generation

Thus, previously to the reconfiguration of the system by the integration of a new component, we will likely need to adapt the component for solving the problems of compatibility detected in the component discovery phase. This will be accomplished by generating an adaptor, that will play the role of wrapper or component in-the-middle, filtering the interactions between the component and the system and ensuring both a correct functioning of the system (verifying for instance the absence of deadlocks or other user defined properties) and the safety of the composition (*i.e.*, that the component is behaving as stated on its interface). In previous works we have developed a methodology

for behavioural adaptation (see [9], where our approach for generating adaptors is presented). Following this methodology, both contract specification and adaptor generation are tool supported [8].

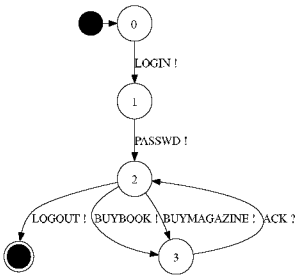
3 Running example

This section presents the running example used throughout the paper. It consists of a client/server system in which the server may be substituted by an alternative server component. This may be necessary in case of server failure, or simply for a change in the client's context or network connection that made unreachable the original server. Suppose that the client wants to buy books and magazines as shown in its behavioural interface in Fig. 2(a). The server A can sell only one book per transaction (see Fig. 2(c)); on the other hand, the server B can sell a bounded number of books and magazines (see Fig. 3(b)). In both cases, sales are represented by a pair of actions (one order and its acknowledgement), and with these two actions we abstract all the details of payment and shipment.

Initially, the client is connected to the server A; we shall call this configuration c_A. The client and the server agree on an adaptation contract $\mathscr{AC}_{C,A}$ (see Fig. 2(b)), which establishes action correspondences between the client and the server A. Obviously, under configuration c_A the client can buy at most one book in each transaction but it is not allowed to buy magazines because this is not supported by the server A. The latter is implicitly defined in the adaptation contract (Fig. 2(b)) since there is no vector allowing the client to perform the action *buyMagazine*!. Finally, the server A does not send the acknowledgement *ack*? expected by the client; this must be worked out by the adaptor, too (see v_4 in Fig. 2(b)).

In an alternative configuration c_B the client is connected to the server B whose protocol is depicted in Fig. 3(b). Similarly, the client and the server agree on an adaptation contract $\mathscr{AC}_{C,B}$ (see Fig. 3(a)). Under configuration c_B, the client can buy a bounded number of books and magazines. In Fig. 3(a), we see that vector v_5 allows the client to buy magazines. Moreover, the server B sends a different acknowledgement for each product (see v_4 and v_6 in Fig. 3(a)).

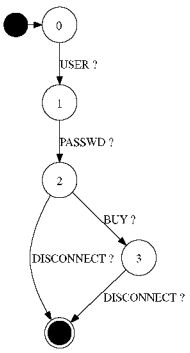
Following the methodology for behavioural adaptation presented in [9], adaptors can be automatically generated for configurations c_A and c_B (see adaptors $A_{C,A}$ and $A_{C,B}$ in Fig. 4). These adaptors ensure by construction that the interaction between the client and servers A or B will take place without deadlock and fulfilling the correspondences of actions described in the corresponding adaptation contracts [9].



(a) LTS of Client C

$v_1 = \langle C.login!, A.user? \rangle$
$v_2 = \langle C.passwd!, A.passwd? \rangle$
$v_3 = \langle C.buyBook!, A.buy? \rangle$
$v_4 = \langle C.ack?, A.\varepsilon \rangle$
$v_5 = \langle C.logout!, A.disconnect? \rangle$

(b) Adaptation Contract $\mathscr{AC}_{C,A}$

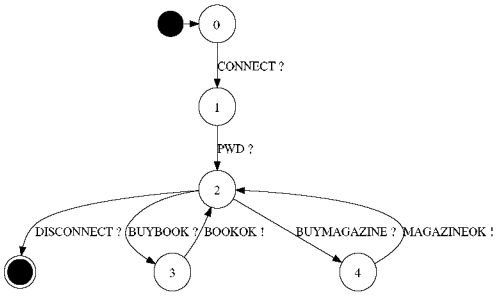


(c) LTS of Server A

Figure 2: Configuration c_A.

$v_1 = \langle C.login!, B.connect? \rangle$
$v_2 = \langle C.passwd!, B.pwd? \rangle$
$v_3 = \langle C : buyBook!, B : buyBook? \rangle$
$v_4 = \langle C.ack?, B.bookOk! \rangle$
$v_5 = \langle C.buyMagazine!, B.buyMagazine? \rangle$
$v_6 = \langle C.ack?, B.magazineOk! \rangle$
$v_7 = \langle C.logout!, B.disconnect? \rangle$

(a) Adaptation Contract $\mathscr{AC}_{C,B}$



(b) LTS of Server B

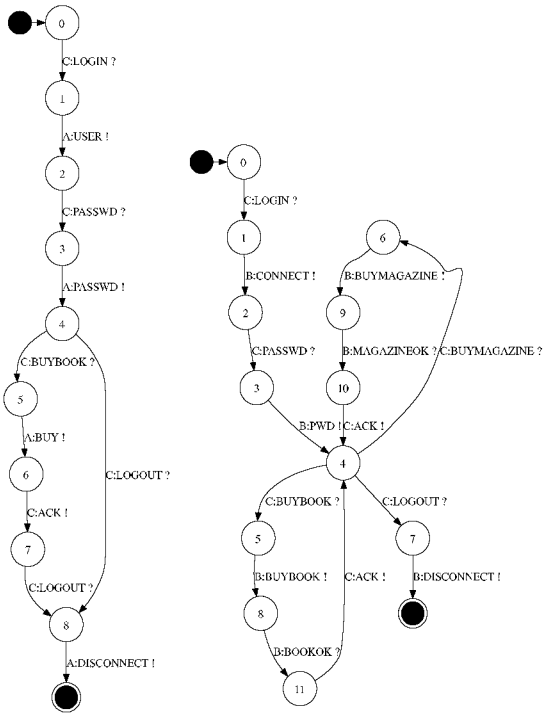
Figure 3: Configuration c_B.

4 Reconfiguration model

This section presents the model that enables both reconfiguration and behavioural adaptation. We define a *reconfiguration contract* to determine how the system may evolve in terms of structural changes.

First, a system architecture consists of a finite number of components. Each *configuration* is a subset of these components connected together by means of adaptation contracts.

Definition 2. [Configuration]. *A configuration* $c = \langle P, \mathscr{AC}, S^\star \rangle$ *is a static structural representation*



(a) Adaptor $A_{C,A}$ (b) Adaptor $A_{C,B}$

Figure 4: Adaptors

of a system's architecture. P is an indexed set of components. $\mathscr{AC}$ is an adaptation contract of components in P. $S^\star$ is a set of reconfiguration states *defined upon P; these are the states in which reconfiguration is allowed.*

Changing a configuration by another is what we call a reconfiguration. A reconfiguration is specified in a *reconfiguration contract* which separates reconfiguration concerns from the business logic of the system. This way, each configuration can be thought of as a static view of the system, while its dynamic view is specified by a reconfiguration contract.

Definition 3. [Reconfiguration Contract]. *A reconfiguration contract $\mathscr{R} = \langle C, c_0, \rightarrow_{\mathscr{R}} \rangle$ is defined as:*

C is a set of static configurations, $c_0 \in C$ is the initial configuration. $\rightarrow_{\mathscr{R}} \subseteq C \times R_{op} \times C$ is a set of reconfiguration operations, with reconfiguration operation $R_{op} \subseteq S_i^\star \times S_j^\star$, $S_i^\star \in c_i, S_j^\star \in c_j, c_{i,j} \in C$.

From the definition above, reconfiguration can only take place at predefined states, for guaranteeing system consistency. One certain state of the source configuration ($s_i^\star$) defines when an architecture can be reconfigured. On the other hand, one state of the target configuration ($s_j^\star$) says what is the starting state in the target configuration to resume the execution. Notice also that the target configuration may require a new adaptation contract (allowing replacing a component by another one that implements a different behavioural interface).

Example. In our running example, there are two configurations:
$c_A = \langle \{C, A\}, \mathscr{AC}_{C,A}, S_A^\star \rangle$, and
$c_B = \langle \{C, B\}, \mathscr{AC}_{C,B}, S_B^\star \rangle$.
The reconfiguration contract $\mathscr{R} = \langle C, c_A, \rightarrow_{\mathscr{R}} \rangle$ is given by:
$C = \{c_A, c_B\}$, and $\rightarrow_{\mathscr{R}} = \{c_A \xrightarrow{r} c_B\}$, with
$r = (s_A^\star, s_B^\star)$.
Hence, r specifies pairs of *reconfiguration states* on which reconfiguration can be performed. Since both servers have different behavioural interfaces, it is not straight-forward to determine how reconfiguration can take place after a transaction between the client and the server has started.
In the simplest scenario, we may consider that reconfiguration from c_A to c_B and back is only allowed at the initial states of the client and the server. This is specified as a unique reconfiguration state $s_i^0 \in S_i^\star, i \in \{A, B\}$ for each configuration, (where $s_A^0 = \{C.s_0, A.s_0\}$ and $s_B^0 = \{C.s_0, B.s_0\}$), and a pair of reconfiguration operations $c_A \xrightarrow{r_{A,B}} c_B$ and $c_B \xrightarrow{r_{B,A}} c_A$, with $r_{A,B} = \{s_A^0, s_B^0\}$ and $r_{B,A} = \{s_B^0, s_A^0\}$ (subindexes in states always refer to state numbers as depicted in Figs. 2 and 3).

In the next section, we will study how other pairs of reconfiguration states —apart from the initial states here— can be obtained.

4.1 Reconfiguration at runtime

In the previous section we have presented our reconfiguration model considering that reconfiguration could be only performed at the initial state of the system (i.e. at static time). Now we will generalise our working scenario allowing reconfiguration to occur when the interactions have already started and the components are in intermediate states (i.e. at dynamic time).

Interactions already performed with the component being substituted cannot be merely ignored; they must be either reproduced up to an equivalent state with the new component, transparently to the rest of the system, or rolled back and compensated when the reproduction of the state is not possible. Both fault-tolerance algorithms, exception handling and roll-back techniques must be developed to this effect, and compensation procedures must be defined when the initiated interactions cannot be correctly finished.

Example. In our running example, if the login phase has already been performed with the system in configuration c_A, and then we need to move to configuration c_B, the server B should be initialised such that the client does not need to re-log in the system. Suppose that the client C has performed a trace $\{login!, passwd!\}$: Then, the initialisation trace for the server B would be $\{connect?, pwd?\}$. Once the server B is initialised as indicated, the system can be reconfigured in order to use the new component. The substitution of the server A by the server B does not affect the client C in the sense it does not need to re-log in the system. In fact,

the client continues working on transparently, though it is warned that the adaptation contract has changed.
This way, we have implicitly defined two new reconfiguration states: $s^2_A = \{C.s_2, A.s_2\}$ for configuration c_A and $s^2_B = \{C.s_2, B.s_2\}$ for c_B, and one reconfiguration operation $c_A \xrightarrow{r_2} c_B$, with $r_2 = \{s^2_A, s^2_B\}$.

In the next section, we will present several notions of substitutability that will help us defining additional reconfiguration states and operations for our system.

5 Notions of substitutability

One of the key elements in allowing safe reconfiguration is to determine whether a certain component can be easily replaced by another one.

A relation of equivalence —such as bisimulation ($\sim$) in CCS— cannot be used for these purposes. Indeed, since there is mismatch among the interfaces of the components, a test based on bisimulation would immediately reject servers A and B as equivalent ($A \not\sim B$). Even if we accommodated name mismatch between both servers by using the adaptation contracts $\mathscr{AC}_{C,A}$ and $\mathscr{AC}_{C,B}$ for building name substitutions σ_A, σ_B according to the correspondence of names described in those contracts, the renamed components $A\sigma_A$ and $B\sigma_B$ would still remain not bisimilar, due to behavioural mismatch between them. Thus, we need to define a notion of substitutability adequate for our purposes, indicating whether the replacement of the server A by the server B (or *vice versa*) is suitable in a certain system willing to perform this reconfiguration.

5.1 Contextual equivalence

As we have seen, an equivalence relation like bisimulation is not well suited for our purposes since it takes into account all visible actions possibly performed by the components and ignores the context in which those components operate, and how this context affects them. A proof of equivalence would yield whether two components are interchangeable *in any system*, while we just need to prove if they can be exchanged in a *given* system.

Hence, we need to take into account the influence of the context and to ignore the actions performed by the former and novel components such that the rest of the system continues working transparently. This allows both former and novel components to have different behavioural interfaces as far as their adapted versions provide the same functionality from the point of view of the context. For representing how the context affects the behaviour of a component, we will use the adaptor generated for this component within this context, as described in section 2.

Definition 4. [Contextual interchangeability]. *Two components A and B are interchangeable in the context of another component C iff:*

$$(A||A_{C,A})/L_A \sim (A||A_{C,B})/L_B$$

where $A_{C,A}$, (resp. $A_{C,B}$) is the adaptor necessary for making A (resp. B) interact successfully with C, and L_A (resp. L_B) is the alphabet or set of labels used by A (resp. B) in its communications.

In the definition above, for checking contextual interchangeability we just have to compose the components A and B involved, together with the corresponding adaptors generated for interacting with the context C, and to hide the labels (L_A or L_B) through which the components and their adaptors communicate. The resulting processes represent the components as seen from the point of view of the context C. If they are equivalent —which can be easily checked with CCS tools like the Concurrency Workbench (CWB)—, they can be freely substituted one by another. Any action performed by one of them in the context of C can be exactly reproduced by the other one.

Example. In our running example, consider now a client $C2$ that buys exactly one book in each transaction:
$C2 = login!\ passwd!\ buybook!\ ack?\ logout!\ 0$
$C2$ can interact with server A or B indistinctly. Therefore, the client $C2$ (here playing the role of the context) enforces a behaviour that makes both servers A and B equivalent in the sense above. Hence, we should be able to build a system that is able to reconfigure at any point from the server A to the server B (or from the server B to the server A). Similarly, it is easy to find out that servers A and B are not equivalent in the context of the client C, as originally defined in Fig. 2(a).

5.2 Minimal Disruption

Contextual interchangeability requires that once adapted the components being considered are undistinguishable from the point of view of the context they interact with. A more relaxed notion of substitutability is what we call minimal disruption. Here, only the future actions performed by the environment are taken into account as far as the current system execution —but not *any* possible previous interaction— can be simulated in the new configuration. This is useful when the new configuration has an incompatible behaviour up to a certain point and a compatible one afterwards, but for some specific trace —the current execution— the incompatible part of the behaviour works fine.

Before defining minimal disruption, we have to show how can we enforce a certain component to execute a given trace. This is the purpose of the Definition 5 below.

Definition 5. [Trace-enforcing processes]. *Let $t = \{\langle a_0, \bar{a}_0\rangle, \ldots \langle a_n, \bar{a}_n\rangle\}$ be a trace of actions pairs, where each $\bar{a}_i$ states for the complementary action of a_i (i.e. if a_i is $a!$ then $\bar{a}_i$ is $a?$ and vice versa). Then, we define $Left(t)$ and $Right(t)$ as the processes:*
$Left(t) = a_0 \ldots a_n\ 0$

$Right(t) = \bar{a}_0 \ldots \bar{a}_n\ 0$
obtained by the sequential composition of the left (resp. right) actions from each pair $\langle a_i, \bar{a}_i \rangle$ in t.

Definition 6. [Minimal disruption]. *Two components A and B are minimal disrupting replaceable in the context of another component C, and given a trace of actions t, iff there exist A_t, B_t, C_t such that:*

- $A_t \sim Right(t)||(A||A_{C,A})/L_A$,
- $B_t \sim Right(t)||(B||A_{C,B})/L_B$,
- $C_t \sim Left(t)||C$, *and*
- A_t *and* B_t *are equivalent in the context of* C_t.

where as in Definition 4 $A_{C,A}$, (resp. $A_{C,B}$) is the adaptor necessary for making A (resp. B) interact successfully with C.

Hence, for finding out if two components A, B are interchangeable up to minimal disruption in a certain context C, and given a trace t already executed in that system, we just have to make A, B (composed with their corresponding adaptors), and C execute the corresponding part (left or right actions) of the trace, and then prove if the future behaviour of these components is equivalent from the point of view of the context. Again, all this can be easily checked with the CWB. In that case, we can freely perform the substitution of A by B (or the other way round) at the execution point defined by the trace.

Notice that the trace t used in this notion of minimal disruption shows us how to define a reconfiguration state for each configuration ($\langle C.s_i, A.s_j \rangle$ for c_A, and $\langle C.s_i, B.s_k \rangle$ for c_B) which denote the states in which A, B and C are after being enforced to reproduce the trace t, and the corresponding reconfiguration operations (from c_A to c_B and *vice versa*).

Example. In our running example, consider now that we are in configuration c_B —where the client C is interacting with the server B (adapted through $A_{C,B}$)— and they have already executed the trace $\{\langle C.login!, B.connect?\rangle, \langle C.passwd!, B.pwd?\rangle, \langle C.buybook!, B.buybook?\rangle\}$. If at that point we have to replace the server B with a fresh version of this server (let us call it B') due to server breakdown or connection failure, we have to initialise the new server B' (still adapted through $B_{C,B}$), with the process $connect!\ pwd!\ buybook!\ 0$. Then, the reconfigured system would be able to go on normally.

5.3 History-aware interchangeability

When dealing with component upgrade it is more useful to define a notion of substitutability that we could name as history-aware. Only the current execution needs to be simulated in the new configuration; future actions are allowed to be different. After reconfiguration, the environment may access the new services provided by the new component, or be denied to others that cannot be handled in the new configuration.

Definition 7. [History awareness]. *Two components A and B are history-awareness interchangeable in the context of another component C, and given a trace of actions t, iff there exists A_t, B_t, C_t such that:*

- $A_t \sim Right(t)||(A||A_{C,A})/L_A$,
- $B_t \sim Right(t)||(B||A_{C,B})/L_B$,
- $C_t \sim Left(t)||C$.

where all the processes involved in the definition above are the same as indicated in Definitions 5 and 6.

As we can see, history-aware interchangeability is a precondition for minimal disruption. However, we have preferred to present the notions in this order, from the finest grained to more relaxed notions.

Example. Consider in our running example that we initially are in configuration c_A, with the client logged to the server A (adapted through $A_{C,A}$) and that they have already executed the trace $\{\langle C.login!, A.user?\rangle, \langle C.passwd!, A.passwd?\rangle, \langle C.buybook!, A.buy?\rangle\}$. If at that point we have to move to configuration c_B, replacing the server A by the server B (for instance because the latter just became available and the client prefers it since it offers a wider functionality), we can check that both servers are history-aware interchangeable in the context of C and for the trace given. Thus, for performing the reconfiguration, we will have to initialise the new server B (adapted through $B_{C,B}$), with the process $connect!\ pwd!\ buybook!\ 0$ and then the reconfigured system would proceed without problems (possibly with the client taking advantage of the extended functionality provided by the new server).

Example. Consider now that we are in configuration c_B, and the trace already executed is $\{\langle C.login!, B.connect?\rangle, \langle C.passwd!, B.pwd?\rangle, \langle C.buymagazine!, B.buymagazine?\rangle\}$. If at that point the server B became unavailable, we could not move to configuration c_A since for the trace already executed both configurations do not fulfil the conditions of history awareness substitutability (in fact, they would not fulfil any of the notions of substitutability we have defined so far).

5.4 Advanced notions of substitutability

Apart from those presented in previous sections, more advanced notions of substitutability could be envisioned. For instance, we have identified that it would be useful to endow components with (possibly nested) transactions. Once a transaction is finished, there is no need to reproduce it if the component is substituted. This would lead to a notion of *transaction-aware substitutability*, whose utility is shown with the following example:

Example. In our client/server example, the servers would specify two nested transactions: one covers the full servers' protocol, from the login (either with *A.user* or *B.connect*) to the logout phase (in both cases with *disconnect*?). The other one, would be a sub-transaction, that starts when receiving a buy order, and ends when the acknowledgement has been sent to the client (e.g. from *B.buybook* to *B.bookok* for the server *B*). Then, it would be possible to start the system in configuration c_B, buy some magazines from the server *B* (which is not supported by the server *A*) executing the trace $\{\langle C.login!, B.connect?\rangle, \langle C.passwd!, B.pwd?\rangle, \langle C.buymagazine!, B.buymagazine?\rangle, \langle C.ack?, B.bookok!\rangle\}$ and then move to configuration c_A, substituting *B* by *A*. As the sale sub-transaction has finished, it can now be safely ignored when substituting the server *A*. Hence, the trace we would have to consider is just $\{\langle C.login!, B.connect?\rangle, \langle C.passwd!, B.pwd?\rangle\}$, corresponding to the unfinished full session transaction. Now we can find that the move from configuration c_B to c_A fulfils the conditions of history-awareness. Moreover, this would also prevent the client from buying again an already bought product.

6 Component model support

We plan to validate the ideas presented above through real-world applications on implementations using the Fractal component model [5].

Fractal is a modular, extensible, and programming language independent component model for designing, implementing, deploying, and reconfiguring systems and applications. We consider that it is a suitable setting for showing the benefits of our proposals because it deals explicitly with system reconfiguration, and has been the origin of many interesting formal underpinnings that can be applied to analysis of interface compatibility and verification of system properties [6, 3].

The Fractal model is an open component model, and in that sense it allows for arbitrary classes of controllers and interceptor objects, including user-defined ones. This allows us to define our own reconfiguration controllers that will take care of component discovery, adaptation, initialisation, and system reconfiguration. Moreover, in Fractal all remote invocations go through a membrane that controls the component. This makes the membrane an ideal container for a behavioural adaptor: the membrane will intercept all incoming and outgoing messages and pass them to the behavioural adaptor; the latter will compensate mismatch accordingly to the adaptation rules and orchestrate safe executions.

A good starting point for experimenting with our results is to use the framework developed in [4]. The framework is based on a Fractal-compliant component model and uses custom reconfiguration controllers in order to allow the system to self-adapt to changes in the environment. Their model supports dynamic reconfiguration, dynamic component instantiation, and support for interception of functional requests. Moreover, controllers are implemented in the form of a component-based system, which means that each of our controllers would be seen as a component plugged in the component's membrane.

7 Related work

Dynamic reconfiguration [14] is not a new topic and many solutions have already been proposed in the context of distributed systems and software architectures [9, 10], graph transformation [1, 21], software adaptation [17, 16], metamodelling [8, 14], or reconfiguration patterns [7]. On the other hand, Software Adaptation is a recent solution to build component-based systems accessed and reused through their public interfaces. Adaptation is known as the only way to compose black-box components with mismatching interfaces. However, only few works have focused so far on the reconfiguration of systems whose correct execution is ensured using adaptor components. In the rest of this section, we focus on approaches that tackled reconfiguration aspects for systems developed using adaptation techniques.

First of all, in [17], the authors present some issues raised while dynamically reconfiguring behavioural adaptors. In particular, they present an example in which a couple of reconfigurations is successively applied to an adaptor due to the upgrade of a component in which some actions have been first removed and next added. No solution is proposed in this work to automate or support the adaptor reconfiguration when some changes occur in the system.

Most of the current adaptation proposals may be considered as global, since they proceed by computing global adaptors for closed systems made up of a predefined and fixed set of components. However, notably an incremental approach at the behavioural level is presented in [18, 16]. In these papers, the authors present a solution to build step by step a system consisting of several components which need some adaptations. To do so, they propose some techniques to (i) generate an adaptor for each new component added to the system, and (i) reconfigure the system (components and adaptors) when a component is removed.

8 Conclusions

We have presented a new research track where components must be adapted to allow the system to be dynamically reconfigured. We have discussed some basic requirements for a runtime component substitution, and we have defined new interchangeability notions that allow to accommodate mismatch in behavioural interfaces. These notions are shown adequate for verifying compatibility of such components. For the same reason, we believe component discovery algorithms should also take into account components that have some degree of mismatch, as far as there is a specification of how mismatch can be worked out.

Finally, before reconfiguring the system, we have shown that the new component must be adapted and initialised accordingly to the current system state. These constitute new requirements for the runtime platform that we plan to address in the short-term.

The work presented here should be taken as an initial step towards dynamic reconfiguration where component candidates present both signature and behavioural mismatch. For the sake of simplicity, we have constrained ourselves to describe component protocols using LTS and CCS. However, one major drawback comes from this decision: data values present in message parameters are omitted. Since the protocols between components are often dependent on the data values carried in message parameters this limits the practical use of the proposal. An obvious extension of this work is to consider more expressive notations for describing behavioural interfaces, for instance Symbolic Transitions Systems (STS), or a value-passing process algebra. This will be part of our future work.

Acknowledgements

This work has been partially supported by the project TIN2008-05932 funded by the Spanish Ministry of Science and Innovation, and project P06-TIC-02250 funded by the Andalusian local Government.

References

[1] N. Aguirre and T. Maibaum. A Logical Basis for the Specification of Reconfigurable Component-Based Systems. In *Proc. of FASE'03*, volume 2621 of *LNCS*, pages 37–51. Springer, 2003.

[2] M. Autili, P. Inverardi, A. Navarra, and M. Tivoli. SYNTHESIS: A Tool for Automatically Assembling Correct and Distributed Component-based Systems. In *Proc. of ICSE'07*, pages 784–787. IEEE Computer Society, 2007.

[3] T. Barros, R. Ameur-Boulifa, A. Cansado, L. Henrio, and E. Madelaine. Behavioural models for distributed fractal components. *Annals of Telecommunications*, 64(1):25–43, 2009.

[4] F. Baude, D. Caromel, L. Henrio, and P. Naoumenko. *A Flexible Model And Implementation Of Component Controllers*. Coregrid. Springer, 2008.

[5] Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, and Jean-Bernard Stefani. The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. *Softw. Pract. Exper.*, 36(11-12):1257–1284, 2006.

[6] L. Bulej, T. Bures, T. Coupaye, M. Decky, P. Jezek, Pavel Parizek, F. Plasil, T. Poch, N. Rivierre, O. Sery, and P. Tuma. CoCoME in Fractal. In Andreas Rausch, Ralf Reussner, Raffaela Mirandola, and Frantisek Plasil, editors, *CoCoME*, volume 5153 of *Lecture Notes in Computer Science*, pages 357–387. Springer, 2008.

[7] TomÃąš Bureš, Petr Hnetynka, and František PlÃąšil. Sofa 2.0: Balancing advanced features in a hierarchical component model. In *SERA '06: Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications*, pages 40–48, Washington, DC, USA, 2006. IEEE Computer Society.

[8] Javier Cámara, Jose Antonio Martin, Gwen Salaün, Javier Cubo, Meriem Ouederni, Carlos Canal, and Ernesto Pimentel. Itaca: An integrated toolbox for the automatic composition and adaptation of web services. In *Proc. of ICSE'09*, pages 627–630. IEEE Computer Society, 2009.

[9] C. Canal, P. Poizat, and G. Salaün. Model-Based Adaptation of Behavioural Mismatching Components. *IEEE Transactions on Software Engineering*, 34(4):546–563, 2008.

[10] A. Cansado and C. Canal. On the reconfiguration of components in presence of mismatch. In *Proc. of the 2nd Workshop on Autonomic and SELF-adaptive Systems (WASELF09)*, pages 11–20. SISTEDES, 2009.

[11] A. Ketfi and N. Belkhatir. A Metamodel-Based Approach for the Dynamic Reconfiguration of Component-Based Software. In *Proc. of ICSR'04*, volume 3107 of *LNCS*, pages 264–273. Springer, 2004.

[12] J. Kramer and J. Magee. The Evolving Philosophers Problem: Dynamic Change Management. *IEEE Transactions on Software Engineering*, 16(11):1293–1306, 1990.

[13] J. Kramer and J. Magee. Analysing Dynamic Change in Distributed Software Architectures. *IEE Proceedings - Software*, 145(5):146–154, 1998.

[14] Jasminka Matevska-meyer, Wilhelm Hasselbring, and Ralf H. Reussner. Software architecture description supporting component deployment and system runtime reconfiguration. In *In Proc. 9th Int. Workshop on Component-oriented Programming*, 2004.

[15] N. Medvidovic. ADLs and Dynamic Architecture Changes. In *SIGSOFT 96 Workshop*, pages 24–27. ACM, 1996.

[16] P. Poizat and G. Salaün. Adaptation of Open Component-Based Systems. In *Proc. of FMOODS'07*, volume 4468, pages 141–156. Springer, 2007.

[17] P. Poizat, G. Salaün, and M. Tivoli. On Dynamic Reconfiguration of Behavioural Adaptation. In *Proc. of WCAT'06*, pages 61–69, 2006.

[18] P. Poizat, G. Salaün, and M. Tivoli. An Adaptation-based Approach to Incrementally Build Component Systems. In *Proc. of FACS'06*, volume 182, pages 39–55, 2007.

[19] M. Wermelinger, A. Lopes, and J. L. Fiadeiro. A Graph Based Architectural (Re)configuration Language. In *Proc. of ESEC / SIGSOFT FSE 2001*, pages 21–32. ACM, 2001.

Realizability and Dynamic Reconfiguration of Chor Specifications

Nima Roohi
Sharif University of Technology, Tehran, Iran
E-mail: roohi@ce.sharif.edu

Gwen Salaün
INRIA Grenoble - Rhône-Alpes / VASY France
E-mail: gwen.salaun@inria.fr

Keywords: choreography, realizability, dynamic reconfiguration, process calculus, labeled transition systems

Received: January 6, 2010

Choreography description languages aim at specifying from a global point of view interactions among a set of services involved in a new system. From this specification, local implementations or peers can be automatically generated. Generation of peers that precisely implement the choreography specification is not always possible: this problem is known as realizability. When peers corresponding to this specification are being executed we may want to modify the choreography specification and reconfigure dynamically the system. This is the case for instance if we add or remove interactions due to the addition of functionalities to the system at hand or the loss of a service. In this article, we present our solutions to check if a choreography is realizable and if a specific reconfiguration can be applied or not.

Povzetek: Opisana je metoda preverjanja možnosti implementacije sistema na osnovi opisa.

1 Introduction

A choreography describes how a set of services interact together from a global point of view. Several formalisms have already been proposed to specify choreographies: WS-CDL, collaboration diagrams, process calculi (such as Chor), BPMN, SRML, etc. Choreography specification, correctness, realizability and implementation are crucial issues in Service Oriented Computing. Several works aimed at studying and proposing solutions to the realizability problem [7, 18, 4, 2, 20] that consists in checking if a set of existing peers implements a choreography. In this article, we first present some techniques to check realizability of choreographies. Next, we focus on the dynamic reconfiguration of a choreography which has been distributed and deployed. Such reconfigurations correspond to the addition or removal of some interactions (loss of a service, extension of the functionalities, substitution of a service, etc.).

We use the Chor calculus [18] as choreography specification language, because it is an abstract model of WS-CDL coming with a formal syntax and semantics (not the case of WS-CDL). Our goal here is first to check the realizability of a choreography. To do so, we propose an encoding of Chor into the FSP process algebra and reuse equivalence checking tools to verify that the behaviors of both systems (centralized and distributed) are the same. Next, we formalize a reconfigurability test that checks if a set of peers that have been obtained from a choreography, can be reconfigured with respect to a second choreography specification which consists in an extension (addition of some interactions) or a simplification (removal of some interactions) of the original choreography. If these reconfigurations are possible, new peers are generated and replace the former ones. In addition, we also propose some analysis techniques to check some properties on the reconfiguration, *e.g.,* if modifications coming from the new choreography specification impact current peer behaviors only after their current execution state. Finally, if a choreography is realizable or can be reconfigured, we can automatically generate Java code for the corresponding peers for rapid prototyping purposes.

The rest of this article is organized as follows: Section 2 introduces Chor, Peer, and FSP, respectively as our choreography, peer, and intermediate languages. Section 3 presents some automatic techniques to first convert choreographies to an intermediate language, and then to check whether this choreography is realizable or not. In Section 4, we present our approach to check if some reconfigurations specified as a new choreography can be applied or not. We also present some techniques to analyze the impact of reconfigurations. In section 5 we describe our prototype tool, and comment on some experimental results. Also, we briefly overview code generation for peers. Section 6 compares our approach to related works, and Section 6 ends the article with some concluding remarks.

2 Preliminaries: Chor, Peer, and FSP

2.1 Chor and Peer

Chor [18] is a simple process language, and a simplified model of WS-CDL, for describing peers from a global point of view. From this global specification, behavioral specifications of peers can be generated by projection. In this section, we will overview both the Chor language (global view) and the Peer language (local view) introduced in [18].

Table 1 shows the syntax and semantics of Chor (C, C_1 and C_2 are arbitrary Chor specifications). It uses weak traces (τ actions are hidden) for specifying its semantics (where $[\![C]\!]$ stands for the weak trace set of C). The reader interested in more details on the language may refer to [18]. Also, operators on sets of traces which are used in Table 1 have been formally defined in [19].

The loop operator "$*$" has the highest priority among the others. After that, priority of the sequential composition operator ";" is higher than the other operators, as an example, $*C_1 \sqcap C_2; C_3$ is not ambiguous. Priority of parallel "||" and choice "$\sqcap$" operators is equal, as an example, $C_1 \| *C_2 \sqcap C_3 = (C_1 \| (*C_2)) \sqcap C_3$ (left associativity).

Chor is implemented by the coordination of a set of independent processes. The Peer language is a simple calculus for describing these processes. In this language, ε is an empty process which means do nothing, and for an arbitrary trace t if $P \stackrel{t}{\Longrightarrow} \varepsilon$ we have $t \in [\![P]\!]$ (we use $\dagger$ to denote deadlock). Table 2 gives the syntax and semantics of the Peer language (P, P_1 and P_2 are arbitrary Peer specifications).

The Peer language mainly differs from Chor by the description of interactions. Peer specifies them from a local point of view. Therefore, at the Peer level, an interaction activity is either an emission or a reception, and peers interact together by handshake communication (same channels, opposite directions).

Using rules defined in Table 2, trace sets of Peer processes are obtained as follows:

$$\frac{P \xrightarrow{\sigma} P'}{P \stackrel{\sigma}{\Longrightarrow} P'} \qquad \frac{P \xrightarrow{\sigma} P' \quad P' \stackrel{\sigma'}{\Longrightarrow} P''}{P \stackrel{\sigma^\frown\sigma'}{\Longrightarrow} P''}$$

Last, operator $/ : Peer \times Activity \to Peer$ returns the process obtained after executing the activity which is specified as the second input parameter of the "$/$" operator, and function fst (abbreviation for *first*) : $Peer \to \mathbb{P}(Activity)$, in which $\mathbb{P}(Activity)$ is the power set of all possible activities, computes activities of a Peer process which can be executed first. Formal definitions of operator "$/$" and function fst are as follows ($\perp$ denotes an undefined process):

$\mathsf{fst}(\alpha) \widehat{=} \{\alpha\}$

$\mathsf{fst}(\varepsilon) = \mathsf{fst}(skip) = \mathsf{fst}(P_1 \sqcap P_2) = \mathsf{fst}(*P) \widehat{=} \emptyset$

$\mathsf{fst}(P_1; P_2) \widehat{=} \mathsf{fst}(P_1)$ $\mathsf{fst}(P_1 \| P_2) \widehat{=} \mathsf{fst}(P_1) \cup \mathsf{fst}(P_2)$

$$skip/\alpha \widehat{=} \perp \qquad \alpha/\alpha' \widehat{=} \begin{cases} \varepsilon & \text{if} \quad \alpha = \alpha' \\ \perp & \text{if} \quad \alpha \neq \alpha' \end{cases}$$

$$(P_1; P_2)/\alpha \widehat{=} P_1/\alpha; P_2 \qquad (P_1 \sqcap P_2)/\alpha = (*P)/\alpha \widehat{=} \perp$$

$$(P_1 \| P_2)/\alpha \widehat{=} \begin{cases} P_1/\alpha \| P_2 & \text{if} \quad \alpha \in \mathsf{fst}(P_1) \\ P_1 \| P_2/\alpha & \text{if} \quad \alpha \in \mathsf{fst}(P_2) \\ \perp & \text{else} \end{cases}$$

Example. We will use throughout this article a metal stock market as running example. There are three peers in our example. First, peer Broker selects one of two metals, namely iron and steel, then look at the market as many times as needed until a sale on the selected metal becomes available. Broker sends his/her bid on the selected metal to the second peer (Market) of our example. After receiving a bid, Market performs the following two tasks concurrently: saving the bid in its own database, and checking to see if this bid is better than the best current one or not. Then, Market sends the result of this check and the name of the broker to the announcement Board (third peer of our example). If this bid is the best so far, Board will change the current winner and notifies the broker. Otherwise, Board does nothing (skip). In the Chor specification below, bk, mk, bd respectively stand for Broker, Market, and Board:

$\mathsf{Stock} =$
$(\mathsf{iron}^{\mathsf{bk}} \sqcap \mathsf{steel}^{\mathsf{bk}}); \mathsf{look}^{\mathsf{bk}}; *\mathsf{look}^{\mathsf{bk}}; \mathsf{bid}^{[\mathsf{bk},\mathsf{mk}]};$
$(\mathsf{save}^{\mathsf{mk}} || \mathsf{check}^{\mathsf{mk}}); \mathsf{result}^{[\mathsf{mk},\mathsf{bd}]};$
$(\mathsf{change}^{\mathsf{bd}}; \mathsf{notify}^{[\mathsf{bd},\mathsf{bk}]} \sqcap \mathsf{skip})$

2.2 FSP

FSP is a process calculus that takes inspiration in Milner's Calculus of Communicating Systems (1980) and in Hoare's Communicating Sequential Processes (1985), as explained by Magee and Kramer in [12]. FSP was originally designed for distributed software architecture specification, and distinguishes sequential and composite processes. Table 3 introduces FSP operators which are used in the rest of this article (x, y, *new*, and *old* are actions, P and Q are FSP processes).

3 Realizability of Chor specifications

3.1 Translating Chor into FSP

There are two main solutions in order to perform the realizability check automatically: (i) generate and compare sets of traces for Chor and Peer in an *ad-hoc* manner, or (ii) translate Chor and Peer to some intermediate language and use existing tools to compare their behaviours. We prefer the second solution because it enables the designers to take advantage of existing tools such as equivalence checking to verify realizability, or model-checking tools for validation and verification purposes. We chose FSP because it relies on a simple language yet expressive enough to encode Chor operators. Moreover, FSP is equipped with the LTSA toolbox which provides efficient tools for state space exploration and verification. This encoding allows

Table 1: Syntax & Semantics of Chor

$skip$	means do nothing, its trace set is equal to $\{\langle\rangle\}$
a^i	is an arbitrary local activity performed by peer i, and its trace set is $\{\langle a^i\rangle\}$
$c^{[i,j]}$	is a communication between two peers i (sender) and j (receiver) through channel c, its trace set is $\{\langle c^{[i,j]}\rangle\}$
$C_1;C_2$	means first C_1 and then C_2, $[\![C_1;C_2]\!] = [\![C_1]\!] \frown [\![C_2]\!]$
$C_1 \sqcap C_2$	means either C_1 or C_2, $[\![C_1 \sqcap C_2]\!] = [\![C_1]\!] \cup [\![C_2]\!]$
$C_1 \| C_2$	means C_1 and C_2 run concurrently, $[\![C_1 \| C_2]\!] = [\![C_1]\!] \bowtie [\![C_2]\!]$
$*C$	means execute C an arbitrary number of times, $[\![*C]\!] = [\![C]\!]^*$

Table 2: Syntax & Semantics of Peer

P	$::=$	BP	(basics)	BP	$::=$	$skip$	(no action)
	$\mid$	$P;P$	(sequential)		$\mid$	a	(local)
	$\mid$	$P \sqcap P$	(choice)		$\mid$	$c!$	(send)
	$\mid$	$P \| P$	(parallel)		$\mid$	$c?$	(receive)
	$\mid$	$*P$	(loop)				

Skip: $skip \xrightarrow{\langle\rangle} \varepsilon$ **Local:** $a \xrightarrow{\langle a\rangle} \varepsilon$

Sequential: $\dfrac{P_1 \xrightarrow{\sigma} P_1'}{P_1;P_2 \xrightarrow{\sigma} P_1';P_2}$ $\quad \varepsilon;P \xrightarrow{\langle\rangle} P$

Choice: $P_1 \sqcap P_2 \xrightarrow{\langle\rangle} P_1$ $\quad P_1 \sqcap P_2 \xrightarrow{\langle\rangle} P_2$

Parallel: $\varepsilon \| \varepsilon \xrightarrow{\langle\rangle} \varepsilon$

$$\frac{P_1 \xrightarrow{\sigma} P_1'}{P_1 \| P_2 \xrightarrow{\sigma} P_1' \| P_2} \qquad \frac{c! \in \mathsf{fst}(P_1) \quad c? \in \mathsf{fst}(P_2)}{P_1 \| P_2 \xrightarrow{\langle c\rangle} P_1/c! \| P_2/c?}$$

$$\frac{P_2 \xrightarrow{\sigma} P_2'}{P_1 \| P_2 \xrightarrow{\sigma} P_1 \| P_2'} \qquad \frac{c? \in \mathsf{fst}(P_1) \quad c! \in \mathsf{fst}(P_2)}{P_1 \| P_2 \xrightarrow{\langle c\rangle} P_1/c? \| P_2/c!}$$

Loop: $*P \xrightarrow{\langle\rangle} skip$ $\quad *P \xrightarrow{\langle\rangle} P;*P$

to: (i) validate and verify Chor specifications using the LTSA toolbox, (ii) generate peer protocols from its choreography specified in Chor, (iii) test for realizability of the Chor specification, and (iv) generate Java code from FSP for rapid prototyping purposes. One could decide to specify choreographies and peers directly using FSP. However, domain-specific languages such as Chor and Peer are more adequate to write such specifications, since they provide the exact level of expressiveness to do so.

Basic activities are translated into simple FSP processes with one transition from the source to the final state (we use τ for the *skip* action). The Chor sequential operator is encoded using the FSP sequential operator. As regards the choice operator, we prefix each operand by a τ transition, therefore similarly to the Chor language, selecting a choice operand is performed non-deterministically. In the FSP parallel operator, actions which are in alphabets of both operands can only evolve through synchronization, but the Chor parallel operator does not synchronize activities of its operands (interleaving). Consequently, we first prefix operands of each parallel operator with a unique value, thus no synchronization occurs. Then, we use the renaming operator of FSP to replace these new action names with their original values. The loop operator $*C$ is specified in FSP using a non-deterministic choice between performing *skip*, or performing C and then a recursive call to the FSP process that encodes the loop operator.

Definition 1 (Chor into FSP). *Encoding a Chor specification C into FSP is achieved using function* $\mathsf{c2f} : Chor \rightarrow FSPdescription$, *as presented in Figure 1 ("\" operator hides actions in the FSP process, "/" operator renames actions in the FSP process, and* $\mathsf{ac}(C)$ *returns non-skip basic activities of its Chor operand).*

FSP does not allow actions to have subscript or superscript. Therefore, we respectively translate a^i and $c^{[i,j]}$ into a_i and c_i_j. $\mathsf{c2f}_{pi}$ is a one-to-one function of type $Chor \rightarrow ProcessIdentifier$ generating fresh identifiers (the same ones for identical Chor specifications) as output, which obey naming rules[1] of FSP process identifiers. $T.\mathsf{c2f}_{pi}$ returns a process identifier which is obtained by prefixing the result of $\mathsf{c2f}_{pi}$ by T. For all C and C' such that $\mathsf{c2f}(C)$ has a process identifier $\mathsf{c2f}_{pi}(C')$ in its specification, the result of $\mathsf{c2f}(C')$ must be included in the result of $\mathsf{c2f}(C)$, because whenever we use one FSP identifier in our specification, we must include the specification of that process in our final specification. We proved that this translation preserves the semantics of the Chor language [19].

[1] These rules are defined in Section 2 of Appendix B in [12].

Table 3: FSP Operators and Informal Semantics

$(x\text{->}P)$	describes a process that initially executes action x and then behaves as P.
$(P;Q)$	describes a process that first behaves as P, and then (after completion of P) behaves as Q.
$(x\text{->}P\|y\text{->}Q)$	describes a process that either executes action x and then P, or action y and then Q.
$(P\|\|Q)$	represents the concurrent execution of P and Q. This operator synchronizes shared actions of P and Q.
$x:P$	prefixes each label in the alphabet of P with x.
$P/\{new_1/old_1, \ldots, new_n/old_n\}$	renames action labels. Each old label in P is replaced by the new one.
$P\backslash\{x_1,\ldots,x_n\}$	removes action names $x_1,\ldots,x_n$ from the alphabet of P and makes these actions "silent". These silent actions are labeled by τ. Silent actions in different processes are not shared.
$P@\{x_1,\ldots,x_n\}$	hides all actions in the alphabet of P which do not belong to the set $\{x_1,\ldots,x_n\}$.

Figure 1: Encoding Chor into FSP

$\mathsf{c2f}(skip)$	$\widehat{=}\ SKIP = (skip\text{->}END)\backslash\{skip\}.$
$\mathsf{c2f}(a^i)$	$\widehat{=}\ \mathsf{c2f}_{pi}(a^i) = (a\_i\text{->}END).$
$\mathsf{c2f}(c^{[i,j]})$	$\widehat{=}\ \mathsf{c2f}_{pi}(c^{[i,j]}) = (c\_i\_j\text{->}END).$
$\mathsf{c2f}(C_1;C_2)$	$\widehat{=}\ \mathsf{c2f}_{pi}(C_1;C_2) = \mathsf{c2f}_{pi}(C_1);SKIP;\mathsf{c2f}_{pi}(C_2);END.$
$\mathsf{c2f}(C_1 \sqcap C_2)$	$\widehat{=}\ \mathsf{c2f}_{pi}(C_1 \sqcap C_2) = (z\text{->}\mathsf{c2f}_{pi}(C_1);END\|z\text{->}\mathsf{c2f}_{pi}(C_2);END)\backslash\{z\}.$ assuming z is neither in the alphabet of $\mathsf{c2f}_{pi}(C_1)$ nor $\mathsf{c2f}_{pi}(C_2)$.
$\mathsf{c2f}(C_1\|\|C_2)$	$\widehat{=}\ \|T.\mathsf{c2f}_{pi}(C_1\|\|C_2)=(p1:\mathsf{c2f}_{pi}(C_1)\|p2:\mathsf{c2f}_{pi}(C_2)).$ $\mathsf{c2f}_{pi}(C_1\|\|C_2)=T.\mathsf{c2f}_{pi}(C_1\|\|C_2);SKIP;END/$ $\{ba_1/p1.ba_1\|ba_1\in\mathsf{ac}(C_1)\}\cup\{ba_2/p2.ba_2\|ba_2\in\mathsf{ac}(C_2)\}.$
$\mathsf{c2f}(*C)$	$\widehat{=}\ \mathsf{c2f}_{pi}(*C) = (z\text{->}SKIP;END\|z\text{->}\mathsf{c2f}_{pi}(C);SKIP;\mathsf{c2f}_{pi}(*C))\backslash\{z\}.$ assuming z is not in the alphabet of $\mathsf{c2f}_{pi}(C)$.

Example. Let us illustrate our encoding with some of the FSP processes generated for our example. In Table 4 we can see for instance how the choice operator is performed non-deterministically by prefixing the choice's operands by z and then hiding this action. Figure 2 shows the minimized LTS, obtained by compilation with LTSA, of the generated FSP code ($\mathsf{c2f}_{pi}(\mathsf{Stock})$). First, Broker decides what metal (s)he wants, iron or steel. Then, (s)he looks at the market as many times as needed until a sale on the selected metal becomes available (there is a loop on state 2 in the LTS). After that, (s)he sends his/her bid to the market. Next, Market saves the price and checks it, concurrently (there are two different paths from state 4 to state 6 in the LTS). Then, Market sends the result of the performed check to the board. Finally, Board either does nothing (if the result says the bid was not good enough), or changes itself and notifies the broker (if the result says the bid was the best one so far). This LTS was run several times using LTSA animation techniques, and the system behaved as expected. Model-checking was not required here because we chose a simple example in this article for the sake of comprehension.

3.2 Peer Generation

Given a Chor specification, one can generate the specification of each Peer using *natural projection*. Natural projection[2] of a Chor specification to Peer P first replaces each observable action with *skip* iff P does not perform that action. Chor and Peer share parallel, sequential, choice, and loop operators. For these operators the natural projection replaces each Chor operator by its equivalent in Peer, and applies recursively to their operands. Projection of basic activities from a Chor specification C to a Peer specification P is achieved as follows:

1. each activity not performed by P is replaced by *skip*,
2. a local activity performed by P remains unchanged,
3. a communication activity involving P is replaced by a channel input activity (if P is the receiver) or a channel output activity (if P is the sender).

Generation of FSP processes for an arbitrary Chor specification is performed using function c2f, defined previously in this section. The behavior of each Peer P in the choreography C is generated by hiding in the corresponding FSP ($\mathsf{c2f}_{pi}(C)$) all actions to which P does not participate (Definition 2).

Definition 2. *Given a Chor specification C and a Peer identifier p, the FSP process corresponding to* $\mathsf{nproj}(C,p)$*, the natural projection of the Chor specification C to the Peer p, is generated as follows ($\mathsf{p2f}_{pi}$ is defined similarly to $\mathsf{c2f}_{pi}$):*

[2]The reader may refer to [18] for the formal definition of natural projection.

Table 4: Some FSP Processes Generated for the Running Example

Chor Specification	FSP Process Specification
skip	SKIP = (skip->END)\{skip}.
$iron^{bk}$	Iron_bk = (iron_bk->END).
$look^{bk}$	Look_bk = (look_bk->END).
$bid^{[bk,mk]}$	Bid_bk_mk = (bid_bk_mk->END).
$iron^{bk} \sqcap steel^{bk}$	Ch = (z->Iron_bk;END|z->Steel_bk;END)\{z}.
$*look^{bk}$	L = (z->SKIP;END|z->Look_bk;SKIP;L)\{z}.
$(iron^{bk} \sqcap steel^{bk}); look^{bk}$	S = Ch;SKIP;Look_bk;END.
$save^{mk} \| check^{mk}$	||TP = (p1 : Check_mk||p2 : Save_mk).
	P = TP;SKIP;END/{check_mk/p1.check_mk, save_mk/p2.save_mk}.

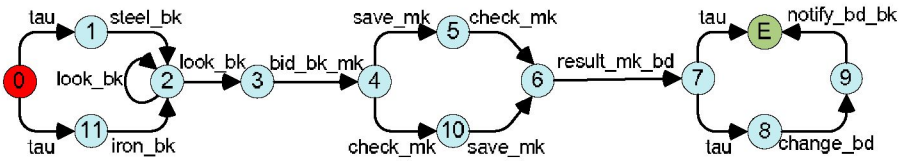


Figure 2: Minimized LTS of the *Stock* Market Case Study

$$p2f(C,p) \widehat{=} p2f_{pi}(C,p) = c2f_{pi}(C)@\{b | b \text{ is an activity of } p\}.$$

As specified in [18] for projecting Chor to peers, the name of each Peer process is taken as a part of each activity name (for instance, here we add it as suffix). Therefore, local activities of different peers are pair-wise different, and peers use exclusive channels for communicating with each other. Thus, each channel synchronizes activities of exactly two peers. Hence, in $p2f_{pi}(C,1) \| \cdots \| p2f_{pi}(C,n)$, only actions which represent communication activities are synchronized with each other, and each of these actions belongs to alphabets of exactly two FSP processes of the parallel operator's operands. We also proved that this translation preserves the semantics of the Peer language [19].

Example. For each Peer P, all actions in $c2f_{pi}(Stock)$ in which P is not involved, are hidden. The three peers of our example are encoded by the following FSP specifications:

Broker = $c2f_{pi}$(Stock);END@{iron_bk, steel_bk, bid_bk_mk, look_bk, notify_bd_bk}.
Market = $c2f_{pi}$(Stock);END@{save_mk, check_mk, bid_bk_mk, result_mk_bd}.
Board = $c2f_{pi}$(Stock);END@{result_mk_bd, notify_bd_bk, change_bd}.

Figure 3 shows the minimized LTSs of these peers generated from the FSP processes presented above.

3.3 Realizability

Definition 3 formalizes the notion of choreography realizability we use in this article. We chose a strong realizability [2, 7] for experimentation purposes, but weak notions could be used instead [7].

Definition 3 (Realizability of Chor). *For a Chor specification C with n peers, we say C is realizable under natural projection, if and only if the following two conditions hold:*

1. $[\![C]\!] = [\![nproj(C,1) \| \cdots \| nproj(C,n)]\!]$
2. $\nexists t \,.\, nproj(C,1) \| \cdots \| nproj(C,n) \stackrel{t}{\Longrightarrow} \dagger$

Both Chor and Peer languages use trace semantics. Therefore, for checking the realizability of a Chor specification we need to compare the trace set of a Chor specification with the trace set of the parallel composition of all peers. We proved in [19] that the trace set of a Chor specification is equal to the trace set of its FSP encoding, we also proved our encoding preserves the semantics of the Peer language. Thus, we have to check that FSP specifications for Chor and peers produce the same set of traces (in which τ actions are hidden) and terminate. Although the Chor specification is deadlock-free, the specification of the final system made of interacting peers (generated using natural projection) may cause deadlock. In addition to check that both specifications have the same set of traces, the parallel composition of the different peers has also to be deadlock-free. This check is easily computed using the LTSA toolbox. Also, one can perform any kind of test that is provided by LTSA, such as checking temporal properties between different activities in the Chor and Peer specifications.

Example. As for the realizability test, we first compute LTSs from FSP processes Stock and Peers, using LTSA. The FSP process for the whole system is: ||Peers = (Broker||Market||Board). Then, we compare trace sets of these processes using *ltscompare*, one of the tools belonging to the mCRL2 toolset[3] [6], and find out they produce

[3]LTSA does not allow to compute trace equivalence of two LTSs.

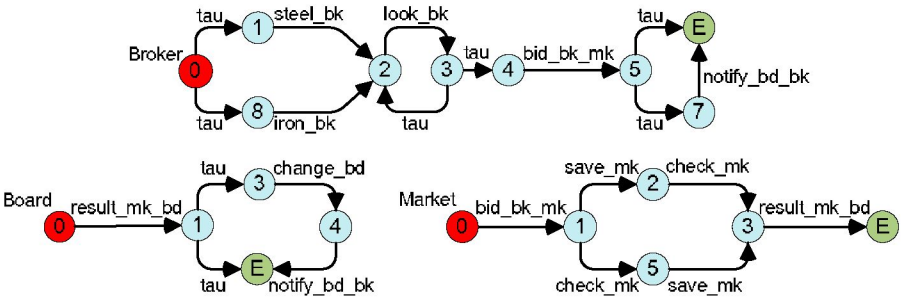


Figure 3: *Stock* Market: Minimized LTSs of Peers

the same set of traces (first realizability condition, Definition 3). For a Chor specification to be realizable, it is also required to satisfy the second condition of Definition 3. LTSA helps us on validating this condition, and using the *check safety* test, we find that the following trace causes deadlock:

$$\langle \mathsf{iron\_bk}, \mathsf{look\_bk}, \mathsf{bid\_bk\_mk}, \mathsf{check\_mk}, \mathsf{save\_mk}, \mathsf{result\_mk\_bd} \rangle$$

Indeed, after Broker sends his/her bid to the market, (s)he should decides if (s)he will be notified by the board or not. On the other hand, Board also makes this decision according to the result which is received from the market. So if peers Broker and Board make different decisions, a deadlock occurs. To make our specification realizable we slightly change it as follows: Whatever value is received from the market, Board always notifies the broker about the result. Thus, the specification of the system becomes as follows:

$\mathsf{Stock} =$
$(\mathsf{iron}^{\mathsf{bk}} \sqcap \mathsf{steel}^{\mathsf{bk}}); \mathsf{look}^{\mathsf{bk}}; *\mathsf{look}^{\mathsf{bk}}; \mathsf{bid}^{[\mathsf{bk},\mathsf{mk}]};$
$(\mathsf{save}^{\mathsf{mk}} || \mathsf{check}^{\mathsf{mk}}); \mathsf{result}^{[\mathsf{mk},\mathsf{bd}]};$
$(\mathsf{change}^{\mathsf{bd}} \sqcap \mathsf{skip}); \mathsf{notify}^{[\mathsf{bd},\mathsf{bk}]}$

This new specification satisfies both realizability conditions.

4 Dynamic reconfiguration of Chor specifications

4.1 Reconfigurability Definition

In this section, we show how we check whether a reconfiguration can be applied or not. Note that here our goal is not to verify the reconfiguration specification, it can be checked beforehand on the choreography specification using validation and verification techniques (see Section 3). Instead, we propose some techniques to check if, from a protocol point of view, a reconfiguration preserves the global flow of control executed so far.

This process accepts as input two choreographies (an initial one, say C_I, and a reconfigured one, say C_R) and a trace t which corresponds to the history of the current execution (sequence of local or communication activities, that interacting peers have performed). Traces only contain observable activities (τ corresponding to internal actions and used to encode non-deterministic choices in peers are not stored in these traces). From the choreography specification C_R, peer LTSs are obtained using techniques presented in Section 3. If the trace t executed by peers obtained out of C_I can also be executed in reconfigured peers generated from C_R, then the reconfiguration can take place.

Definition 4 (Reconfigurability). *Given two choreographies C_I and C_R, two sets of peers P_I and P_R respectively obtained from those choreographies, and a trace t, the current system consisting of peers P_I is reconfigurable to peers P_R if there exists P'_R such that $P_R \stackrel{t}{\Longrightarrow} P'_R$, where $\stackrel{t}{\Longrightarrow}$ stands for the execution of local or communication activities as specified in trace t.*

In practice, a reconfiguration is applied as follows: First, actual peers matching with abstract descriptions (LTSs) derived from the choreography C_R are seeked into databases of peers (*e.g.,* UDDI) or directly reused from the former system for peers which have not been modified. Next, these peers are instantiated and executed (using the history stored in trace t) up to the point where the reconfiguration has been applied (this last part can be enforced by an external controller or a monitoring engine for instance). To sum up, our reconfigurability check aims to be transparent from an external point of view.

Example. Now imagine that after peer Broker selects iron ($\mathsf{t} = \langle \mathsf{iron}^{\mathsf{bk}} \rangle$), we want to reconfigure the current choreography for Stock market, in a way that i) in addition to iron and steel, Broker can select gold, and ii) Broker can send his/her bid to the Market without looking at the market. The new specification of the system is as follows:

$\mathsf{Stock} =$
$(\mathsf{iron}^{\mathsf{bk}} \sqcap \mathsf{steel}^{\mathsf{bk}} \sqcap \mathsf{gold}^{\mathsf{bk}}); *\mathsf{look}^{\mathsf{bk}}; \mathsf{bid}^{[\mathsf{bk},\mathsf{mk}]};$
$(\mathsf{save}^{\mathsf{mk}} || \mathsf{check}^{\mathsf{mk}}); \mathsf{result}^{[\mathsf{mk},\mathsf{bd}]};$
$(\mathsf{change}^{\mathsf{bd}} \sqcap \mathsf{skip}); \mathsf{notify}^{[\mathsf{bd},\mathsf{bk}]}$

Figure 4 shows the minimized LTS of the new peer (Broker). We first compute the parallel composition of peers P_R using LTSA, then we check in this system if it is possible to perform activities which are specified in t. The

Therefore, we first save them in a format *ltscompare* accepts, and then use it to check if LTSs have the same set of traces or not.

answer is yes, and P_I is reconfigurable to P_R. This is automatically checked using a prototype tool we implemented (see Section 5).

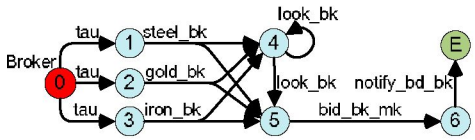


Figure 4: Minimized LTS of new *Broker*

4.2 Reconfigurability Analysis

Our reconfigurability definition, only says if the activities that have occurred so far can be reproduced in the new system, then peers P_I are reconfigurable to peers P_R. We want to go further than this check since in some situations, one may want these reconfigurations to have an immediate impact on the running system, or to preserve the forthcoming behaviour as specified in the former choreography (the system can do at least what was possible before, but it can do more as well).

Therefore, Definition 4 is completed with a couple of analysis of where the modifications take place, that is we check if modifications appear in peers after the current global state, and if the evolutions possible from the current global state are preserved with respect to the former choreography. These analyses may help the designer to decide whether (s)he wants to reconfigure the system or not. Indeed, we can imagine situations in which for instance the designer may want these modifications to immediately impact the whole behaviour.

The first case, referred as *preservative* in the following, is computed by first performing reconfigurability check and finding P'_R. Then if P'_R is found, it is checked that all traces which can be executed from P'_I (assuming $P_I \stackrel{t}{\Longrightarrow} P'_I$) can also be executed from P'_R.

Definition 5 (Preservative Reconfiguration). *Given two choreographies C_I and C_R, two sets of peers P_I and P_R respectively obtained from previous choreographies, and a trace t, new peers P_R are preservative with respect to former peers P_I, if P_I is reconfigurable to P_R and $[\![P'_I]\!] \subseteq [\![P'_R]\!]$, assuming $P_I \stackrel{t}{\Longrightarrow} P'_I$ and $P_R \stackrel{t}{\Longrightarrow} P'_R$.*

Note that P'_I and P'_R obtained by application of trace t are unique, because τ transitions have been removed from peers and they have been determinized (no two transitions holding the same label going out from the same state) after performing the realizability check presented in Section 3.2.

The second case, referred as *modificative* in the following, is computed by first extracting the current global state from the trace t, and checking for each reconfigured peer if all new interactions are reachable from its current execution state.

Definition 6 (Modificative Reconfiguration). *Given two choreographies C_I and C_R, two sets of peers P_I and P_R respectively obtained from previous choreographies, and a trace t, new peers P_R are modificative with respect to former peers P_I if in addition to be reconfigurable, for each peer $p_i \in P'_R$, $s_i \in (s_1,\ldots,s_n)$, we have* $\mathsf{reachable}(s_i,p_i) \cap M_i = M_i$, *where $P_R \stackrel{t}{\Longrightarrow} P'_R$, $(s_1,\ldots,s_n)$ is the current global state of peers P'_R, and M_i stands for all the modifications (added or removed interactions) applied between C_I and C_R for peer i.*

Given a peer i, modifications for this peer between choreographies C_I and C_R are obtained by computing the difference of both alphabets $A_{Ii} \backslash A_{Ri}$ ($A_{Ri} \backslash A_{Ii}$, resp.) if some interactions are removed (added, resp.). Function reachable from a state s and a peer LTS p is defined as follows:

$\forall s,p \,.\, \mathsf{reachable}(s,p) = \emptyset \Leftrightarrow \nexists s', l \,.\, (s,l,s') \in T$

$\forall s,p,l \,.\, l \in \mathsf{reachable}(s,p) \Leftrightarrow \exists s' \,.\, (s,l,s') \in T \vee$

$\quad (\exists l' \,.\, (s,l',s') \in T \wedge l \in \mathsf{reachable}(s',p))$

where T is the transition relation belonging to the peer LTS $p = (A,S,I,F,T)$.

Last, realizability of choreography C_R can be checked using techniques presented in Section 3, and this realizability result is another analysis on which the user can rely on to decide whether or not applying the reconfiguration.

Example. Suppose that in addition to be reconfigurable, we want our system to verify both properties. Since peers P_R are reconfigurable with respect to former peers P_I, we know P'_R exists such that $P_R \stackrel{t}{\Longrightarrow} P'_R$. Therefore, assuming $P_I \stackrel{t}{\Longrightarrow} P'_I$, for reconfiguration to be preservative we need $[\![P'_I]\!] \subseteq [\![P'_R]\!]$. This check can be performed using the *ltscompare* tool, and by performing that check we find that our reconfiguration example is preservative.

As regards the modificative reconfiguration property, $\mathsf{M_{bk}} = \{\mathsf{gold_{bk}}\}$, $\mathsf{M_{mk}} = \mathsf{M_{bd}} = \emptyset$. After selecting iron, there is no way to perform $gold_{bk}$. Consequently, our reconfiguration is not modificative. We have to wait for the current execution to get finished first, and then reconfigure the peers if we want this property to be satisfied.

5 Prototype tool

All the steps of the approach we have presented in Sections 3 and 4 are automatically computed by a prototype tool we implemented (see an overview in Figure 5). Boxes and diamonds with dashed borders are optional. We explicitly wrote names of tools that we did not implement at the bottom of each box or diamond. In Section 3 we have mentioned that one reason to choose an intermediate language, is that we can reuse tools which have already been created for that language. If we assume each box or diamond as a unit of work, one can see that using our approach, we only implemented 41.4 percent of our prototype tool, and 58.6 remaining percent are already implemented in existing tools.

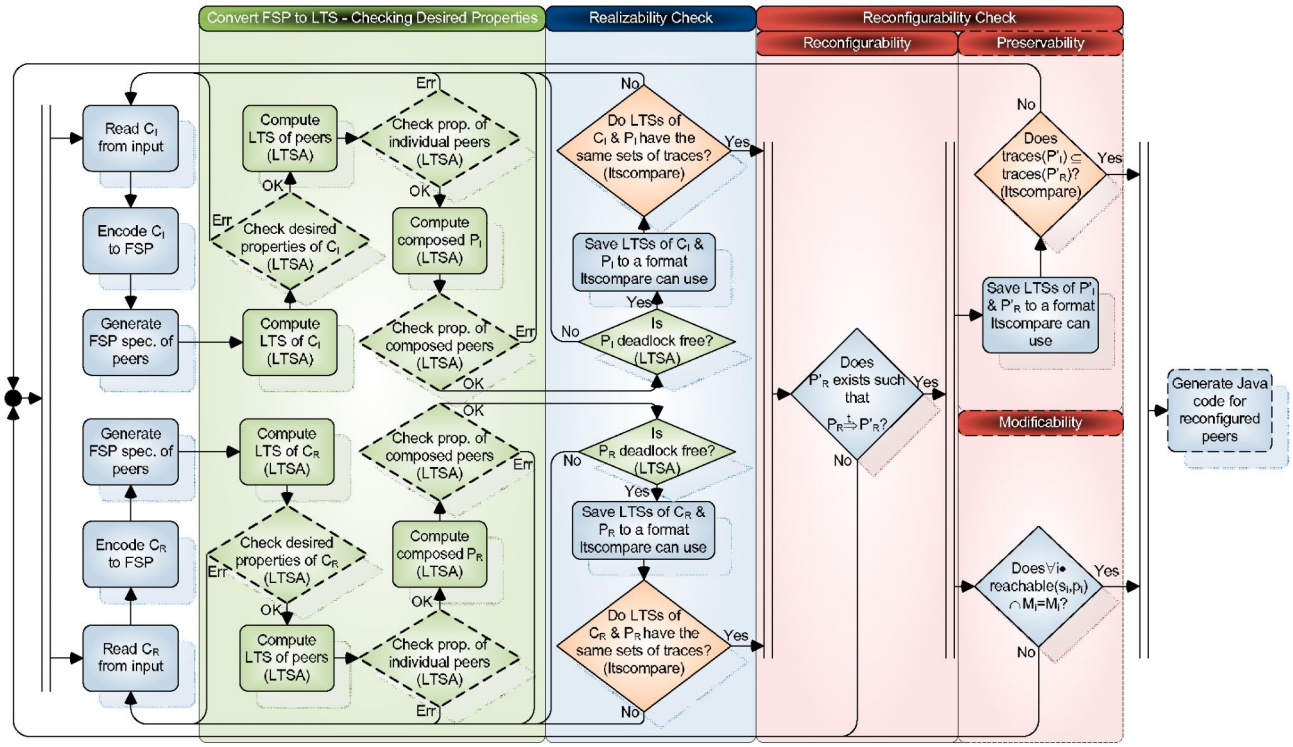


Figure 5: Overview of our prototype tool

5.1 Experimental Results

Table 5 shows experimental results on some of the examples of our database. Each row of this table shows results for one reconfiguration request (C_I, C_R, and t), and respectively presents the number of: peers, distinct basic activities used in the Chor specification, basic activities used in the Chor specification, basic activities in t (length of t), FSP processes resulting while encoding the Chor specification into FSP, and states and transitions in the *minimized* LTS corresponding to the parallel composition of peers (P_I and P_R). It also presents result of realizability plus different types of reconfigurability checks and amount of consumed time and memory. For keeping the table as simple as possible, we chose examples in which number of peers in C_I and C_R are equal. Also number of (distinct) basic activities, and FSP processes for C_I and C_R are close to each other (the maximum is shown).

Note that measured time and memory for the reconfigurability check include time and memory required for the realizability check. Also, time and memory for the two other types of reconfigurability check include time and memory required for the basic reconfigurability check in addition to the realizability check.

Whenever a reconfigurability check fails, there is no need to check the preservability or modificability properties, since being reconfigurable is a precondition for being preservative or modificative.

5.2 Code Generation

As mentioned earlier, the final step is to produce Java code following guidelines presented in [12]. Like the other steps, this is completely automated by our tool. Figure 6 shows a simplified version of some classes produced for our running example. We define an **interface** Channel and implement it in a **class** ChannelImpl. For each channel in the specification, one instance of ChannelImpl is created in **class** ChannelServer and registered in a server. Also, for each peer we create one interface and one class. The interface contains methods for local and communication activities performed by the peer and must be implemented by the user, because the semantics of basic activities used in the specification is not defined. Code in the class file implements the peer protocol and should not be changed. The user only needs to implement interfaces of peers and distributes classes to different locations, as (s)he needs.

Let us comment in more details, for illustration purposes, method run in **class** mkController. We can notice that for each operand of the parallel operator we created one separate thread, and used **class** CyclicBarrier (the Java utility class) to guarantee that the execution of both threads must be finished before the next activities are performed (cb1.wait() and cb2.wait()). Also, SynchronousQueue used in **class** ChannelImpl is another Java class which synchronizes its read/write operations, therefore our communication mechanism remains synchronous.

Table 5: Experimental Results

P	dBA	BA	t	FSP	States		Trans.		Realizability			Reconf.			Preservability			Modificability		
3	5	5	0	10	3	5	2	6	√	96ms	796K	√	174ms	754K	√	187ms	771K	√	173ms	870K
3	8	9	1	18	12	19	16	28	√	140ms	815K	√	224ms	824K	√	248ms	886K	√	243ms	949K
5	16	17	4	29	31	37	52	66	√	174ms	354K	√	392ms	900K	√	404ms	916K	√	378ms	1024K
5	16	36	6	30	65	194	108	449	×	146ms	978K	×	146ms	978K	×	1s	978K	√	146ms	978K
4	5	8	8	15	67	47	489	255	√	833ms	1187K	√	1.1s	2052K	×	1.1s	2055K	√	1.1s	2056K
5	14	672	9	29	757	755	1428	1416	√	3.2s	2965K	√	12.1s	3116K	×	80s	3130K	×	12s	3014K
6	6	6	0	16	95	141	340	602	√	4.7s	2501K	√	5.2s	5212K	√	5.5s	5516K	√	5.2s	5559K
7	13	13	8	23	250	374	725	1277	√	1.3s	4170K	√	1.7s	6375K	√	1.7s	6942K	√	1.7s	6454K
7	17	834	11	35	932	934	1372	1372	√	8.5s	2739K	√	42s	2732K	√	244s	2947K	×	42s	2982K

```
public class mkController extends Thread {
  private final mk mk;
  private final Channel bk;
  private final Channel bd;
  public mkController(mk mk, String server) throws
    RemoteException,NamingException{
    this.mk = mk;
    final Context namingContext = new InitialContext();
    bk = (Channel) namingContext
      .lookup("rmi://"+server+"/bk_mk");
    bd = (Channel) namingContext
      .lookup("rmi://"+server+"/mk_bd");}
  public void run() {
    final Serializable msg1 = bk.recv();
    mk.recv_from_bk(msg1);
    final CyclicBarrier cb1 = new CyclicBarrier(2);
    new Thread(new Runnable() {
      public void run() {
        mk.check();
        cb1.wait();}}).start();
    new Thread(new Runnable() {
      public void run() {
        mk.save();
        cb1.wait();}}).start();
    final Serializable msg2 = mk.send_bd_value();
    bd.send(msg2);}}
```

```
public interface mk {
  void save();
  void check();
  void recv_from_bk(Serializable value);
  Serializable send_bd_value();}
public class ChannelImpl implements Channel {
  public ChannelImpl() throws RemoteException {
    UnicastRemoteObject.exportObject(this, 0);}
  private final SynchronousQueue syncQueue = new
    SynchronousQueue();
  public void send(Serializable value) throws
    RemoteException,InterruptedException{
    syncQueue.put(value);}
  public Serializable recv() throws
    RemoteException,InterruptedException{
    return syncQueue.take();}}
public class ChannelServer {
  public ChannelServer() throws
    RemoteException,NamingException{
    final Channel bk_mk = new ChannelImpl();
    final Channel mk_bd = new ChannelImpl();
    final Channel bd_bk = new ChannelImpl();
    final Context namingContext = new InitialContext();
    namingContext.bind("rmi:bk_mk", bk_mk);
    namingContext.bind("rmi:mk_bd", mk_bd);
    namingContext.bind("rmi:bd_bk", bd_bk);}}
```

Figure 6: *Stock* Market: Java Code

6 Related works

Several works aimed at studying and defining the conformance and/or realizability problem for choreography. In [3], the authors define models for choreography and orchestration, and formalise a conformance relation between both models. These models are assumed given as input whereas we focus on the generation of one from the other (generation of peers from a global specification). In [22], the authors focus on *Let's dance* models for choreographies, and define for them an algorithm that determines if a global model is locally enforceable, and another algorithm for generating local models from global ones. In [15], the authors show through a simple example how BPEL stubs can be derived from WS-CDL choreographies. However, due to the lack of semantics of both languages, correctness of the generation cannot be ensured.

Some works define several realizability notions, and classify them in a hierarchy [7]. Bultan and Fu [2] tackle the realizability issue in the context of asynchronous communication, and recently defined some sufficient conditions to test realizability of choreographies specified with collaboration diagrams. In [18, 11], formal languages to describe choreographies were proposed. Conformance with respect to an orchestration specification and implementability issues were studied from a formal point of view.

Other works [4, 18] propose well-formedness rules to enforce the specification to be realizable. For example, in [4], the authors rely on a π-calculus-like language and session types to formally describe choreographies. Then, they identify three principles for global description under which they define a sound and complete end-point projection, that is the generation of distributed processes from the choreography.

Dynamic reconfiguration [14] is not a new topic and many solutions have already been proposed in the context of distributed systems and software architectures [9, 10], graph transformation [1, 21], software adaptation [17, 16], or metamodelling [8, 13]. However, to the best of our knowledge, nobody has already worked on the reconfiguration of service interactions initially described using a

choreography specification.

As regards tools automating the realizability test, WSAT [5] takes conversation protocols as input, and checks a set of realizability conditions on them. Another tool-supported approach [20] computes realizability using a LOTOS encoding. However, in [20] the choreography language, namely collaboration diagrams, is less expressive than Chor (no choice and a loop operator restricted to a single message), and the proposal focuses only on abstract languages (no relationships with implementations or real code).

7 Concluding remarks

In this article, we have presented an encoding of the choreography calculus Chor into the process algebra FSP. This encoding allows to generate a set of peers corresponding to the choreography, and in a second step to check that (i) they realize the original choreography, and (ii) they ensure some expected properties (by animation and model-checking with LTSA). If the choreography is not realizable or erroneous, the Chor specification can be corrected and the process started again. If a choreography is as expected by the designer, Java code can be generated for rapid prototyping purposes. We have also proposed some techniques to verify if some reconfigurations can be applied dynamically on some peers that have been generated from a choreography specification. For illustration purposes, we have used the Chor language and transition systems to describe peers. Reconfigurations have been specified as a new version of the choreography where some interactions have been added or removed. Our approach is completely automated by a prototype tool we implemented and applied to a large number of examples.

Our main perspective plans to extend our approach to consider asynchronous communication. In this article, we have focused on synchronous communication, and it makes the realizability and reconfigurability checking easier. Dealing with asynchronous communication is a realistic assumption with respect to implementation platforms, however it complicates the analysis and verification stage. Asynchronous communication can be specified using queues. In this context, realizability and reconfigurability results depend on queue size, and some theoretical issues are still open problems such as the relationships of realizability results for queues of size one, queues of size k, and infinite queues. We also plan to extend our analysis techniques to take other kinds of reconfigurations into account. As an example, in some situations one may wish to reduce the behaviour of the interacting peers while producing only traces that were executable before reconfiguring the system (this is the opposite of the preservative property presented in Section 4).

References

[1] N. Aguirre and T. Maibaum. A Logical Basis for the Specification of Reconfigurable Component-Based Systems. In *Proc. of FASE'03*, volume 2621 of *LNCS*, pages 37–51. Springer, 2003.

[2] T. Bultan and X. Fu. Specification of Realizable Service Conversations using Collaboration Diagrams. *Service Oriented Computing and Applications*, 2(1):27–39, 2008.

[3] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Choreography and Orchestration Conformance for System Design. In *Proc. of Coordination'06*, volume 4038 of *LNCS*, pages 63–81. Springer, 2006.

[4] M. Carbone, K. Honda, and N. Yoshida. Structured Communication-Centred Programming for Web Services. In *Proc. of ESOP'07*, volume 4421 of *LNCS*, pages 2–17. Springer, 2007.

[5] X. Fu, T. Bultan, and J. Su. WSAT: A Tool for Formal Analysis of Web Services. In *Proc. of CAV'04*, volume 3114 of *LNCS*, pages 510–514. Springer, 2004.

[6] J. F. Groote, A. Mathijssen, M. Reniers, Y. Usenko, and M. van Weerdenburg. The Formal Specification Language mCRL2. In *Proc. of MMOSS'07, Dagstuhl seminar*, 2007.

[7] R. Kazhamiakin and M. Pistore. Analysis of Realizability Conditions for Web Service Choreographies. In *Proc. of FORTE'06*, volume 4229 of *LNCS*, pages 61–76. Springer, 2006.

[8] A. Ketfi and N. Belkhatir. A Metamodel-Based Approach for the Dynamic Reconfiguration of Component-Based Software. In *Proc. of ICSR'04*, volume 3107 of *LNCS*, pages 264–273. Springer, 2004.

[9] J. Kramer and J. Magee. The Evolving Philosophers Problem: Dynamic Change Management. *IEEE Transactions on Software Engineering*, 16(11):1293–1306, 1990.

[10] J. Kramer and J. Magee. Analysing Dynamic Change in Distributed Software Architectures. *IEE Proceedings - Software*, 145(5):146–154, 1998.

[11] J. Li, H. Zhu, and G. Pu. Conformance Validation between Choreography and Orchestration. In *Proc. of TASE'07*, pages 473–482. IEEE Computer Society, 2007.

[12] J. Magee and J. Kramer. *Concurrency: State Models & Java Programs, 2nd edition*. Wiley, 2006.

[13] J. Matevska-Meyer, W. Hasselbring, and R. Reussner. Software Architecture Description Supporting Component Deployment and System Runtime Reconfiguration. In *Proc. of WCOP'04*, 2004.

[14] N. Medvidovic. ADLs and Dynamic Architecture Changes. In *SIGSOFT 96 Workshop*, pages 24–27. ACM, 1996.

[15] J. Mendling and M. Hafner. From Inter-organizational Workflows to Process Execution: Generating BPEL from WS-CDL. In *Proc. of OTM'05 Workshops*, volume 3762 of *LNCS*, pages 506–515. Springer, 2005.

[16] P. Poizat and G. Salaün. Adaptation of Open Component-based Systems. In *Proc. of FMOODS'07*, volume 4468 of *LNCS*, pages 141–156. Springer, 2007.

[17] P. Poizat, G. Salaün, and M. Tivoli. On Dynamic Reconfiguration of Software Adaptations. In *Proc. of WCAT'06*, 2006.

[18] Z. Qiu, X. Zhao, C. Cai, and H. Yang. Towards the Theoretical Foundation of Choreography. In *Proc. of WWW'07*, pages 973–982. ACM Press, 2007.

[19] N. Roohi, G. Salaün, and S. H. Mirian. Analyzing Chor Specifications by Translation into FSP. In *Proc. of FOCLASA'09*, volume 255 of *ENTCS*, pages 159–176, 2009.

[20] G. Salaün and T. Bultan. Realizability of Choreographies using Process Algebra Encodings. In *Proc. of IFM'2009*, volume 5423 of *LNCS*, pages 167–182. Springer, 2009.

[21] M. Wermelinger, A. Lopes, and J. L. Fiadeiro. A Graph Based Architectural (Re)configuration Language. In *Proc. of ESEC / SIGSOFT FSE 2001*, pages 21–32. ACM, 2001.

[22] J. Maria Zaha, M. Dumas, A. H. M. ter Hofstede, A. P. Barros, and G. Decker. Service Interaction Modeling: Bridging Global and Local Views. In *Proc. of EDOC'06*, pages 45–55. IEEE Computer Society, 2006.

Model-Based Dependable Composition of Self-Adaptive Systems

Javier Cubo, Carlos Canal and Ernesto Pimentel
Dept. of Computer Science, University of Málaga, Campus de Teatinos, 29071, Málaga, Spain
Email: {cubo,canal,ernesto}@lcc.uma.es, Web: http://www.lcc.uma.es/~{cubo,canal,ernesto}

Keywords: self-adaptive, context-aware, SOA, transition systems, model transformation, dependable composition, adaptation, evolution, fault tolerance, error recovery

Received: October 15, 2010

Building mobile and pervasive systems as a selection, composition, adaptation and evolution of pre-existing software entities may arise dynamically and continuously different issues related to inconsistencies, changes or faults. We propose an approach to detect and handle these issues with the appropriate methodology in every case. This is performed by tackling three great challenges in software engineering related to self-adaptive systems: (i) their formalisation, by using model-based SOA, which bridge the business and software processes, (ii) their development and maintenance, by performing adaptation and/or evolution when inconsistencies or changes occur, and (iii) their monitoring to handle faults, by using error recovery techniques. We use an example based on an intelligent transportation system to validate our proposal.

Povzetek: Opisana je sestava prilagodljivih sistemov na osnovi modela.

1 Introduction

The increased usage of mobile and portable devices has given rise over the last few years to a new market of mobile and pervasive applications. These applications may be executed on either mobile computers (laptops, tabletPCs, etc.), or wireless hand-held devices (PDAs, smart phones, etc.), or embedded systems (PDAs, on-board computer, intelligent transportation or buildings systems, etc.), or even use sensors or RFID tags. Their main goal is to provide connectivity and services at any time, adapting and monitoring when required and improving the user experience. These systems are different to the traditional distributed computing systems. On one hand, a mobile system is able to change location allowing the communication via mobile devices. On the other hand, a pervasive application attempts to create an ambient intelligence environment to make the computing part of it and its enabling technologies essentially transparent. This results in some new issues related to inconsistencies, changes or faults, which arise dynamically and continuously whilst composing services in these systems, and which have to be detected and handled. These issues can be classified into four main categories [23]: (i) mismatch problems, (ii) requirement and configuration changes (iii) network and remote system failures, and (iv) internal service errors. The first refers to the problems that may appear at different interoperability levels (*i.e.*, signature, behavioural or protocol, quality of service and semantic or conceptual levels), and the Software Adaptation paradigm tackles these problems in a non-intrusive way [8].The second is prompted by continuous changes over time (new requirements or services fully created at run-time), and Software Evolution (or Software Maintenance) focuses on solving them in an intrusive-way [19]. The third and fourth are related to networks (network connection break-off or remote host unavailable) and services (suspension of services during the execution or system run-time error) failures respectively, that are both addressed by Fault Tolerance (or Error Recovery) mechanisms [24]. Developing real-world mobile and pervasive systems handling all these faults is extremely complex and error-prone. Therefore, it is essential to determine an effective methodology to develop this kind of system.

Self-adaptive software provides a broadly inclusive adaptation methodology that spans a wide range of adaptive behaviours [22]. One of the key aspects of self-adaptive software is that it supports both software adaptation and evolution, by addressing mismatch problems and requirement or configuration changes. Another advantage of self-adaptive systems is that they adapt the software systems to changing operational contexts and environments, thereby reducing human effort in the human-computer interaction. Context-awareness provides the most relevant information (location, identity, time and activity) to users and stakeholders, adapting themselves to their changing situation, preferences and requirements, and optimising the quality of service [12]. Therefore, context information plays an important role in software adaptation and evolution to control the scope of change. However, current programming technology offers only very weak support for developing context-aware applications, and new research is urgently needed to develop novel Context-Oriented Programming (COP) mechanisms [21]. As regards failures related to networks and services, fault tolerance mechanisms to be exploited for the development of dependable systems allow the handling of exceptions raised by adaptive demand, returning back the self-adaptive system to any earlier stable state. The choice of fault tolerance mechanisms depends on the fault assumptions and on the system's char-

acteristics and requirements. There are two main classes of error recovery [24]:backward and forward error recovery. The former is based on rolling services back to the previous correct state in the presence of failure. The latter involves transforming the system services into any correct state, and relies on an exception handling mechanism.

Self-adaptive systems requires high reusability, dependability, robustness, adaptability, and availability. In order to reduce efforts and costs, these systems may be developed using existing *Commercial-Off-The-Shelf* (COTS) components or (Web) services. In contrast to the traditional approach in which software systems are implemented from scratch, COTS and services can be developed by different vendors using different languages and different computer platforms. Although the reuse of software has matured and has overcome some of the previously mentioned problems, it has not become standard practice yet, since reusing components or services requires the selection, composition, adaptation and evolution of prefabricated software parts, by means of their public interfaces, in order to solve different problems. Thus, it is desired to reduce the effort of adapting and maintaining existing services in order to achieve a cost-effective and dependable development of self-adaptive systems. Component-Based Software Engineering (CBSE) [25] and Service-Oriented Architecture (SOA) [13] promote software reuse by selecting and assembling pre-existing software entities (COTS and services, respectively)[1]. These software development paradigms allow the building of fully working systems as efficiently as possible in order to improve the level of self-adaptive software reusability, dependability and adaptability [8]. On one hand, current industrial platforms using CBSE provide only some of the means to describe components at their signature level (*e.g.*, CORBA's IDL[2]). On the other hand, one way to implement SOA is using WSDL for describing services, SOAP[3] for communication, UDDI for service registry and discovery, and BPEL [1] for service orchestration. However, BPEL is not yet widely considered in current XML-based industrial service technology, which, in addition, only supports queries based on keywords and categories. This may bring about erroneous executions and/or low-precision results in realistic and complex applications, as it neither handles the order in which the service messages are exchanged with its environment, nor is it able to discover semantic capabilities of services (functionality) nor can it be adapted to a changing environment without human intervention. Therefore, behavioural descriptions, and multiple context and semantic (*e.g.*, by means of ontologies) information must be specified and managed in real-world services to avoid undesirable situations during their interaction, such as deadlocks or livelocks, and to improve their features (such as QoS). In this sense, our proposal tackles the need to support the variability of the adaptation process in self-adaptive systems by using context-aware, semantic-based, model-based adaptation, and dependency analysis mechanisms.

Approach and Contributions. We propose an approach to detect and handle the different inconsistencies, changes or faults arisen in self-adaptive systems. This is performed by tackling three great challenges in software engineering related to self-adaptive systems: (i) their formalisation, by using model-based SOA, which bridges the business and software processes, (ii) their development and maintenance, by performing adaptation and/or evolution when required, and (iii) their monitoring to handle faults, by using error recovery techniques.

In order to achieve these goals, we make the following contributions. Firstly (i) we develop a model transformation process to allow us the discovery, composition, adaptation and maintenance of services. This process, according to the Model-Driven Architecture (MDA)[4], takes a source model (BPEL or WF [10], both implemented as SOAs) and produces a target model (in our case transition systems), and vice versa. Secondly, (ii) we use software adaptation and evolution concern respectively with adapting or changing the software during its execution. Both paradigms typically tackle the adapting and the evolving of software separately depending on the changes being made. However, we propose a model-based approach using self-adaptive techniques through both paradigms to reduce the effort and cost of modifying the system. In this way, our approach will assist respectively to the application developers and software designers to first apply software adaptation (non-intrusive way) when that paradigm may solve the problem, and only in the case it is not enough, software evolution (intrusive way) will be used. Finally, (iii) we combine both backward and forward error recovery techniques to maintain consistency, correctness, and coordination of changes, and to handle errors in self-adaptive systems. We have developed a prototype tool on Python, which implements our approach, integrated inside the toolbox ITACA [7]. `ITACA`[5] (Integrated Toolbox for the Automatic Composition and Adaptation of Web Services) is a toolbox under implementation at the University of Málaga for the automatic composition and adaptation of services accessed through their interfaces. The toolbox fully covers an adaptation process which goes from behavioural model extraction from existing service interface descriptions, to the final adaptor implementation.

Figure 1 is an overview of our approach, which focuses on systems made up of a service repository, clients (considered as services as well), and a shared domain ontology. When a user performs a request, *e.g.*, from a mobile device, our process is executed. First, (1) abstract interface specifications (Context-Aware Symbolic Transition Systems, CA-STSs, presented in Section 3.1) are extracted from the BPEL or WF services, by means of our model transformation process (Section 3.2). Then, (2) a discov-

[1] In the sequel, we use service to refer both terms.

[2] www.omg.org/technology/documents/formal/components.htm

[3] http://www.w3.org/TR/soap/

[4] http://www.omg.org/docs/omg/03-06-01.pdf

[5] Accesible at http://itaca.gisum.uma.es

ery process based on semantic and compatibility mechanisms finds the services satisfying that request, and identifies possible mismatches and changes that will determine whether the services involved need adaptation and/or evolution (Section 4.1). If mismatches or changes occur, then (3) observation planning will determine *when*, *where*, *what*, and *how* [6] to perform adaptation and/or evolution depending on whether the changes are related to anticipated or unanticipated adaptation, respectively. Next, (4.a) in the case that adaptation is required, a CA-STS adaptor will be generated in a non-intrusive way (Section 4.1), and (4.b) if evolution is needed, then first the designer will have to modify the system in an intrusive way, and second the adaptor will be generated (Section 4.1). Then, (5) from the CA-STS adaptor, the corresponding BPEL or WF adaptor service is generated using our model transformation process (Section 4.1), and the whole system is deployed, allowing the BPEL or WF services to interact via the BPEL or WF adaptor. Finally, (6) a fault tolerance process handles exceptions raised by adaptive demand, returning back the system to any earlier stable situation, by using error recovery techniques (Section 4.2).

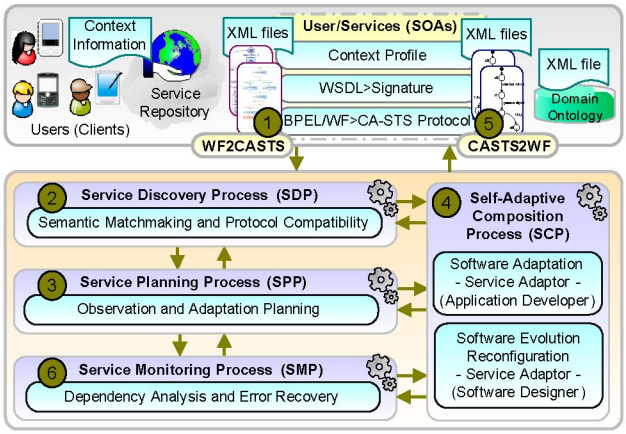


Figure 1: Overview of our proposal

Outline. The remainder of this article is structured as follows. In Section 2, we introduce a case study which will be used throughout this article for illustrative purposes. Section 3 presents our model-based SOA approach. In Section 4.1 the dependable composition process is described. Sections 5 presents works related to model-based transformation, and self-adaptive and error recovery techniques. Finally, Section 6 ends the article with a discussion about the evaluation of our approach and some concluding remarks.

2 Motivating example: ITS

To illustrate our proposal, we describe a case study in which services connected to an Intelligent Transportation System (ITS) require and provide context-aware transportation facilities. We consider different scenarios where city users of transport (passengers or drivers) are interested in planning their route on their hand-held devices (mobile phones or onboard computer), by receiving data from a Route service. In addition to the Route service, a Map service may also reply the user's requests as the Route service is not available anymore. We consider two kind of users: 1) bus/metro passenger, and 2) drivers. The latter have two different profiles: driving a private vehicle (car), or a taxi. Users receive different results of their route, depending on their profiles. We assume services can respond to the users' requests, but issues related to inconsistencies, changes or faults may arise at run-time, making it necessary to detect and handle them.

Passenger scenario. A passenger communicates with the system to obtain the best itinerary to a destination by bus/metro. The Route service response depends on certain context information, *i.e.*, the passenger location and destination, as well as the traffic or transport timetable, so the result may vary frequently.

Driver scenario. Let us imagine the Route service typically calculates a route requested by drivers based on traffic congestion (considering vehicles that enter and leave an area). In a normal situation, a car/taxi driver can change the route dynamically on being advised by the ITS of rerouting alternatives. In this scenario, we describe three different cases:

- A) Drivers request that the Route service considers the context information related to the weather as a new requirement to calculate the rerouting.

- B) From the previous case, vehicles driving in a specific area discover a new Car Parking service provided by the context of the new environment, but not considered initially by the system. Drivers would like to request this new service, so the ITS should include that service into the system.

- C) Considering the requirements of the two previous cases, we imagine that in a certain moment the connection with the Route service is lost. This service will be replaced automatically and quickly at run-time by another service with similar functions and considered at design-time by the system, *i.e.*, the Map service, which will also help to guide the driver.

This case study presents a service-oriented pervasive system with context-awareness features. Self-adaptive software, in addition to tackle different adaptive behaviours, is useful for dealing with all forms of embedded or pervasive software. We use a structured modelling approach to specify service-oriented architectures, because it is easier to determine when a new service is needed, as well as when it is more cost-effective and efficient to alter an existing service, develop a new one, or acquire a third-party service, and to manage fault tolerance mechanisms. Since models tend to be represented using a graphical notation, the model-based methodology involves using visual-modeling languages. We adopt an expressive and user-friendly graphical notation based on transition systems, which reduces the complexity of modelling services, as we will show in the next section.

3 Model-based SOA

In this section, we describe our formal model to specify services using Context-Aware Symbolic Transition Systems (CA-STS). Different automata-based or Petri net-based models can be used to describe behavioural interfaces. We have chosen CA-STS, which is based on transition systems, because it is simple, graphical, and provides a good level of abstraction to tackle discovery, verification, composition, or adaptation issues [14, 15]. Furthermore, any formalism to describe dynamic behaviour may be expressed in terms of a transition system [14]. Thus, our approach becomes general enough to be applied in other applications. In addition, we relate our interface model to implementation platforms. There exists several platforms or languages that can be used to develop services, such as UML[6], BPEL or WF. First, we present the syntax and operational semantics of our interface model. Second, we describe a textual grammar to abstract implementation details of WF activities, and define our transformation process to extract CA-STS specifications from WF services.

3.1 CA-STS Interface Model

We consider systems consisting of context-aware clients and services. We assume both client and service interfaces are specified using context profiles, signatures and protocols. Context profiles define information which may change according to the client preferences and service environment. Signatures correspond to operations profiles. Protocols are represented using transition systems. Client and services interact according to the operational semantics we will define later.

Context Profile, Signature and Protocol.
A *context* is defined as *"the information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to interaction between a user and an application including the user and application themselves"* [12]. Context information can be represented in different ways and can be classified in four main categories [17]: (i) user context: role, preferences, language, calendar, social situation or privileges, (ii) device/computing context: network connectivity, device capabilities or server load, (iii) time context: current time, day, year, month or season, and (iv) physical context: location, weather or temperature. For our purpose, we only need a simple representation where contexts in both clients and services are defined by *context attributes* with associated values. In addition, we differentiate between static context attributes (*e.g.*, role, day, ...) and dynamic ones (*e.g.*, network connectivity, current time, location, ...). Dynamic attributes can change continuously at run-time, so they have to be dynamically evaluated during the service composition. Finally, both clients and services are characterised by public (*e.g.*, weather, temperature, season, ...) and private (*e.g.*, personal data, local resources, ...) context attributes. Thus, we represent and gather the service context information by using a *context profile*, which is a set of tuples (CA, CV, CK, CT), where: CA is a context attribute (or simply context) with its corresponding value CV, CK determines if CA is static or dynamic, and CT indicates if CA is public or private. For instance, (*user, driver, static, public*), indicates that *user* is a public and static context which corresponds to the user profile *driver* as value.

A *signature* corresponds to a set of operation profiles. This set is a disjoint union of provided and required operations. An operation profile is the name of an operation, together with its argument types (input/output parameters) and its return type.

A *protocol* is represented using a Symbolic Transition Graph (STG) [16] extended with value passing, context variables and conditions, that we call Context-Aware Symbolic Transition System (CA-STS). Conditions specify how applications should react (*e.g.*, to context changes). We take advantage of using ontologies described in a specific domain to capture and manage the semantic information of the services in a system by comparing concepts, such as context information, operation names, arguments and types. In this way, we can determine the relationship between the different concepts that belong to that domain.

Let us introduce the notion of variable, expression, and label required by our CA-STS protocol. We consider two kinds of *variables*, those representing regular variables or static context attributes, and those corresponding to dynamic context attributes (named context variables). In order to distinguish between them, we will mark the context variables with the symbol "$\sim$" over the specific variable. An *expression* is defined as a variable or a term constructed with a function symbol f (an identifier) applied to a sequence of expressions, $i \in f(F_1, \ldots, F_n)$, F_i being expressions.

Definition 1 (CA-STS label). *A label corresponding to a transition of a CA-STS is either an internal action τ (tau) or a tuple (B, M, D, F) representing an event, where: B is a condition (boolean expression that manages both conditional choices and context changes), M is the operation name, D is the direction of operations (! and ? represent emission and reception, respectively), and F is a list of expressions if the operation corresponds to an emission, or a list of variables if the operation is a reception.*

Definition 2 (CA-STS Protocol). *A Context-Aware Symbolic Transition System (CA-STS) Protocol is a tuple (A, S, I, Fc, T), where: A is an alphabet which corresponds to the set of CA-STS labels associated to transitions, S is a set of states, $I \in S$ is the initial state, $Fc \subseteq S$ are correct final states (deadlock-free), and $T \subseteq S \times A \times S$ is the transition function whose elements $(s_1, a, s_2) \in T$ are usually denoted by $s_1 \xrightarrow{a} s_2$.*

Finally, a *CA-STS interface* is constituted by a tuple (CP, SI, P), where: CP is a context profile, and SI is the signature of the CA-STS protocol P. Both client and services consist of a set of interfaces. For instance, let us

[6] http://www.omg.org/technology/documents/formal/uml.htm

focus on the client shown in Figure 6. It has an interface (CP_U, SI_U, P_U), where CP_U refers to the context information related to the user location (dynamic context attribute *loc*), user profile and device used by the client of the user (static context attributes *user* and *dev* respectively), SI_U is formed by all the operation profiles, such as $l_{u_1} = reqR!dest, \tilde{loc}, us\tilde{er}$, and P_U is the protocol which indicates the CA-STS behaviour. For example, l_{u_1} means that a client with the context information *loc* and *user* issues an emission looking for a route from his/her location to a destination, and then this client receives a possible route $l_{u_2} = getR?route$, and so on. Note we have left out the return types of the arguments to simplify the notation. Initial and final states are depicted in CA-STSs using bullet arrows and hollow states, respectively. Our proposal is suitable for synchronous systems where clients interact with services, such as mobile systems. We adopt a synchronous and binary communication model (explained in next section, Figure 3). Clients can execute several protocols simultaneously, *i.e.*, concurrent interactions (in a binary model). Client and service protocols can be instantiated several times.

At the user level, client and service interfaces can be specified by using: (i) context information into XML files for context profiles, (ii) WSDL for signatures, and (iii) business processes defined in industrial platforms, such as Abstract BPEL (ABPEL) [1] or WF workflows (AWF) [10], for protocols. We assume context information is inferred from the client requests (HTTP header of SOAP messages), thereby as a change occurs the new value of the context attribute is automatically sent to the corresponding service (controlled in rules presented in Figure 2). We also consider processes (clients and services) implemented as business processes which provide the WSDL and protocol descriptions.

Next, we define the CA-STS operational semantics.

Operational Semantics of CA-STS.

We formalise first the operational semantics for one CA-STS service, and second for the composition of n CA-STS services. In the following, we use a pair $\langle s, E\rangle$ to represent an active state $s \in S$ and an environment E. An environment is a set of pairs $\langle x, v\rangle$ where x is a variable, and v is the corresponding value of x (it can be also represented by $E(x)$). The function *type* returns the type of a variable. We use boolean expressions b to describe CA-STS conditions. Regular and context variables are evaluated in emissions and receptions (by considering the current value of the context, *e.g.*, the current date), respectively. Therefore, two evaluation functions are used to compute expressions in an environment: (i) ev evaluates regular variables or expressions, and (ii) ev_c evaluates context variables changing dynamically. We define ev as follows:

$$ev(E,x) \triangleq \begin{cases} E(x) & \text{if } x \text{ is a regular variable} \\ x & \text{if } x \text{ is a context variable} \end{cases}$$

$$ev(E, f(v_1,\ldots,v_n)) \triangleq f(ev(E,v_1),\ldots,ev(E,v_n))$$

Function ev_c is defined in a similar way to ev, only considering context variables, since we first apply ev to evaluate the regular variables: $ev_c(E,x) \triangleq E(x)$, where x is a context variable. We also define an environment overloading operation "$\oslash$" in such a way that given an environment E, $E \oslash \langle x, v\rangle$ denotes a new environment, where the value corresponding to x is v.

We present in Figure 2 the semantics of a CA-STS ($\rightarrow_o$), with three rules that formalise the meaning of each kind of CA-STS label: internal actions τ (INT), emissions (EM), and receptions (REC); and one rule to simulate the dynamic update of the environment according to the context changes at run-time (DYN). Note that according to Definition 1, $b \in B$ is a condition, $a \in M$ is an operation name, and $x \in F$ and $v \in F$ correspond to a list of variables and expressions, respectively. A condition b may contain regular and/or context variables and both of them must be evaluated in the environment of the source service (sender), because the decision is taken in the sender. However, evaluation of expressions v only affects regular variables (rule EM), since context variables will be evaluated in the target service (receiver) to consider the context values when the message is received (see rule COM in Figure 3). We assume that the dynamic modification of the environment will be determined by different external elements depending on the type of the context (*e.g.*, user intervention, location update by means of a GPS, time or temperature update, and so on). Then, we model this situation by assuming a transition relation which indicates the environment update as a change occurs, denoted by $E \rightsquigarrow_d E'$, where $E'(x) \neq E(x)$ only if x is a dynamic context variable, and in which case the new value of x is automatically sent to the corresponding service.

$$\frac{(s \xrightarrow{b,\tau} s') \in T \quad ev_c(ev(E,b),b) = \texttt{true}}{\langle s,E\rangle \xrightarrow{\tau}_o \langle s',E\rangle} \quad \text{(INT)}$$

$$\frac{(s \xrightarrow{b,a?x} s') \in T \quad ev_c(ev(E,b),b) = \texttt{true}}{\langle s,E\rangle \xrightarrow{a?x}_o \langle s',E\rangle} \quad \text{(REC)}$$

$$\frac{(s \xrightarrow{b,a!v} s') \in T \quad ev_c(ev(E,b),b) = \texttt{true} \quad v' = ev(E,v)}{\langle s,E\rangle \xrightarrow{a!v'}_o \langle s',E\rangle} \quad \text{(EM)}$$

$$\frac{E \rightsquigarrow_d E'}{\langle s,E\rangle \xrightarrow{\tau}_o \langle s,E'\rangle} \quad \text{(DYN)}$$

Figure 2: Operational Semantics of one CA-STS

The operational semantics of n ($n > 1$) CA-STSs ($\rightarrow_c$) is formalised using two rules: a first synchronous communication rule (COM, Figure 3) in which value-passing and variable substitutions rely on a late binding semantics [20] and where the environment E is updated; and a second independent evolution rule (INE$_\tau$, Figure 3). A list of pairs $\langle s_i, E_i\rangle$ is represented by $[as_1,\ldots,as_n]$. Rule COM uses the function ev_c to evaluate dynamically in the receiver the context changes related to the dynamic context attributes of the sender. Regular variables have been evaluated previously in the rule EM when the message is emitted. This dynamic evaluation handled in the operational semantics allows the modelling of service protocols depending on con-

text changes. Rule INE_τ is executed in the case of an internal service propagation that gives rise to either a state (related to the rule INT) or an environment (rule DYN) change. Thus, transitions $\rightarrow_c$ do not distinguish between internal evolutions coming from either internal actions in services or dynamic updates in the environment.

$$\frac{i,j \in \{1..n} \quad i \neq j \quad type(x) = type(v) \quad \langle s_i, E_i\rangle \xrightarrow{a!v}_o \langle s_i', E_i\rangle \quad \langle s_j, E_j\rangle \xrightarrow{a?x}_o \langle s_j', E_j\rangle \quad E_j' = E_j \oslash \langle x, ev_c(E_j, v)\rangle}{[as_1, \ldots, \langle s_i, E_i\rangle, \ldots, \langle s_j, E_j\rangle, \ldots, as_n] \xrightarrow{a!v}_c [as_1, \ldots, \langle s_i', E_i\rangle, \ldots, \langle s_j', E_j'\rangle, \ldots, as_n]} \quad \text{(COM)}$$

$$\frac{i \in \{1..n} \quad \langle s_i, E_i\rangle \xrightarrow{\tau}_o \langle s_i', E_i'\rangle}{[as_1, \ldots, \langle s_i, E_i\rangle, \ldots, as_n] \xrightarrow{\tau}_c [as_1, \ldots, \langle s_i', E_i'\rangle, \ldots, as_n]} \quad (\text{INE}_\tau)$$

Figure 3: Operational Semantics of n CA-STSs

Following, we present our model transformation process by using WF services as illustration purpose.

3.2 Model Transformation Process

To perform the service discovery, composition, adaptation and maintenance, we first need to define a textual notation to abstract and formalise services implemented in the WF platform. Second, we define our model transformation process.

Abstraction of WF Workflows.
To relate our model transformation process with realistic and complex examples, we use the WF platform, which belongs to the .NET Framework 3.5 and is supported by Visual Studio 2008. We have chosen WF because it makes the implementation of services easier thanks to its workflow-based graphical support and the automation of the code generation, and it is an useful and interesting alternative compared to the well-know BPEL. Nevertheless, we have also validated our proposal using BPEL as shown in [7]. In addition, the .NET Framework is widely used in many companies, and WF is increasingly prevalent in the software engineering community [26].

In order to illustrate the motivating example presented in Section 2, we use a representative kernel of the WF activities, namely `Code`, `Sequence`, `Terminate`, `Receive`, `Send`, `IfElse`, `While`, and `Listen` with `EventDriven` activities, that are general enough to describe any service.

In Table 1, we formalise the textual grammar (left hand-side) defined for the WF activities considered (on the right hand-side the informal meaning of these activities is provided), which abstracts several implementation details. Our grammar considers as input textual workflows (defined in XML files) corresponding to the graphical description of the WF workflows, with WF activities $\mathscr{A}$, where C, C_i are boolean conditions, I, I_i (inputs), O, O_i (outputs) are parameters of activities, and Id are service identifiers.

The WF platform is capable of developing workflows in different scenarios, from simple sequential ones to realistic and complex state machine-based workflows involving human interaction. The programming languages available in the platform are *Visual Basic* and *C#*. Our examples have been implemented in *C#*.

$\mathscr{A}$ ::= `Code`	*executes code*
| `Terminate`	*ends WF*
| `Receive(`$Id,Op[,O,I_1,\ldots,I_n]$`)`	*receives msg*
| `Send(`$Id,Op[,O_1,\ldots,O_n,I]$`)`	*sends msg*
| `Sequence(`$\mathscr{A}_1,\mathscr{A}_2$`)`	*executes* $\mathscr{A}_1,\mathscr{A}_2$
| `IfElse(`$(C_1,\mathscr{A}_1),\ldots,(C_n,\mathscr{A}_n),\mathscr{A}$`)`	$\mathscr{A}_i$ *if* C_i *or* $\mathscr{A}$
| `While(`$C,\mathscr{A}$`)`	$\mathscr{A}$ *while* C
| `Listen(`$E_1,\ldots,E_n$`)`	*fires one* E_i
E ::= `EventDriven(Receive(`$Id,Op[,I_i]$`),`$\mathscr{A}$`)`	$\mathscr{A}$ *when* Id

Table 1: Grammar for the WF abstract notation

Example. We have designed WF workflows for the User Route request, and for the Route and Map services. WF provides a WSDL description for each WF workflow. For space reasons, in Figure 4 only the WF workflow that represents the behaviour of the User Route request is shown.

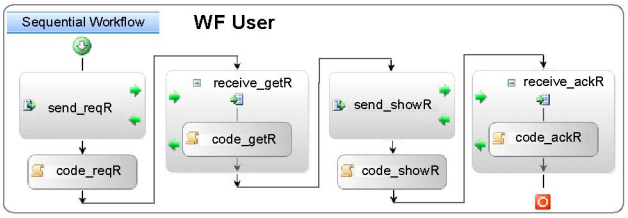


Figure 4: WF workflow of the User's request

Next, we present how we extract CA-STS specifications from WF services.

From WF to CA-STSs.
CA-STSs are used as an abstraction to focus on behavioural composition issues by describing service interfaces in a standard notation. These CA-STSs are automatically generated from WF services. For each WF service, our model transformation process parses the three XML files corresponding to its context information, WSDL description, and WF workflow. A new XML file containing the information about its context profile, signature, and CA-STS protocol is automatically generated. This XML corresponds to the behavioural interface of a CA-STS specification. This process has been implemented following the patterns of our transformation process presented in Figure 5.

We have developed an ad-hoc transformation language to translate WF activities (WF workflows defined in XML files) in CA-STS elements (XML files represented in a graphical notation by means of transition systems) and vice versa. The extracted CA-STS specifications must preserve the semantics of workflows as encoded in the WF platform. A formal proof of semantics preservation between both levels has not been achieved yet since the WF formal semantics is not rigorously documented. Our encoding has been deduced from our experiments using the WF platform. The main ideas of the CA-STS specification obtained from abstract description of workflow constructs are the following: (i) `Code` is an internal transition, (ii) `Terminate` corresponds to a final state, (iii) `Receive` and `Send` are reception and emission, respectively, (iv) `Sequence` must

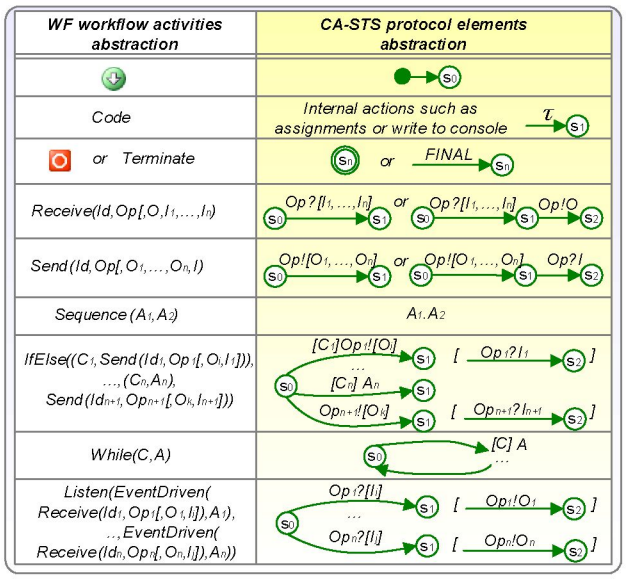


Figure 5: Patterns of our model transformation process from WF to CA-STS and vice versa

preserve the activities' order, (v) `IfElse` corresponds to an internal choice, (vi) `While` is translated as a looping behaviour, and `Listen` corresponds to an external choice. Initial and final states in the CA-STS come respectively from the initial and final states that appear in the workflow. There is a single initial state that corresponds to the beginning of the workflow. Final states correspond either to a `Terminate` or to the end of the workflow, so several final states may appear in the CA-STS because several branches in the workflow may lead to a final state.

Example. We apply the model transformation process to the WF services of our case study in order to obtain the corresponding CA-STS specifications. Figure 6 shows the interfaces of the User (passenger or driver) and the Route and Map services modelled using our CA-STS interface model. Each interface has a context profile, a signature and a CA-STS protocol.

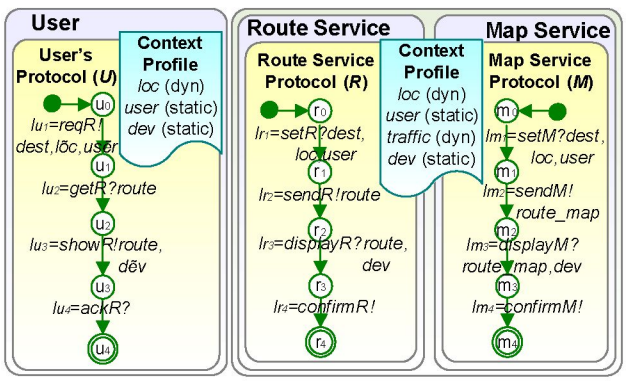


Figure 6: CA-STS of User and Route and Map services

4 Dependable composition of self-adaptive SOA

This section presents our approach to tackle self-adaptive systems changing dynamically over time and must continue offering services as inconsistencies, changes or faults occur. We aim to combine self-adaptive composition and error recovery techniques to perform adaptation and evolution strategies and to handle errors, respectively.

4.1 Self-Adaptive Composition

Composing services relates to dealing with assembly of autonomous services given their interfaces. We need to address the specification of the composition, and to ensure the services are composed in a consistent way.

Firstly, our discovery process (SDP module in Figure 1) finds the most appropriate services for a user's request. To do that, it is based on semantic matchmaking and protocol compatibility techniques [9]. The first is used to establish a ranked list of the services that better match the user's request, by comparing the semantic matching of the context profiles and all the operation profiles (names, arguments and types) *w.r.t.* an ontology defined in the ITS domain. The second checks if the services selected are compatible with the user at the protocol level. There exists different notions of compatibility in synchronous communication, such as opposite behaviours, unspecified reception, and deadlock-freeness [4]. We have chosen the deadlock-freeness notion to illustrate our proposal, but other definitions could also be used. This compatibility definition guarantees that all the interactions between two services are performed in a satisfactory way, leading to a correct final state.

Secondly, once our approach discovers services, changes during the service composition may occur in many different ways. On one hand, when adaptability is anticipated and limited to some variation points (*e.g.*, software product line), the different changes to be adapted at run-time are known at design-time. On the other hand, in the unanticipated adaption, the possible variations are recognised and computed at run-time, being, for instance, new services discovered and assembled dynamically using self-awareness and environmental context information by means of planning techniques. Planning (SPP module in Figure 1) is a key feature for self-adaptive systems. Observation planning determines *when*, *where*, *what*, and *how* [6] to perform adaptation and/or evolution to solve faults. Adaptation planning aims to prepare the system to be adapted by using an *adaptation contract*. Software adaptation covers all the changes related to the anticipated adaptation. In addition, it is also characterised by highly dynamic run-time procedures that occur as devices and applications move from network to network, changing their contexts, and enhancing the flexibility and maintainability of systems. Therefore, software adaptation can also address these cases of unanticipated adaptation. Software evolution refers to the continuous changes over time, tackling other cases of unan-

ticipated adaptation, such as new requirements or services fully created at run-time.

Then, self-adaptive techniques by combining both adaptation and evolution paradigms (SCP module in Figure 1) supposes a contribution of the approach presented in this work, where the actors are the application developers and the software designers, respectively. We have made the distinction between software adaptation, where application developers generate third-party adaptors (using the adaptation contract) in a non-intrusive way, and software evolution, where software designers modify the software entities in an intrusive way and then an adaptor is generated. To perform this, we follow a two-process methodology, by modeling self-adaptive systems with a combination of anticipated and unanticipated adaptation. First, the application developers, who do not have knowledge of the source code and documentation, take advantage of adaptors to automatically adapt software when it is not necessary to modify the code. When the first process is not enough to adapt the system to the new situation because changes in the requirements or an addition/removal of a service occur, then our approach help the software designers to perform evolution. Therefore, they select a minimal set of changes to adapt software, as they are familiar with such software system. Note that this intrusive way of adapting the system requires that the designer has knowledge not only about the system, but also about our approach.

From the (matching) tuples of sets of correspondences obtained in the discovery process, we can automatically generate an adaptation contract when any fault or change occurs during the service interaction. Moreover, we also want composition to distinguish between the available contexts when translating the messages among services. Using a non-contextual approach, message correspondences are fixed. This prevents inconsistencies or changes in these connections being taken into account, and motivates the need for the new capabilities that our approach provides in order to achieve message translation depending on contexts. Therefore, we define the adaptation contract between events in the CA-STS protocols by means of vectors expressing interactions among service messages to specify the evolution of every service depending on its contexts. These interactions denote a service communication and are formalised through *synchronisation vectors* [2], which allow messages with different names and even different numbers of parameters to be synchronised. Each event appearing in one vector is executed by one service, and the overall result corresponds to a synchronisation between all the services involved. A vector may involve any number of services.

Definition 3 (Synchronisation Vector). *A synchronisation vector (or vector for short) for a set of protocols* $P_i = (A_i, S_i, I_i, Fc_i, T_i)$, $i \in \{1,..,n\}$, *is a tuple* $\langle v_1, \ldots, v_n \rangle$ *with* $v_i \in A_i \cup \{\varepsilon\}$, ε *meaning that a service does not participate in a synchronisation.*

However, vectors are not sufficient to support more advanced adaptation scenarios such as contextual rules, choice between vectors or, more generally, ordering (*e.g.*, when one message in some service corresponds to several in another service, which requires the application of several vectors). The order in which vectors have to be applied can be specified using different notations such as regular expressions, Labelled Transition Systems (LTSs), or (Hierarchical) Message Sequence Charts (MSCs). Due to their readability and user friendliness, we chose to specify adaptation contracts using LTSs whose labels are tuples. This tuple-LTS is made up of a set of tuples $\langle v, a \rangle$, where v is a vector on transitions and a indicates if v has been executed, interrupted or not executed (values can be `successful_execution`, `int_execution` or `not_executed` represented with S, I and N, respectively). Therefore, this tuple-LTS is essential in some situations in which faults, such as deadlocks or livelocks, can be avoided by applying some vectors in a specific order. If the order among correspondences between services does not matter, the tuple-LTS contains one state with all transitions looping on it.

Next, we introduce the formal notion of adaptation contract, which is used to model the composition of services making use of vectors and tuple-LTS.

Definition 4 (Adaptation Contract). *An adaptation contract for a set of services* Ws_i, $i \in 1,...,n$, *is defined as a couple* (V_{Ws_i}, T_{lts}), *where* V_{Ws_i} *is a set of vectors for services* Ws_i, *and* T_{lts} *is a tuple-LTS that indicates the interaction order of the vectors* V_{Ws_i}.

Finally, by using the adaptation contract and CA-STS services, we generate a third-party CA-STS adaptor, that is in charge of coordinating the services in the system *w.r.t.* the set of interactions defined in the contract (according to the rule COM, Figure 3). For limitations of space and since it is not a new contribution of this article, the adaptor generation is detailed in our previous works [7, 10]. Adaptor is platform independent, and it can be refined *w.r.t.* a specific platform, such as the WF platform (using our transformation process, Figure 5).

Next, we describe our fault tolerance process, which handles exceptions by using error recovery technique.

4.2 Error Recovery Mechanism

Monitoring (SMP module in Figure 1) is necessary to maintain consistency, correctness, and coordination of changes, as well as to handle errors. We focus on atomic actions, that allow programmers to apply both backward and forward error recovery. These techniques use appropriate exception handling mechanisms, which enable dealing with dependability of composed services. Exception handling is the method of building a system to detect and recover from exceptional conditions (unexpected occurrences). Protecting a system from the effects of exceptional conditions is a difficult task, since all unexpected occurrences can not be anticipated easily while designing the system. It is necessary to build exception handlers in order to detect and handle these exception conditions by avoiding application failures. We perform fault tolerance

mechanisms to handle exceptions raised by adaptive demand, returning back the system to any earlier stable state. To do that, we define an *error recovery algorithm* based on backward and forward error recovery, handling possible failures.

First, we need to define a data structure, called *vector dependency*, to track dependencies among the synchronisation vectors of an adaptation contract. We base this on the tuple-LTS previously generated to obtain the vector dependencies, since the tuple-LTS indicates the order of interactions of the vectors.

Definition 5 (Vector dependency). *A vector dependency between two vectors v_1 and v_2 is a link relationship such as $v_1 \xrightarrow{P} v_2$, where P can define both either functional or non-functional properties (such as temporal requirements or resources), and it must always be true to move from v_1 to v_2.*

Then, we define an *interaction set*, which is generated as a set of vector dependencies. This set is used to handle failures, by identifying all the vectors affected by these failures. Thus, an interaction set contains all the vectors in the adaptation contract of the communication between services involved in the interaction.

Our algorithm is executed when any failure related to networks or services occurs, by performing the following steps: (1) identify the last vector to be executed in the interaction set where the fail occurred, (2) change the status of all vectors of that interaction set whose events are directly involved in the error to `int_execution`, (3) change the status of all vectors related to other interaction sets which depended on the vectors involved in the failed interaction set to `int_execution`, (4) wait for a timeout if the service that provoked the error can be re-established, or swap the failed service with another service capable of performing similar roles, (5) if there are not services to swap, then an exception will be triggered to all the vectors involved in the error and the execution will stop, otherwise (6) re-execute all vectors in the interaction set that are labeled as `int_execution` and therefore change the value of those vectors to `successful_execution`.

Next, we illustrate both self-adaptive and error recovery processes by using the different scenarios described in our case study.

Example. Considering the full approach presented above, we address the scenarios of our case study ITS.

Firstly, common to all the scenarios, to illustrate the discovery process, we focus on the user's request (passenger or driver). Our process selects Route and Map services in that order according to the semantic matchmaking, and two (matching) tuples of sets of correspondences between operation profiles are returned and presented below (labels l_{u_1}, l_{r_1}, etc., are represented in Figure 6).

$MT_{U,R} = \{(l_{u_1}, l_{r_1}), (l_{u_2}, l_{r_2}), (l_{u_3}, l_{r_3}), (l_{u_4}, l_{r_4})\}$

$MT_{U,M} = \{(l_{u_1}, l_{m_1}), (l_{u_2}, l_{m_2}), (l_{u_3}, l_{m_3}), (l_{u_4}, l_{m_4})\}$

Once our process has discovered the services, we need to handle the inconsistencies, changes or faults which have arisen while composing services in our four scenarios.

Passenger scenario. In this scenario, our self-adaptive process applies software adaptation due to the mismatch problems in the behavioural interfaces. An adaptor is generated by means of the adaptation contract between the User (passenger) and the Route service. The contract is made up of the set of vectors presented below and the tuple-LTS depicted in Figure 7. The ITS knows at design-time the different contexts considered at run-time in this scenario, so it is enough with anticipated adaptation for the response given by the Route service.

$\{v_1 = \langle l_{u_1}, l_{r_1}\rangle, v_2 = \langle l_{u_2}, l_{r_2}\rangle,$
$v_3 = \langle l_{u_3}, l_{r_3}\rangle, v_4 = \langle l_{u_4}, l_{r_4}\rangle\}$

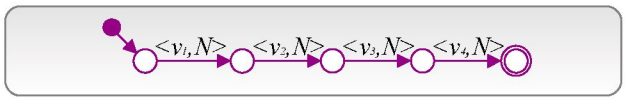


Figure 7: Tuple-LTS indicating the interaction order between the User and the Route service

It is worth mentioning that in every tuple $\langle v_i, a\rangle$, a is always initialised to N (`not_executed`), and during the composition process this value will change to either S (`successful_execution`) when the vector v_i is executed or to I when it is interrupted (`int_execution`).

Driver scenario. Here we have three different cases.

- A) and **- B)** Both cases needs unanticipated adaptation based on software evolution. Neither the new requirement (the context information related to weather) requested by drivers to obtain the rerouting, nor the new service (Parking service) provided at run-time by the driver location, were considered by the ITS at design-time. Therefore, the software designer has to modify the code of the Route service to include the weather in the context profile, and to incorporate the new Car Parking service into the ITS. Our approach will reconfigure dynamically the new full system to allow to the Users (drivers) to carry on communicating correctly with the Route service, and to discover the new service when they require it. The new CA-STS interfaces corresponding to the User and the Route, Map and Parking services are shown in Figure 8.

Note that the modifications are represented by dashed lines and bold text (*e.g.*, weather). In addition, conditions have been added (*e.g.*, $[user == \text{“}passenger\text{”}]$ in l_{u_3}) to determine that only user profile driver will request the parking service. The designer needs to know about the system and our approach to perform these kinds of modifications.

We continue focusing on the interaction between the User and Route service, but now also including the Parking service. Then, software adaptation is applied in the new service interfaces, by generating a new adaptation contract to avoid the new mismatches. Below we present the set of vectors (labels l_{u_1}, l_{r_1}, etc., are represented in Figure 8), and the tuple-LTS (Figure 9).

$\{v_1 = \langle l_{u_1}, l_{r_1}\rangle, v_2 = \langle l_{u_2}, l_{r_2}\rangle, v_3 = \langle l_{u_3}, l_{r_3}\rangle,$
$v_4 = \langle l_{u_4}, l_{r_4}\rangle, v_5 = \langle l_{u_5}, l_{p_1}\rangle, v_6 = \langle l_{u_6}, l_{p_2}\rangle,$
$v_7 = \langle l_{u_7}, l_{p_3}\rangle, v_8 = \langle l_{u_8}, l_{p_4}\rangle, v_9 = \langle l_{u_9}, l_{r_3}\rangle,$
$v_{10} = \langle l_{u_{10}}, l_{p_5}\rangle\}$

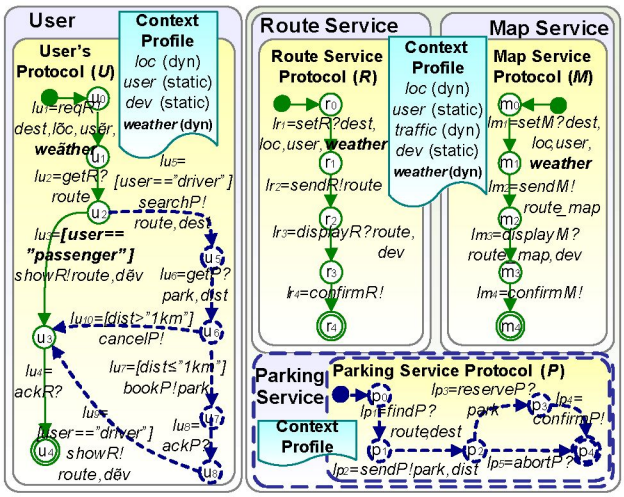


Figure 8: CA-STS of User and Route, Map and Parking services after applying our process in A) and B)

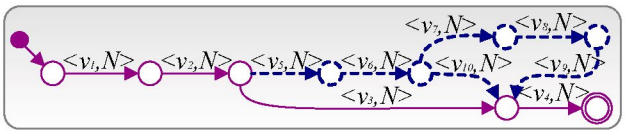


Figure 9: Tuple-LTS indicating the interaction order between the User and the Route and Parking services

Finally, we can generate an adaptor for the interaction between the User and Route and Parking services.

- C) This case considers the modifications performed in the previous cases. Our error recovery algorithm is applied here, since a suspension of the Route service during the service interaction occurred.

Before performing our algorithm, we generate the vector dependencies and the interaction sets corresponding to the communication between the User and the Route and Parking services, by means of the tuple-LTS presented in Figure 9.

$$\{v_1 \xrightarrow{P_{v_1,v_2}} v_2,\ v_2 \xrightarrow{P_{v_2,v_3}} v_3,\ v_2 \xrightarrow{P_{v_2,v_5}} v_5,$$
$$v_3 \xrightarrow{P_{v_3,v_4}} v_4,\ v_5 \xrightarrow{P_{v_5,v_6}} v_6,\ v_6 \xrightarrow{P_{v_6,v_7}} v_7,$$
$$v_6 \xrightarrow{P_{v_6,v_{10}}} v_{10},\ v_7 \xrightarrow{P_{v_7,v_8}} v_8,\ v_8 \xrightarrow{P_{v_8,v_9}} v_9,$$
$$v_9 \xrightarrow{P_{v_9,v_4}} v_4,\ v_{10} \xrightarrow{P_{v_{10},v_4}} v_4\}$$

We assume properties P_{v_i,v_j} are defined correctly according to the requirements of the user's request. The interaction sets are as follows: $I_{u,r} = \{v_1, v_2, v_3, v_4, v_9\}$ and $I_{u,p} = \{v_5, v_6, v_7, v_8, v_{10}\}$, corresponding to the communication of the User with the Route service and the User with the Parking service, respectively.

Now, to illustrate the algorithm, we assume that during the interaction between the User and the Route and Parking services, a failure occurs in vector v_4 (of the previous contract) corresponding to the confirmation of the Route service, *i.e.*, the correspondence between $l_{u_4} = ackR?$ and $l_{r_4} = confirmR?$). Therefore, our process changes to `int_execution` all the vectors involved in that error, and automatically selects another service previously considered and discovered, *i.e.*, the Map service, which replaces the Route service at run-time. This is possible because the designer developed the ITS to support a possible connection loss of the Route service, so a reconfiguration of the system is unnecessary, which reduces effort and cost.

A new adaptation contract (vectors and tuple-LTS), with its corresponding adaptor, is generated to solve the new mismatch problems in the interaction of the User and the Map and Parking services.

$\{v_1 = \langle l_{u_1}, l_{m_1}\rangle, v_2 = \langle l_{u_2}, l_{m_2}\rangle, v_3 = \langle l_{u_3}, l_{m_3}\rangle,$
$v_4 = \langle l_{u_4}, l_{m_4}\rangle, v_5 = \langle l_{u_5}, l_{p_1}\rangle, v_6 = \langle l_{u_6}, l_{p_2}\rangle,$
$v_7 = \langle l_{u_7}, l_{p_3}\rangle, v_8 = \langle l_{u_8}, l_{p_4}\rangle, v_9 = \langle l_{u_9}, l_{m_3}\rangle,$
$v_{10} = \langle l_{u_{10}}, l_{p_5}\rangle\}$

The corresponding tuple-LTS is equivalent to that presented in Figure 9, but replacing the synchronisation vectors v_1, v_2, v_3, v_4, v_9 of the previous contract (with Route service) with the vectors v_1, v_2, v_3, v_4, v_9 related to the new contract (with Map service).

5 Related work

This section compares our approach with related works in software composition, especially those which focus on model-based transformation, software adaptation and/or evolution, and error recovery.

With respect to the relationship between existing programming languages and platforms, the work presented in [5] outlines a methodology for the automated generation of adaptors capable of solving behavioural mismatches between BPEL processes (some interaction scenarios cannot be resolved). In [3], the authors present techniques to provide semi-automated support for identification and resolution of mismatches between service interfaces and protocols, and generate adaptation behavioural specifications based on SCA architecture. Compared to these works, we generate WF adaptor services that consider not only signature and protocol mismatches, but also context-aware and semantic issues. In addition, our approach is able to reorder messages among services when required, since our discovery process allows this facility automatically. This is necessary to ensure correct interaction in the case where communicating entities have messages which are not ordered as required.

Some research works have tackled software adaptation and evolution in an architecture-driven style [22], or repair programs by means of recommending adaptive changes [11]. Another important dimension is using formal methods to describe software systems more formally and to understand the cause of changes (domain structure) [8, 21]. In [19], several approaches for supporting static or dynamic adaptability and evolvability by means of a wide diversity of research domains (requirements, architecture, data, runtime and language evolution, SOAs), are presented. In our approach, we take advantage of both adaptation and/or evolution, in a model-based approach, depending on the needs of anticipated and unanticipated adaptation, and the four

categorising features, *when*, *where*, *what*, and *how*.

As regards fault tolerance mechanisms, one of the most beneficial ways of applying fault tolerance is by associating its measures with system structuring units [24]. Structuring units, which decrease system complexity and make it easier for developers to apply fault tolerance, can be: distributed transactions and atomic actions. Distributed transactions use backward error recovery [18] as the main fault tolerance measure in order to satisfy the ACID (atomicity, consistency, isolation, durability) properties. Transactions suppose a powerful abstraction to address failures occurring in closed systems. However, they impose highly severe constraints over systems in open environments such as SOA (*e.g.*, real-time systems do not have time to go back). In our approach, we use atomic actions, that allow programmers to apply both backward and forward error recovery to satisfy certain properties for composing service as a failure occur. Forward error recovery uses appropriate exception handling without impacting on the autonomy of services whilst exploiting their possible support for dependability. In addition, to handle exceptions optimally, our error recovery mechanism specifies that services return exceptions quickly, since notification delay can affect the SOA performance, especially in complex workflow systems.

Summarising, our approach combines efforts to detect and handle the different changes or faults arising in self-adaptive systems, by modelling SOA, performing adaptation and/or evolution when required, and monitoring failures with error recovery techniques.

6 Discussion and conclusions

Self-adaptive software requires high dependability, robustness, adaptability, and availability. Our approach maintains system consistency and integrity by examining each change and removing those that render the system inconsistent or unsafe. We focus on the development and maintenance of reliable software systems through self-adaptive and error recovery techniques. In addition, we give model-based SOA a push showing its usefulness to manage self-adaptive systems.

On one hand, in many occasions, the necessary effort to develop and maintain the reliable software intensive systems can be solved by using third-party services. In fact, if we weigh up the cost-effectiveness in terms of the effort required to adapt the system to changes occurred, the best solution is not modifying the code when it is not required, because an intrusive way always requires a reconfiguration of the system that is less efficient, *w.r.t.* time required, than fixing mismatch problems between services by using an adaptor. Regarding this consideration, our proposal always performs with the least effort possible to adapt the system. This is illustrated in our case study, where our approach generated an adaptor in all the situations to fix mismatches and manage context changes. Only in a 50% of cases (driver scenarios A) and B)), our approach needed to modify the system and apply reconfiguration. In a 25% of cases (driver scenario C)), it was necessary to apply error recovery mechanisms.

On the other hand, the development and maintenance of self-adaptive systems using a model-based SOA approach turns out cost-effective. First, because our self-adaptive system provides dependable services to the user, reduces the strong dependence on human resources, and reacts to different events more quickly, being capable of changing its behavior at run-time depending on the context information. Second, due to the model-based SOA facilities, such as integration, interoperability, flexibility, and incorporating of new requirements. Therefore, our model transformation process provides a level of abstraction to tackle discovery, planning, monitoring, adaptation and evolution issues easily and independently of the development platform. Our case study consisted of two kinds of users (with two profiles) and four services in total, so it was not difficult to manage. But, when an organization has a large number of services connected, the management of the service network can become extremely difficult, since all the services are directly connected, which can be unmanageable. In those cases, a model-based SOA may be even more beneficial. A first evaluation to check the scalability of our approach was obtained validating it in several examples with up to 10 services (a booking on-line system, a travel agency, an on-line computer material store, or the case study presented in this work) applied to the dependable composition of services implemented using indistinctly BPEL and WF. In a not far future, we hope that a wide number of companies adopt model-based SOA to definitively bridge the gap between business and information technology, by making the development and maintenance of large software projects more agile.

As regards future work, we plan to develop a full-scale system to check our approach that we successfully applied to a small-scale system. We also want to extend our proposal to deal with security properties in the vector dependencies, by improving the exception management in our fault tolerance mechanism, and tackling in further depth the quality of service. In addition, our approach has some limitations, such as the need of studying how to manage the complexity of the hand-code in case designers must modify the system.

Acknowledgement

This work is partially supported by the project TIN2008-05932 funded by the Spanish Ministry of Science and Innovation (MICINN) and FEDER. The authors are grateful to the anonymous referees who helped to improve the contents and quality of this article.

References

[1] T. Andrews *et al.*. *Business Process Execution Language for Web Services (WSBPEL)*. 2005.

[2] A. Arnold. *Finite Transition Systems*. International Series in Computer Science. Prentice-Hall, 1994.

[3] H. R. M. Nezhad *et al.* Semi-Automated Adaptation of Service Interactions. In *Proc. of WWW'07*, ACM, 2007.

[4] L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella. When are Two Web Services Compatible? In *Proc. of TES'04*, vol. 3324 of *LNCS*, 2004.

[5] A. Brogi and R. Popescu. Automated Generation of BPEL Adapters. In *Proc. of ICSOC'06*, volume 4294 of *LNCS*, 2006.

[6] J. Buckley *et al.* Towards a Taxonomy of Software Change. *Software Maintenance and Evolution: Research and Practice*, 17, 2005.

[7] J. Cámara, J.A. Martín, G. Salaün, J. Cubo, M. Ouederni, C. Canal, and E. Pimentel. ITACA: An Integrated Toolbox for the Automatic Composition and Adaptation of Web Services. In *Proc. of ICSE'09*, IEEE CS, 2009.

[8] C. Canal, P. Poizat, and G. Salaün. Model-Based Adaptation of Behavioural Mismatching Components. *IEEE Transactions on Software Engineering*, 34, 2008.

[9] J. Cubo, C. Canal, and E. Pimentel. Context-Aware Service Discovery and Adaptation Based on Semantic Matchmaking. In *Proc. of ICIW'10*. IEEE CS, 2010.

[10] J. Cubo *et al.* A Model-Based Approach to the Verification and Adaptation of WF/.NET Components. In *Proc. of FACS'07*, vol. 215 of *ENTCS*, 2007.

[11] B. Dagenais and M.P. Robillard. Recommending Adaptive Changes for Framework Evolution. In *Proc. of ICSE'08*, ACM, 2008.

[12] A. Dey and G. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *Proc. of Workshop on the What, Who, Where, When and How of Context-Awareness*, 2000.

[13] T. Erl. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall, 2005.

[14] H. Foster, S. Uchitel, and J. Kramer. LTSA-WS: A Tool for Model-based Verification of Web Service Compositions and Choreography. In *Proc. of ICSE'06*, ACM, 2006.

[15] X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *Proc. of WWW'04*, ACM, 2004.

[16] M. Hennessy and H. Lin. Symbolic Bisimulations. *Theor. Comput. Sci.*, 138, 1995.

[17] S. Kouadri and B. Hirsbrunner. Towards a Context-Based Service Composition Framework. In *Proc. of ICWS'03*, 2003.

[18] S. KumarGupta *et al.* Backward Error Recovery Protocols in Distributed Mobile Systems: A Survey. *Journal of Theor. and Applied Inform. Technology*, 4, 2008.

[19] T. Mens and S. Demeyer. *Software Evolution*. Springer-Verlag. 2008.

[20] R. Milner, J. Parrow, and D. Walker. Modal Logics for Mobile Processes. *Theor. Comput. Sci.*, 114, 1993.

[21] O. Nierstrasz *et al.* Model-Centric, Context-Aware Software Adaptation. In *SEAMS*, vol. 5525 of *LNCS*, 2009.

[22] P. Oreizy *et al.* An Architecture-Based Approach to Self-Adaptive Software. *IEEE Intelligent Systems*, 14, 1999.

[23] A. Gorbenko *et al.* Experimenting with Exception Propagation Mechanisms in Service-Oriented Architecture. In *Proc. of WEH'08*, ACM, 2008.

[24] F. Tartanoglu *et al.* Dependability in the Web Services Architecture. In *ADS*, vol. 2677 of *LNCS*, 2003.

[25] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Adisson-Wesley, 2nd edition, 2003.

[26] M. Zapletal. Deriving Business Service Interfaces in Windows Workflow from UMM Transactions. In *Proc. of ICSOC'08*, vol. 5364 of *LNCS*, 2008.

An Overview of Independent Component Analysis and Its Applications

Ganesh R. Naik and Dinesh K Kumar
School of Electrical and Computer Engineering
RMIT University, Australia
E-mail: ganesh.naik@rmit.edu.au

Overview paper

Keywords: independent component analysis, blind source separation, non-gaussianity, multi run ICA, overcomplete ICA, undercomplete ICA

Received: July 3, 2009

Independent Component Analysis (ICA), a computationally efficient blind source separation technique, has been an area of interest for researchers for many practical applications in various fields of science and engineering. This paper attempts to cover the fundamental concepts involved in ICA techniques and review its applications. A thorough discussion of the applications and ambiguities problems of ICA has been carried out.Different ICA methods and their applications in various disciplines of science and engineering have been reviewed. In this paper, we present ICA methods from the basics to their potential applications to serve as a comprehensive single source for an inquisitive researcher to carry out research in this field.

Povzetek: Podan je pregled tehnike ICA (Independent Component Analysis).

1 Introduction

The problem of source separation is an inductive inference problem. There is not enough information to deduce the solution, so one must use any available information to infer the most probable solution. The aim is to process these observations in such a way that the original source signals are extracted by the adaptive system. The problem of separating and estimating the original source waveforms from the sensor array, without knowing the transmission channel characteristics and the source can be briefly expressed as problems related to BSS. In BSS the word blind refers to the fact that we do not know how the signals were mixed or how they were generated. As such, the separation is in principle impossible. Allowing some relatively indirect and general constrains, we however still hold the term BSS valid, and separate under these conditions.

There appears to be something magical about blind source separation; we are estimating the original source signals without knowing the parameters of mixing and/or filtering processes. It is difficult to imagine that one can estimate this at all. In fact, without some a priori knowledge, it is not possible to uniquely estimate the original source signals. However, one can usually estimate them up to certain indeterminacies. In mathematical terms, these indeterminacies and ambiguities can be expressed as arbitrary scaling, permutation and delay of estimated source signals [1]. These indeterminacies preserve, however, the waveforms of the original sources. Although these indeterminacies seem to be rather severe limitations, in a great number of applications these limitations are not essential, since the most relevant information about the source signals is contained in the temporal waveforms or time-frequency patterns of the source signals and usually not in their amplitudes or the order in which they are arranged in the output of the system. However, for some applications especially biomedical signal models such as sEMG signals, there is no guarantee that the estimated or extracted signals have exactly the same waveforms as the source signals.

Independent component analysis (ICA) is one of the most widely used BSS techniques for revealing hidden factors that underlie sets of random variables, measurements, or signals. ICA is essentially a method for extracting individual signals from mixtures. Its power resides in the physical assumptions that the different physical processes generate unrelated signals. The simple and generic nature of this assumption allows ICA to be successfully applied in diverse range of research fields.

In this paper, we first set the scene of the blind source separation problem. Then, Independent Component Analysis is introduced as a widely used technique for solving the blind source separation problem. A general description of the approach to achieving separation via ICA and the underlying assumptions of the ICA framework and important ambiguities that are inherent to ICA are discussed in section 3. A description of specific details of different ICA methods are given in Sections 4, and the paper concludes with applications of BSS and ICA methods.

2 Blind source separation (BSS)

Consider a situation in which we have a number of sources emitting signals which are interfering with one another. Fa-

miliar situations in which this occurs are a crowded room with many people speaking at the same time, interfering electromagnetic waves from mobile phones or crosstalk from brain waves originating from different areas of the brain. In each of these situations the mixed signals are often incomprehensible and it is of interest to separate the individual signals. This is the goal of Blind Source Separation. A classic problem in BSS is the cocktail party problem. The objective is to sample a mixture of spoken voices, with a given number of microphones - the observations, and then separate each voice into a separate speaker channel - the sources. The BSS is unsupervised and thought of as a black box method. In this we encounter many problems, e.g. time delay between microphones, echo, amplitude difference, voice order in speaker and underdetermined mixture signal.

Herault and Jutten [2] proposed that, in a artificial neural network like architecture the separation could be done by reducing redundancy between signals. This approach initially lead to what is known as independent component analysis today. The fundamental research involved only a handful of researchers up until 1995. It was not until then, when Bell and Sejnowski [3] published a relatively simple approach to the problem named infomax, that many became aware of the potential of ICA. Since then a whole community has evolved around ICA, centralized around some large research groups and its own ongoing conference, International Conference on independent component analysis and blind signal separation. ICA is used today in many different applications, e.g. medical signal analysis, sound separation, image processing, dimension reduction, coding and text analysis [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14].

In ICA the general idea is to separate the signals, assuming that the original underlying source signals are mutually independently distributed. Due to the field's relatively young age, the distinction between BSS and ICA is not fully clear. When regarding ICA, the basic framework for most researchers has been to assume that the mixing is instantaneous and linear, as in infomax. ICA is often described as an extension to PCA, that uncorrelates the signals for higher order moments and produces a non-orthogonal basis. More complex models assume for example, noisy mixtures, [15, 16], nontrivial source distributions, [17, 18], convolutive mixtures [19, 20, 21], time dependency, underdetermined sources [22, 23], mixture and classification of independent component [4, 24]. A general introduction and overview can be found in [25].

3 Independent component analysis

Independent Component Analysis (ICA) is a statistical technique, perhaps the most widely used, for solving the blind source separation problem [25, 26]. In this section, we present the basic Independent Component Analysis model and show under which conditions its parameters can be estimated.

3.1 ICA model

The general model for ICA is that the sources are generated through a linear basis transformation, where additive noise can be present. Suppose we have N statistically independent signals, $s_i(t), i = 1,...,N$. We assume that the sources themselves cannot be directly observed and that each signal, $s_i(t)$, is a realization of some fixed probability distribution at each time point t. Also, suppose we observe these signals using N sensors, then we obtain a set of N observation signals $x_i(t), i = 1,...,N$ that are mixtures of the sources. A fundamental aspect of the mixing process is that the sensors must be spatially separated (e.g. microphones that are spatially distributed around a room) so that each sensor records a different mixture of the sources. With this spatial separation assumption in mind, we can model the mixing process with matrix multiplication as follows:

$$x(t) = As(t) \tag{1}$$

where $\boldsymbol{A}$ is an unknown matrix called the *mixing matrix* and $\boldsymbol{x}$(t), $\boldsymbol{s}$(t) are the two vectors representing the observed signals and source signals respectively. Incidentally, the justification for the description of this signal processing technique as *blind* is that we have no information on the mixing matrix, or even on the sources themselves.

The objective is to recover the original signals, $s_i(t)$, from only the observed vector $x_i(t)$. We obtain estimates for the sources by first obtaining the "unmixing matrix" $\boldsymbol{W}$, where, $W = A^{-1}$.

This enables an estimate, $\hat{s}(t)$, of the independent sources to be obtained:

$$\hat{s}(t) = Wx(t) \tag{2}$$

The diagram in Figure 1 illustrates both the mixing and unmixing process involved in BSS. The independent sources are mixed by the matrix **A** (which is unknown in this case). We seek to obtain a vector y that approximates s by estimating the unmixing matrix **W**. If the estimate of the unmixing matrix is accurate, we obtain a good approximation of the sources.

The above described ICA model is the simple model since it ignores all noise components and any time delay in the recordings.

3.2 Independence

A key concept that constitutes the foundation of independent component analysis is statistical independence. To simplify the above discussion consider the case of two different random variables s_1 and s_2. The random variable s_1 is independent of s_2, if the information about the value of s_1 does not provide any information about the value of s_2, and vice versa. Here s_1 and s_2 could be random signals originating from two different physical process that are not related to each other.

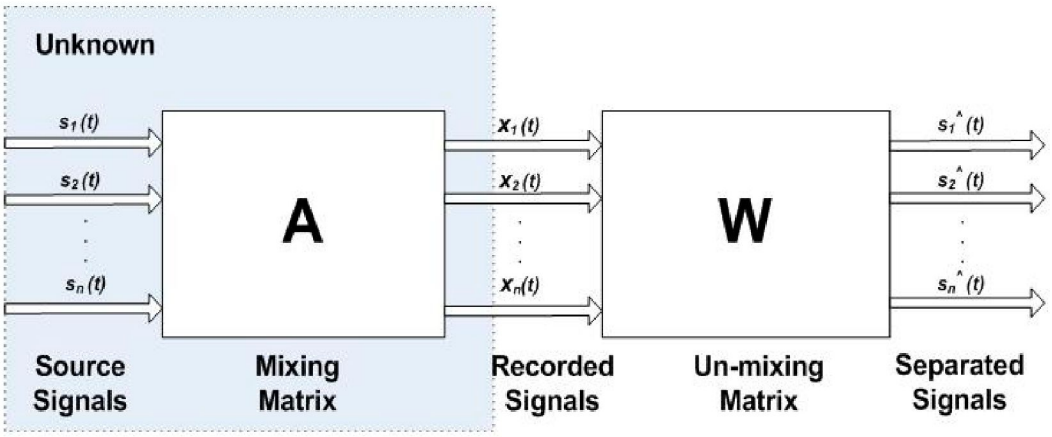


Figure 1: Blind source separation (BSS) block diagram. $s(t)$ are the sources. $x(t)$ are the recordings, $\hat{s}(t)$ are the estimated sources A is mixing matrix and W is un-mixing matrix

3.2.1 Independence definition

Mathematically, statistical independence is defined in terms of probability density of the signals. Consider the joint probability density function (pdf) of s_1 and s_2 be $p(s_1,s_2)$. Let the marginal pdf of s_1 and s_2 be denoted by $p_1(s_1)$ and $p_2(s_2)$ respectively. s_1 and s_2 are said to be independent if and only if the joint pdf can be expressed as;

$$p_{s_1,s_2}(s_1,s_2) = p_1(s_1)p_2(s_2) \tag{3}$$

Similarly, independence could be defined by replacing the pdf by the respective cumulative distributive functions as;

$$E\{p(s_1)p(s_2)\} = E\{g_1(s_1)\}E\{g_2(s_2)\} \tag{4}$$

where E{.} is the expectation operator. In the following section we use the above properties to explain the relationship between uncorrelated and independence.

3.2.2 Uncorrelatedness and Independence

Two random variables s_1 and s_2 are said to be uncorrelated if their covariance $C(s_1,s_1)$ is zero.

$$\begin{aligned} C(s_1,s_2) &= E\{(s_1-m_{s1})(s_2-m_{s2})\} \\ &= E\{s_1s_2 - s_1m_{s2} - s_2m_{s1} + m_{s1}m_{s2}\} \\ &= E\{s_1s_2\} - E\{s_1\}E\{s_2\} \\ &= 0 \end{aligned} \tag{5}$$

where m_{s1} is the mean of the signal. Equation 4 and 5 are identical for independent variables taking $g_1(s_1) = s_1$. Hence independent variables are always uncorrelated. How ever the opposite is not always true. The above discussion proves that independence is stronger than uncorrelatedness and hence independence is used as the basic principle for ICA source estimation process. However uncorrelatedness is also important for computing the mixing matrix in ICA.

3.2.3 Non-Gaussianity and Independence

According to central limit theorem the distribution of a sum of independent signals with arbitrary distributions tends toward a Gaussian distribution under certain conditions. The sum of two independent signals usually has a distribution that is closer to Gaussian than distribution of the two original signals. Thus a gaussian signal can be considered as a liner combination of many independent signals. This furthermore elucidate that separation of independent signals from their mixtures can be accomplished by making the linear signal transformation as non-Gaussian as possible.

Non-Gaussianity is an important and essential principle in ICA estimation. To use non-Gaussianity in ICA estimation, there needs to be quantitative measure of non-Gaussianity of a signal. Before using any measures of non-Gaussianity, the signals should be normalised. Some of the commonly used measures are kurtosis and entropy measures, which are explained next.

– Kurtosis

Kurtosis is the classical method of measuring Non-Gaussianity. When data is preprocessed to have unit variance, kurtosis is equal to the fourth moment of the data.

The Kurtosis of signal (s), denoted by *kurt* (s), is defined by

$$kurt(s) = E\{s^4\} - 3(E\{s^4\})^2 \tag{6}$$

This is a basic definition of kurtosis using higher order (fourth order) cumulant, this simplification is based on the assumption that the signal has zero mean. To simplify things, we can further assume that (s) has been normalised so that its variance is equal to one: $E\{s^2\} = 1$.

Hence equation 6 can be further simplified to

$$kurt(s) = E\{s^4\} - 3 \tag{7}$$

Equation 7 illustrates that kurtosis is a nomralised form of the fourth moment $E\{s^4\} = 1$. For Gaussian signal,

$E\{s^4\} = 3(E\{s^4\})^2$ and hence its kurtosis is zero. For most non-Gaussian signals, the kurtosis is nonzero. Kurtosis can be both positive or negative. Random variables that have positive kurtosis are called as *super Gaussian* or *platykurtotic*, and those with negative kurtosis are called as *sub Gaussian* or *leptokurtotic*. Non-Gaussianity is measured using the absolute value of kurtosis or the square of kurtosis.

Kurtosis has been widely used as measure of Non-Gaussianity in ICA and related fields because of its computational and theoretical and simplicity. Theoretically, it has a linearity property such that

$$kurt(s_1 \pm s_2) = kurt(s_1) \pm kurt(s_2) \tag{8}$$

and

$$kurt(\alpha s_1) = \alpha^4 kurt(s_1) \tag{9}$$

where α is a constant. Computationally kurtosis can be calculated using the fourth moment of the sample data, by keeping the variance of the signal constant.

In an intuitive sense, kurtosis measured how "spikiness" of a distribution or the size of the tails. Kurtosis is extremely simple to calculate, however, it is very sensitive to outliers in the data set. It values may be based on only a few values in the tails which means that its statistical significance is poor. Kurtosis is not robust enough for ICA. Hence a better measure of non-Gaussianity than kurtosis is required.

– Entropy

Entropy is a measure of the uniformity of the distribution of a bounded set of values, such that a complete uniformity corresponds to maximum entropy. From the information theory concept, entropy is considered as the measure of randomness of a signal. Entropy H of discrete-valued signal S is defined as

$$H(S) = -\sum P(S = a_i) log P(S = a_i) \tag{10}$$

This definition of entropy can be generalised for a continuous-valued signal (s), called differential entropy, and is defined as

$$H(S) = -\int p(s) log p(s) ds \tag{11}$$

One fundamental result of information theory is that Gaussian signal has the largest entropy among the other signal distributions of unit variance. entropy will be small for signals that have distribution concerned on certain values or have pdf that is very "spiky". Hence, entropy can be used as a measure of non-Gaussianity.

In ICA estimation, it is often desired to have a measure of non-Gaussianity which is zero for Gaussian signal and nonzero for non-Gaussian signal for computational simplicity. Entropy is closely related to the code length of the random vector. A normalised version of entropy is given by a new measure called Negentropy J which is defined as

$$J(S) = H(s_{gauss}) - H(s) \tag{12}$$

where s_{gauss} is the Gaussian signal of the same covariance matrix as (s). Equation 12 shows that Negentropy is always positive and is zero only if the signal is a pure gaussian signal. It is stable but difficult to calculate. Hence approximation must be used to estimate entropy values.

3.3 Mathematical Independence

Mathematical properties of matrices were investigated to check the linear dependency and independency of global matrices (Permutation matrix P)

3.3.1 Rank of the matrix

Rank of the matrix will be less than the matrix size for linear dependency and rank will be size of matrix for linear independency, but this couldn't be assured yet due to noise in the signal. Hence determinant is the key factor for estimating number of sources.

3.3.2 Determinant of the matrix

In real time applications Determinant value should be zero for linear independency and should be more than zero (close to 1) for linear independency [27].

3.4 ICA Assumptions and Ambiguities

ICA is distinguished from other approaches to source separation in that it requires relatively few assumptions on the sources and on the mixing process. The assumptions and of the signal properties and other conditions and the issues related to ambiguities are discussed below:

3.4.1 ICA Assumptions

– *The sources being considered are statistically independent*

The first assumption is fundamental to ICA. As discussed in Section 3.2, statistical independence is the key feature that enables estimation of the independent components $\hat{s}_(t)$ from the observations $x_i(t)$.

– *The independent components have non-Gaussian distribution*

The second assumption is necessary because of the close link between Gaussianity and independence. It is impossible to separate Gaussian sources using the ICA framework described in Section 3.2 because the sum of two or more Gaussian random variables is itself Gaussian. That is, the sum of Gaussian sources is indistinguishable from a single Gaussian source in the ICA framework, and for this reason Gaussian sources are forbidden. This is not an overly restrictive assumption as in practice most sources of interest are non-Gaussian.

– *The mixing matrix is invertible*

The third assumption is straightforward. If the mixing matrix is not invertible then clearly the unmixing matrix we seek to estimate does not even exist.

If these three assumptions are satisfied, then it is possible to estimate the independent components modulo some trivial ambiguities (discussed in Section 3.4). It is clear that these assumptions are not particularly restrictive and as a result we need only very little information about the mixing process and about the sources themselves.

3.4.2 ICA Ambiguity

There are two inherent ambiguities in the ICA framework. These are (i) magnitude and scaling ambiguity and (ii) permutation ambiguity.

– *Magnitude and scaling ambiguity*

The true variance of the independent components cannot be determined. To explain, we can rewrite the mixing in equation 1 in the form

$$\begin{aligned} x &= As \\ &= \sum_{j=1}^{N} a_j s_j \end{aligned} \tag{13}$$

where a_j denotes the jth column of the mixing matrix A. Since both the coefficients a_j of the mixing matrix and the independent components s_j are unknown, we can transform Equation 13.

$$x = \sum_{j=1}^{N} (1/\alpha_j a_j)(\alpha_j s_j) \tag{14}$$

Fortunately, in most of the applications this ambiguity is insignificant. The natural solution for this is to use assumption that each source has unit variance: $E\{s_{j^2}\} = 1$. Furthermore, the signs of the of the sources cannot be determined too. This is generally not a serious problem because the sources can be multiplied by -1 without affecting the model and the estimation

– *Permutation ambiguity*

The order of the estimated independent components is unspecified. Formally, introducing a permutation matrix P and its inverse into the mixing process in Equation 1.

$$\begin{aligned} x &= AP^{-1}Ps \\ &= A^{'}s^{'} \end{aligned} \tag{15}$$

Here the elements of P s are the original sources, except in a different order, and $A' = AP^{-1}$ is another unknown mixing matrix. Equation 15 is indistinguishable from Equation 1 within the ICA framework, demonstrating that the permutation ambiguity is inherent to Blind Source Separation. This ambiguity is to be expected Ű in separating the sources we do not seek to impose any restrictions on the order of the separated signals. Thus all permutations of the sources are equally valid.

3.5 Preprocessing

Before examining specific ICA algorithms, it is instructive to discuss preprocessing steps that are generally carried out before ICA.

3.5.1 Centering

A simple preprocessing step that is commonly performed is to "center" the observation vector x by subtracting its mean vector $m = E\{x\}$. That is then we obtain the centered observation vector, x_c, as follows:

$$x_c = x - m \tag{16}$$

This step simplifies ICA algorithms by allowing us to assume a zero mean. Once the unmixing matrix has been estimated using the centered data, we can obtain the actual estimates of the independent components as follows:

$$\hat{s}(t) = A^{-1}(x_c + m) \tag{17}$$

From this point on, all observation vectors will be assumed centered. The mixing matrix, on the other hand, remains the same after this preprocessing, so we can always do this without affecting the estimation of the mixing matrix.

3.5.2 Whitening

Another step which is very useful in practice is to prewhiten the observation vector x. Whitening involves linearly transforming the observation vector such that its components are uncorrelated and have unit variance [27]. Let x_w denote the whitened vector, then it satisfies the following equation:

$$E\{x_w x_w^T\} = I \tag{18}$$

where $E\{x_w x_w^T\}$ is the covariance matrix of x_w. Also, since the ICA framework is insensitive to the variances of the independent components, we can assume without loss of generality that the source vector, s, is white, i.e. $E\{ss^T\} = I$

A simple method to perform the whitening transformation is to use the eigenvalue decomposition (EVD) [27] of x. That is, we decompose the covariance matrix of x as follows:

$$E\{xx^T\} = VDV^T \tag{19}$$

where V is the matrix of eigenvectors of $E\{xx^T\}$, and D is the diagonal matrix of eigenvalues, i.e. $D =$

$diag\{\lambda_1, \lambda_2, ..., \lambda_n\}$. The observation vector can be whitened by the following transformation:

$$x_w = VD^{-1/2}V^T x \tag{20}$$

where the matrix $D^{-1/2}$ is obtained by a simple component wise operation as $D^{-1/2} = diag\{\lambda_1^{-1/2}, \lambda_2^{-1/2}, ..., \lambda_n^{-1/2}\}$. Whitening transforms the mixing matrix into a new one, which is orthogonal

$$x_w = VD^{-1/2}V^T As = A_w s \tag{21}$$

hence,

$$\begin{aligned} E\{x_w x_w^T\} &= A_w E\{ss^T\}A_w^T \\ &= A_w A_w^T \\ &= I \end{aligned} \tag{22}$$

Whitening thus reduces the number of parameters to be estimated. Instead of having to estimate the n^2 elements of the original matrix A, we only need to estimate the new orthogonal mixing matrix, where An orthogonal matrix has $n(n-1)/2$ degrees of freedom. One can say that whitening solves half of the ICA problem. This is a very useful step as whitening is a simple and efficient process that significantly reduces the computational complexity of ICA. An illustration of the whitening process with simple ICA source separation process is explained in the later section.

3.6 Simple Illustrations of ICA

To clarify the concepts discussed in the preceding sections two simple illustrations of ICA are presented here. The results presented below were obtained using the FastICA algorithm, but could equally well have been obtained from any of the numerous ICA algorithms that have been published in the literature (including the Bell and Sejnowski algorithm).

3.6.1 Separation of Two Signals

This section explains the simple ICA source separation process. In this illustration two independent signals, s_1 and s_2,

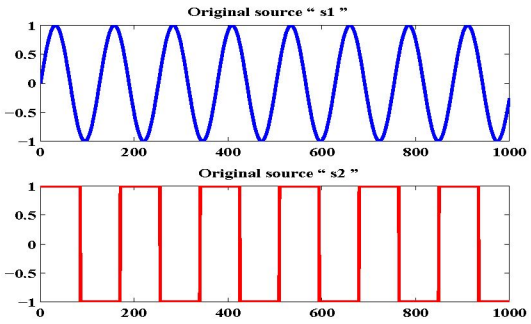


Figure 2: Independent sources $s1$ and $s2$

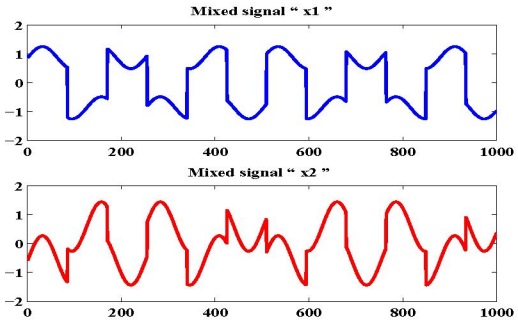


Figure 3: Observed signals, $x1$ and $x2$, from an unknown linear mixture of unknown independent components

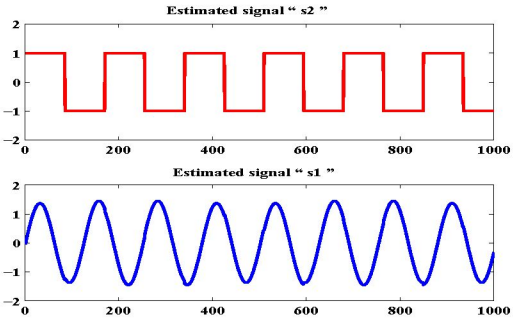


Figure 4: Estimates of independent components

are generated. These signals are shown in Figure2. The independent components are then mixed according to equation 1 using an arbitrarily chosen mixing matrix A, where

$$A = \begin{pmatrix} 0.3816 & 0.8678 \\ 0.8534 & -0.5853 \end{pmatrix}$$

The resulting signals from this mixing are shown in Figure 3. Finally, the mixtures x_1 and x_2 are separated using ICA to obtain s_1 and s_2, shown in Figure 4. By comparing Figure 4 to Figure 2 it is clear that the independent components have been estimated accurately and that the independent components have been estimated without any knowledge of the components themselves or the mixing process.

This example also provides a clear illustration of the scaling and permutation ambiguities discussed in Section 3.4. The amplitudes of the corresponding waveforms in Figures 2 and 4 are different. Thus the estimates of the independent components are some multiple of the independent components of Figure 3, and in the case of $s1$, the scaling factor is negative. The permutation ambiguity is also demonstrated as the order of the independent components has been reversed between Figure 2 and Figure 4.

3.6.2 Illustration of Statistical Independence in ICA

The previous example was a simple illustration of how ICA is used; we start with mixtures of signals and use ICA to

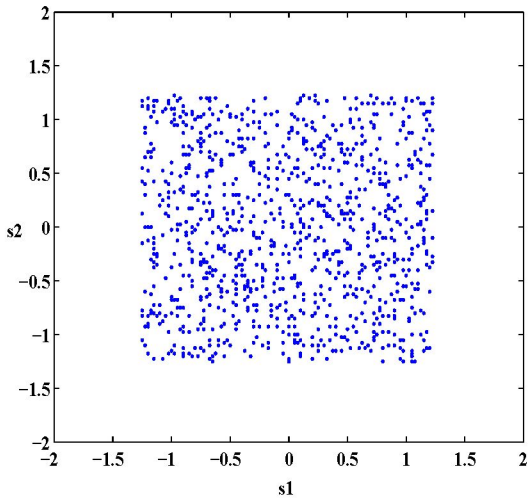


Figure 5: Original sources

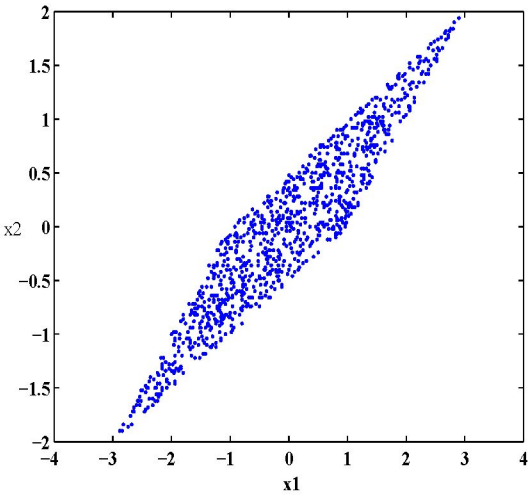


Figure 6: Mixed sources

separate them. However, this gives no insight into the mechanics of ICA and the close link with statistical independence. We assume that the independent components can be modeled as realizations of some underlying statistical distribution at each time instant (e.g. a speech signal can be accurately modeled as having a Laplacian distribution). One way of visualizing ICA is that it estimates the optimal linear transform to maximise the independence of the joint distribution of the signals X_i.

The statistical basis of ICA is illustrated more clearly in this example. Consider two random signals which are mixed using the following mixing process:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$$

Figure 5 shows the scatter-plot for original sources s_1 and s_2. Figure 6 shows the scatter-plot of the mixtures. The

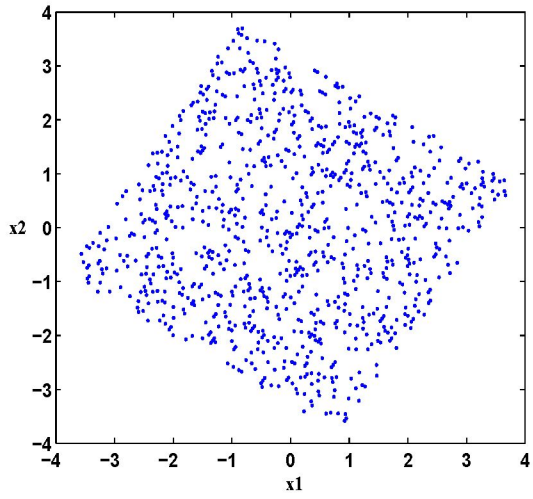


Figure 7: Joint density of whitened signals obtained from whitening the mixed sources

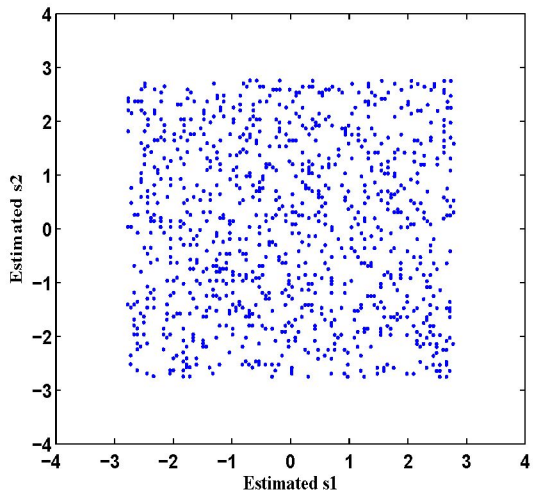


Figure 8: ICA solution (Estimated sources)

distribution along the axis x_1 and x_2 are now dependent and the form of the density is stretched according to the mixing matrix. From the Figure 6 it is clear that the two signals are not statistically independent because, for example, if x_1 = -3 or 3 then x_2 is totally determined. Whitening is an intermediate step before ICA is applied. The joint distribution that results from whitening the signals of Figure 6 is shown in Figure 7. By applying ICA, we seek to transform the data such that we obtain two independent components.

The joint distribution resulting from applying ICA to x_1 and x_2 is shown in Figure 7. This is clearly the joint distribution of two independent, uniformly distributed random variables. Independence can be intuitively confirmed as each random variable is unconstrained regardless of the value of the other random variable (this is not the case for x_1 and x_2. The uniformly distributed random variables in

Figure 8 take values between 3 and -3, but due to the scaling ambiguity, we do not know the range of the original independent components. By comparing the whitened data of Figure 7 with Figure 8, we can see that, in this case, prewhitening reduces ICA to finding an appropriate rotation to yield independence. This is a simplification as a rotation is an orthogonal transformation which requires only one parameter.

The two examples in this section are simple but they illustrate both how ICA is used and the statistical underpinnings of the process. The power of ICA is that an identical approach can be used to address problems of much greater complexity.

3.7 ICA Algorithms

There are several ICA algorithms available in literature. How ever the following three algorithms are widely used in numerous signal processing applications. These includes FastICA, JADE, and Infomax. Each algorithm used a different approach to solve equation.

3.7.1 FastICA

FastICA is a fixed point ICA algorithm that employs higher order statistics for the recovery of independent sources. FastICA can estimate ICs one by one (deflation approach) or simultaneously (symmetric approach). FastICA uses simple estimates of Negentropy based on the maximum entropy principle, which requires the use of appropriate nonlinearities for the learning rule of the neural network.

Fixed point algorithm is based on the mutual information. Which can be written as:

$$I(s) = \int f_s(s) log \frac{f_s(s)}{\prod f_{s_i}(s_i)} ds \tag{23}$$

This measure is kind of distance of independence. Minimising mutual information leads to ICA solution. For the fast ICA algorithm the above equation is re written as

$$I(s) = J(s) - \sum_i Js_i + \frac{1}{2} log \frac{\prod C_{ii}}{detC_{ss}} \tag{24}$$

where $\hat{s} = Wx$, C_{ss} is the correlation matrix, and c_{ii} is the ith diagonal element of the correlation matrix. The last term is zero because s_i are supposed to be uncorrelated. The first term is constant for a problem, because of the invariance in Negentropy. The problem is now reduced to separately maximising the Negentropy of each component.

Estimation of Negentropy is a delicate problem. The papers [28][[1] and [2] [29]

have addressed this problem. For the general version of fixed point algorithm, the approximation was based on a maximum entropy principle. The algorithm works with whitened data, although aversion of non-whitened data exists.

– Criteria

The maximisation is preferred over the following index

$$J_G(w) = [E\{G(w^T v)\} - E\{G(v)\}^2 \tag{25}$$

to find one independent component, with v standard gaussian variable, and G, the one unit contrast function.

– Update rule

Update rule for the generic algorithm is

$$\begin{aligned} w^* &= E\{vg(w^T v)\} - E\{g^{'}(w^T v)\}w \\ w &= w^*/\|w^*\| \end{aligned} \tag{26}$$

to extract one component. There is symmetric version of the FP algorithm, whose update rule is

$$\begin{aligned} W^* &= E\{g(Wv)v^T\} - Diag(E\{g^{'}(Wv)\})W \\ W &= (W^* W^{*T})^{-1/2} W^* \end{aligned} \tag{27}$$

where $Diag(v)$ is a diagonal matrix with $Diag_{ii}(v) = v_i$.

– Parameters

FastICA uses the following nonlinear parameters for convergence.

$$g(y) = \begin{cases} y^3 \\ tanh(y) \end{cases} \tag{28}$$

The choice is free except that the symmetric algorithm with *tanh* non linearity does not separate super Gaussian signals. Otherwise the choice can be devoted to the other criteria, for instance the cubic non linearity is faster, whereas the *tanh* linearity is more stable. These questions are addressed in [25]

In practice, the expectations in FastICA must be replaced by their estimates. The natural estimates are of course the corresponding sample means. Ideally, all the data available should be used, but this is often not a good idea because the computations may become too demanding. Then the averages can be estimated using a smaller sample, whose size may have a considerable effect on the accuracy of the final estimates. The sample points should be chosen separately at every iteration. If the convergence is not satisfactory, one may then increase the sample size. This thesis uses FastICA algorithm for all applications.

3.7.2 Infomax

The BSS algorithm, proposed by Bell and Sejnowski, [3], is also a gradient based neural network algorithm, with a learning rule for information maximization of information. Infomax uses higher order statistics for the information maximization. In perfect cases, it does provide the best estimate to ICA components. The strength of this algorithm comes from its direct relationship to information theory.

The algorithm is derived through an information maximisation principle, applied here between the inputs and the non linear outputs. Given the form of joint entropy

$$H(s_1,s2) = H(s_1) + H(s_2) - I(s_1,s_2) \tag{29}$$

Here for two variables $s = g(Bx)$, it is clear that maximising the joint entropy of the outputs amounts to minimising mutual information $I(y_1,y_2)$, unless it is more interesting to maximise the individual entropies than to reduce the mutual information. This is the point, where the nonlinear function plays an important role.

The basic idea of the information maximisation is to match the slope of the nonlinear function with the input probability density function. That is

$$s = g(x,\theta) \simeq \int_{-\inf}^{x} f_x(t)dt \tag{30}$$

In case of perfect matching $f_s(s)$ looks like an uniform variable, whose entropy is large. If this is not possible because the shapes are different, the best solution found in some case is to mix the input distributions so that the resulting mix matches the slope of the transfer function better than a single input distribution. In this case the algorithm does not converge, and the separation is not achieved.

– Criteria

The algorithm is a stochastic gradient ascent that maximises the joint entropy (Eqn. 12).

– Update rule

In its original form, the update rule is

$$\begin{aligned} \Delta B &= \lambda[[B^T]^{-1} + (1-2g(Bx+b_0))x^T] \\ \Delta b &= \lambda[1-2g(Bx+b_0)] \end{aligned} \tag{31}$$

– Parameters

The nonlinear function used in the original algorithm is

$$g(s) = \frac{1}{1+e^{-s}} \tag{32}$$

and in the extended version, it is

$$g(s) = s \pm tanh(s) \tag{33}$$

where the sign is that of the estimated kurtosis of the signal.

The information maximization algorithm (often referred as infomax) is widely used to separate super-Gaussian sources. Infomax is a gradient-based neural network algorithm, with a learning rule for information maximization. Infomax uses higher order statistics for the information maximization. The information maximization is attained by maximizing the joint entropy of a transformed vector. $z = g(Wx)$, where g is a point wise sigmoidal nonlinear function.

4 ICA for different conditions

One of the important conditions of ICA is that the number of sensors should be equal to the number of sources. Unfortunately, the real source separation problem does not always satisfy this constraint. This section focusses on ICA source separation problem under different conditions where the number of sources are not equal to the number of recordings.

4.1 Overcomplete ICA

Overcomplete ICA is one of the ICA source separation problem where the number of sources are greater than the number of sensors, i.e $(n > m)$. The ideas used for overcomplete ICA originally stem from coding theory, where the task is to find a representation of some signals in a given set of generators which often are more numerous than the signals, hence the term overcomplete basis. Sometimes this representation is advantageous as it uses as few 'basis' elements as possible, referred to as sparse coding. Olshausen and Field [30] first put these ideas into an information theoretic context by decomposing natural images into an overcomplete basis. Later, Harpur and Prager [31] and, independently, Olshausen [32] presented a connection between sparse coding and ICA in the square case. Lewicki and Sejnowski [22] then were the first to apply these terms to overcomplete ICA, which was further studied and applied by Lee et al. [33]. De Lathauwer et al. [34] provided an interesting algebraic approach to overcomplete ICA of three sources and two mixtures by solving a system of linear equations in the third and fourth-order cumulants, and Bofill and Zibulevsky [35] treated a special case ('delta-like' source distributions) of source signals after Fourier transformation. Overcomplete ICA has major applications in bio signal processing, due to the limited number of electrodes (recordings) compared to the number active muscles (sources) involved (in certain cases unlimited).

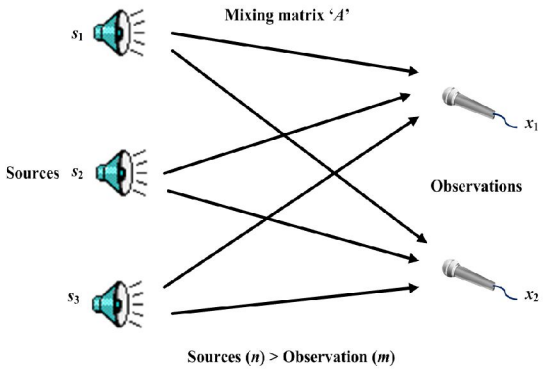


Figure 9: Illustration of "overcomplete ICA"

In overcomplete ICA, the number of sources exceed number of recordings. To analyse this, consider two recordings $x_1(t)$ and $x_2(t)$ from three independent sources $s_1(t)$, $s_2(t)$ and $s_3(t)$. The $x_i(t)$ are then weighted sums

of the $s_i(t)$, where the coefficients depend on the distances between the sources and the sensors (refer Figure 9):

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t) + a_{13}s_3(t) \quad (34)$$
$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t) + a_{23}s_3(t)$$

The a_{ij} are constant coefficients that give the mixing weights. The mixing process of these vectors can be represented in the matrix form as (refer Equation 1):

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

The unmixing process and estimation of sources can be written as (refer Equation 2):

$$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

In this example matrix A of size 2×3 matrix and unmixing matrix W is of size 3×2. Hence in overcomplete ICA it always results in pseudoinverse. Hence computation of sources in overcomplete ICA requires some estimation processes.

4.1.1 Overcomplete ICA methods

There are two common approaches of solving the overcomplete problem.

- Single step approach where the mixing matrix and the independent sources are estimated at once in a single algorithm
- Two step algorithm where the mixing matrix and the independent component values are estimated with different algorithms.

Lewicki and Sejnowski [22] proposed the single step approach, which is a natural solution to decomposition by finding the maximum a posteriori representation of the data. The prior distribution on the basis function coefficients removes the redundancy in the representation and leads to representations that are sparse and are nonlinear functions of the data. The probabilistic approach to decomposition also leads to a natural method of denoising. From this model, they derived a simple and robust learning algorithm by maximizing the data likelihood over the basis functions. Another approach in single step was proposed by Shriki et al. [36] using recurrent model, i.e., the estimated independent sources are computed taking into account the influence of other independent sources.

One of the disadvantage of single step approach is that it is complex and computationally expensive. Hence many researchers have proposed the two step method, where the mixing matrix is estimated in the first step and the sources are recovered in the next step. Zibulevsky et al. [35] proposed a sparse overcomplete ICA with delta distributions. Fabian Theis [37, 38] proposed geometric overcomplete ICA. Recently Waheed et. al [39, 40] demonstrated algebraic overcomplete ICA. In this thesis Zibulevsky's sparse overcomplete ICA is utilised, which is explained in the next section.

4.1.2 Sparse overcomplete ICA

Sparse representation of signals which is modeled by matrix factorisation has been receiving a great deal of interest in recent years. The research community has investigated many linear transforms that make audio, video and image data sparse, such as the Discrete Cosine Transform (DCT), the Fourier transform, the wavelet transform and their derivatives. [41]. Chen et al. [42] discussed sparse representations of signals by using large scale linear programming under given overcomplete basis (e.g., wavelets). Olshausen et al. [43] represented sparse coding of images based on maximum posterior approach but it was Zibulevsky et al. [35] who noticed that in the case of sparse sources, their linear mixtures can be easily separated using very simple "geometric" algorithms. Sparse representations can be used in blind source separation. When the sources are sparse, smaller coefficients are more likely and thus for a given data point t, if one of the sources is significantly larger, the remaining ones are likely to be close to zero. Thus the density of data in the mixture space, besides decreasing with the distance from the origin shows a clear tendency to cluster along the directions of the basis vectors. Sparsity is good in ICA for two reasons. First the statistical accuracy with which the mixing matrix A can be estimated is a function of how non-Gaussian the source distributions are. This suggests that the sparser the sources are the less data is needed to estimate A. Secondly the quality of the source estimates given A, is also better for sparser sources. A signal is considered sparse when values of most of the samples of the signal do not differ significantly from zero. These are from sources that are minimally active. Zibulevsky et al. [35] have demonstrated that when the signals are sparse, and the sources of these are independent, these may be separated even when the number of sources exceeds the number of recordings. [35]. The overcomplete limitation suffered by normal ICA is no longer a limiting factor for signals that are very sparse. Zibulevsky also demonstrated that when the signals are sparse, it is possible to determine the number of independent sources in a mixture of unknown signal numbers.

- Source estimation

The first step in two step approach is source separation. Here the source separation process is explained by taking sparse signal as an example. A signal is considered to be sparse if its pdf is close to Laplacian or super-Gaussian. In this case, the basic ICA model in Equation 1 is modified to have more robust representation which can be expressed as,

$$x = As + \xi \tag{35}$$

where ξ represents noise in the recordings. It is assumed that the independent sources s can be sparsely represented in a proper signal dictionary

$$s_i = \sum_{k=1}^{K} C_i^k \varphi_k \tag{36}$$

where φ_k are the *atoms* or *elements* of the dictionary. Important examples are wavelet-related dictionaries such as wavelet and wavelet packets [41]. Equation 36 can be expressed in matrix notation as

$$s = C\Phi \tag{37}$$

by substituting Equation 37 into 35 gives

$$x = AC\Phi + \xi \tag{38}$$

The goal is to estimate the mixing matrix A and the coefficients C at the same time so that C is as sparse as possible. and $X \approx AC\Phi$, given only the observed data x and the dictionary Φ
Using maximum a posteriori approach, the above goal can be expressed as

$$\max_{A,C} P(A,C|x) \propto max_{A,C} P(x|A,C)P(A)P(C) \tag{39}$$

Taking into account Equation 35 and Gaussian noise, the conditional probability $P(x|A,C)$ can be expressed as

$$P(x|A,C) \propto \prod_i exp[-\frac{(x_i - (AC\Phi)_i)^2}{2\sigma^2}] \tag{40}$$

Since C is assumed to be sparse, it can be approximated with the following pdf

$$p_i(C_i^k) \propto exp[-(\beta_i h(C_i^k))] \tag{41}$$

and hence

$$p(C) \propto \prod_{i,k} exp[-(\beta_i h(C_i^k))] \tag{42}$$

Assuming the pdf of $P(A)$ to be uniform, Equation 39 can now be simplified as

$$\max_{A,C} P(A,C|x) \propto max_{A,C} P(x|A,C)P(C) \tag{43}$$

Finally, the optimisation problem can be formed by substituting 40 and 42 into 43, taking the logarithm and inverting the sign

$$\max_{A,C} P(A,C|x) \propto min_{A,C} \frac{1}{2\sigma^2} |AC\Phi - x\|_F^2 + \sum_{i,k} (\beta_i h(C_i^k)) \tag{44}$$

There are several measures of sparsity. The simplest measure is the l_0 norm. One of the drawback of this measure is that, it is discontinuous and difficult to optimise, and also very sensitive to noise. The closest approximation of l_0 is l_1 norm. The validity of this measure can be shown by simplifying equation 44 under zero noise assumption and under Laplacian prior distributions with $h(C_i^k) = |C_i^k|$. Under these assumptions the optimisation problem can be decomposed into K smaller problems for each data point c^k at time point
$k = 1...K$ as

$$\min_{c^k} \sum_i |c_i^k| \tag{45}$$

subject to $Ac^k\varphi_k = x_k$. If small signal s is sparse in time domain then c^k in equation 45 can be uploaded with s^k.

$$\min_{s^k} \sum_i |s_i^k| \tag{46}$$

subject to $As^k = x^k$. Equation 46 can be formulated as linear programming in basic form as

$$\min_{s^k} c^T s^k| \tag{47}$$

subject to $As^k = x^k$, $s^k \geq 0$ where $s^k \Leftrightarrow [u^k; v^k], A \Leftrightarrow [A; -A]$ and $c \Leftrightarrow [1; 1]$.

– Estimating the mixing matrix

The second step in two step approach is estimating the mixing matrix. There exists various methods to compute the mixing matrix in sparse overcomplete ICA. The most widely used techniques are:

(i) C-means clustering

(ii) Algebraic method and

(iii) Potential function based method

All the above mentioned methods are based on the clustering principle. The difference is the way they estimate the direction of the clusters. The sparsity of the signal plays an important role for estimating the mixing matrix. A simple illustration that is useful to understand this concept can be found in

4.2 Undercomplete ICA

The mixture of unknown sources is referred to as undercomplete when the numbers of recordings m, more than the number of sources n. In some applications, it is desired to have more recordings than sources to achieve better separation performance. It is generally believed that with more recordings than the sources, it is always possible to get better estimate of the sources. This is not correct unless prior to separation using ICA, dimensional reduction is conducted. This can be achieved by choosing the same

number of principal recordings as the number of sources discarding the rest. To analyse this, consider three recordings $x_1(t)$, $x_2(t)$ and $x_3(t)$ from two independent sources $s_1(t)$ and $s_2(t)$. The $x_i(t)$ are then weighted sums of the $s_i(t)$, where the coefficients depend on the distances between the sources and the sensors (refer Figure 10):

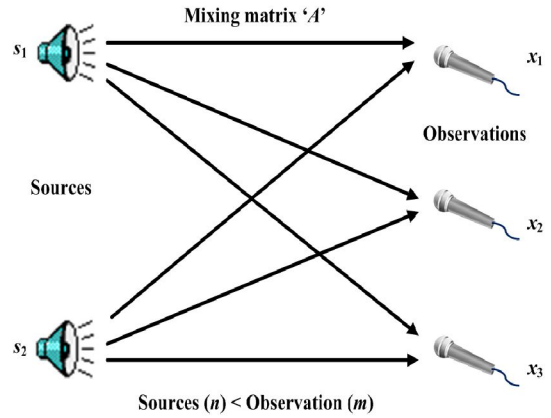


Figure 10: Illustration of "undercomplete ICA"

$$\begin{aligned} x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) \\ x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t) \\ x_3(t) &= a_{31}s_1(t) + a_{32}s_2(t) \end{aligned} \tag{48}$$

The a_{ij} are constant coefficients that gives the mixing weights. The mixing process of these vectors can be represented in the matrix form as:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$$

The unmixing process using the standard ICA requires a dimensional reduction approach so that, if one of the recordings is reduced then the square mixing matrix is obtained, which can use any standard ICA for the source estimation. For instance one of the recordings say x_3 is redundant then the above mixing process can be written as:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$$

Hence unmixing process can use any standard ICA algorithm using the following:

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The above process illustrates that, prior to source signal separation using undercomplete ICA, it is important to reduce the dimensionality of the mixing matrix and identify the required and discard the redundant recordings. Principal Component Analysis (PCA) is one of the powerful dimensional reduction method used in signal processing applications, which is explained next.

4.2.1 Undercomplete ICA using dimensional reduction method

When the number of recordings n are more than the number of sources m, there must be information redundancy in the recordings. Hence the first step is to reduce the dimensionality of the recorded data. If the dimensionality of the recorded data is equal to that of the sources, then standard ICA methods can be applied to estimate the independent sources. An example of this stages methods is illustrated in [44].

One of the popular method used in dimensional reduction method is PCA. PCA uses the decorrelated method to reduce the recorded data x using a matrix V

$$z = Vx \tag{49}$$

such that $Ezz^T = I$. The transformation matrix V is given by

$$V = D^{\frac{1}{2}}E^T \tag{50}$$

where D and E are the Eigenvalue and Eigenvector decomposition of covariance matrix C_x

$$C_x = ED^{\frac{1}{2}}E^T \tag{51}$$

Now it can be proven that

$$\begin{aligned} E\{zz^T\} &= VE\{xx^T\}V^T \\ &= D^{-1/2}E^T EDE^T ED^{-1/2} \\ &= I \end{aligned} \tag{52}$$

The second stage is using any of the standard ICA algorithms discussed in Section 3.2 to estimate the sources. In fact, whitening process through PCA is standard preprocessing in ICA. It means that applying any standard ICA algorithms that incorporates PCA will automatically reduce the dimension before running ICA.

4.3 Sub band decomposition ICA

Despite the success of using standard ICA in many applications, the basic assumptions of ICA may not hold for certain situations where there may be dependency among the signal sources. The standard ICA algorithms are not able to estimate statistically dependent original sources. One proposed technique [13] is that while there may be a degree of dependency among the wide band source signals, narrow band filtering of these signals can provide independence among these signal sources. This assumption is true when each unknown source can be modeled or represented as a linear combination of narrow-band sub-signals. Sub band decomposition ICA, an extension of ICA, assumes that each source is represented as the sum of some independent subcomponents and dependent subcomponents, which have different frequency bands.

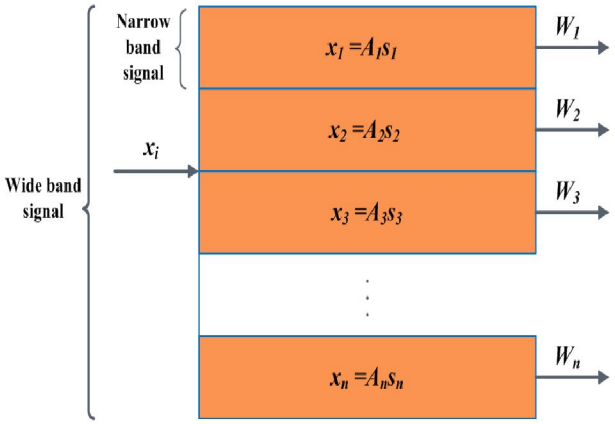


Figure 11: Sub band ICA block diagram.

Such wide-band source signals are a linear decomposition of several narrow-band sub components (refer Figure 11):

$$s(t) = s_1(t) + s_2(t) + s_3(t), \ldots, s_n(t) \tag{53}$$

Such decomposition can be modeled in the time, frequency or time frequency domains using any suitable linear transform. A set of unmixing or separating matrices: $W_1, W_2, W_3, \ldots, W_n$ are obtained where W_1 is the unmixing matrix for sensor data $x_1(t)$ and W_n is the unmixing matrix for sensor data $x_n(t)$. If the specific sub-components of interest are mutually independent for at least two sub-bands, or more generally two subsets of multi-band, say for the sub band "p" and sub band "q" then the global matrix

$$G_{pq} = W_p \times W_q^{-1} \tag{54}$$

will be a sparse generalized permutation matrix P with special structure with only one non-zero (or strongly dominating) element in each row and each column [27]. This follows from the simple mathematical observation that in such case both matrices W_p and W_q represent pseudo-inverses (or true inverse in the case of square matrix) of the same true mixing matrix A (ignoring non-essential and unavoidable arbitrary scaling and permutation of the columns) and by making an assumption that sources for two multi-frequency sub-bands are independent. This provides the basis for separation of dependent sources using narrow band pass filtered sub band signals for ICA.

4.4 Multi run ICA

One of the most effective ways of modeling vector data for unsupervised pattern classification or coding is to assume that the observations are the result of randomly picking out of a fixed set of different distributions. ICA is an iterative BSS technique. At each instance original signals are estimated from the mixed data. The quality of estimation of the original signals depends mainly on the unmixing matrix W. Due to the randomness associated with the estimation of the unmixing matrix and the iterative process, there is a randomness associated with the quality of separation.

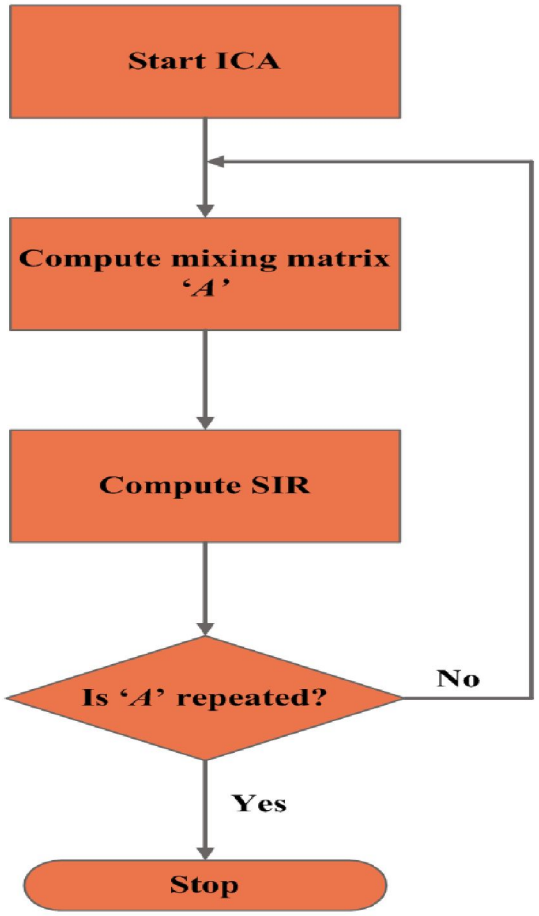


Figure 12: Multi run ICA mixing matrix computation flow chart

Multi run ICA has been proposed to overcome this associated randomness. [45]. It is the process where the ICA algorithm will be computed many times; at each instance different mixing matrices will be estimated. $A_1, A_2, \ldots, A_n$. Since it is an iterative technique with inbuilt quantisation, repeat analysis yields similarity matrices at some stage. Hence mixing matrices A_1, A_2 etc, will repeat after certain iterations. To estimate the sources from the mixed data ICA requires just one mixing matrix, the best unmixing matrix would give clear source separation, hence the selection of the best matrix is the key criterion in multi run ICA. There exists several methods to compute the quality of the mixing matrices, they are

- Signal to Noise Ratio (SNR)
- Signal to Interference Ratio (SIR)
- Signal to Distortion Ratio (SDR) and
- Signal to Artefacts Ratio (SAR)

In bio signal and audio applications, SIR has found to be a popular tool to measure the quality separation. Once the

best unmixing matrix is estimated, then any normal ICA method can be used for source separation. The multi run ICA computational process flow chart is shown in Figure 12.

5 Applications of ICA

The success of ICA in source separation has resulted in a number of practical applications. These includes,

- Machine fault detection [46, 47, 48, 49]
- Seismic monitoring [50, 51]
- Reflection canceling [52, 53]
- Finding hidden factors in financial data [54, 55, 56]
- Text document analysis [4, 5, 6]
- Radio communications [57, 58]
- Audio signal processing [20, 13]
- Image processing [13, 14, 59, 60, 61, 62, 63]
- Data mining [64]
- Time series forecasting [65]
- Defect detection in patterned display surfaces [66, ?]
- Bio medical signal processing [7, 67, 8, 9, 10, 11, 12, 68, 69].

Some of the major applications are explained in detail next:

5.1 Biomedical Applications of ICA

Exemplary ICA applications in biomedical problems include the following:

- Fetal Electrocardiogram extraction, i.e removing/filtering maternal electrocardiogram signals and noise from fetal electrocardiogram signals [70, 71].
- Enhancement of low level Electrocardiogram components [70, 71]
- Separation of transplanted heart signals from residual original heart signals [72]
- Separation of low level myoelectric muscle activities to identify various gestures [73, 74, 75, 76]

One successful and promising application domain of blind signal processing includes those biomedical signals acquired using multi-electrode devices: Electrocardiography (ECG), [77, 70, 72, 71, 78, 79, 69], Electroencephalography (EEG)[70, 71, 72, 80, 81, 82], Magnetoencephalography (MEG) [83, 84, 85, 86, 80, 87] and sEMG. Surface EMG is an indicator of muscle activity and related to body movement and posture. It has major applications in biosignal processing, next section explains sEMG and its applications.

5.2 Telecommunications

Telecommunication is one of the emerging application with respect to ICA, it has major application in code Division Multiple Access (CDMA) mobile communications. This problem is semi-blind, in the sense that certain additional prior information is available on the CDMA data model [88]. But the number of parameters to be estimated is often so high that suitable BSS, techniques taking into account the available prior knowledge, provide a clear performance improvement over more traditional estimation techniques.

5.3 Feature extraction

ICA is successfully applied for face recognition and lip reading. The goal in the face recognition is to train a system that can recognise and classify familiar faces, given a different image of the trained face. The test images may show the faces in a different pose or under different lighting conditions. Traditional methods for face recognition have employed PCA-like methods. Barlett and Sejnowski compare the face recognition performance of PCA and ICA for two different tasks:

1. different pose and
2. different lighting conditions

they show that for both the tasks, ICA outperforms PCA.

5.4 Sensor Signal Processing

A sensor network is a very recent, widely applicable and challenging field of research. As the size and cost of sensors decrease, sensor networks are increasingly becoming an attractive method to collect information in a given area. Multi-sensor data often presents complimentary information about the region surveyed and data fusion provides an effective method to enable comparison, interpretation and analysis of such data. Image and video fusion is a sub area of the more general topic of data fusion, dealing with image and video data. Cvejic et al [89] have applied the ICA for improving the fusion of multimodal surveillance images in sensor networks. ICA is also used for robust speech recognition using various sensor combinations

5.5 Audio signal processing

One of the most practical uses for BSS is in the audio world. It has been used for noise removal without the need of filters or Fourier transforms, which leads to simpler processing methods. There are various problems associated with noise removal in this way, but these can most likely be attributed to the relative infancy of the BSS field and such limitations will be reduced as research increases in this field [90, 25].

Audio source separation is the problem of automated separation of audio sources present in a room, using a set of differently placed microphones, capturing the auditory

scene. The whole problem resembles the task a human listener can solve in a cocktail party situation, where using two sensors (ears), the brain can focus on a specific source of interest, suppressing all other sources present (also known as cocktail party problem) [20, 25].

5.6 Image Processing

Recently, Independent Component Analysis (ICA) has been proposed as a generic statistical model for images [90, 59, 60, 61, 62, 63]. It is aimed at capturing the statistical structure in images that is beyond second order information, by exploiting higher-order statistical structure in data. ICA finds a linear non orthogonal coordinate system in multivariate data determined by second- and higher-order statistics. The goal of ICA is to linearly transform the data such that the transformed variables are as statistically independent from each other as possible. ICA generalizes PCA and, like PCA, has proven a useful tool for finding structure in data. Bell and Sejnowski proposed a method to extract features from natural scenes by assuming linear image synthesis model [90]. In their model, a set of digitized natural images were used. they considered each patch of an image as a linear combination of several underlying basic functions. Later Lee et al [91] proposed an image processing algorithm, which estimates the data density in each class by using parametric nonlinear functions that fit to the non-Gaussian structure of the data. They showed a significant improvement in classification accuracy over standard Gaussian mixture models. Recently Antoniol et al [92] demonstrated that the ICA model can be a suitable tool for learning a vector base for feature extraction to design a feature based data dependent approach that can be efficiently adopted for image change detection. In addition ICA features are localized and oriented and sensitive to lines and edges of varying thickness of images. Furthermore the sparsity of ICA coefficients should be pointed out. It is expected that suitable soft-thresholding on the ICA coefficients leads to efficient reduction of Gaussian noise [60, 62, 63].

6 Conclusions

This paper has introduced the fundamentals of BSS and ICA. The mathematical framework of the source mixing problem that BSS/ICA addresses was examined in some detail, as was the general approach to solving BSS/ICA. As part of this discussion, some inherent ambiguities of the BSS/ICA framework were examined as well as the two important preprocessing steps of centering and whitening. Specific details of the approach to solving the mixing problem were presented and two important ICA algorithms were discussed in detail. Finally, the application domains of this novel technique are presented. Some of the futuristic works on ICA techniques, which need further investigation are discussed. The material covered in this paper is important not only to understand the algorithms used to perform BSS/ICA, but it also provides the necessary background to understand extensions to the framework of ICA for future researchers.

References

[1] L. Tong, Liu, V. C. Soon, and Y. F. Huang, "Indeterminacy and identifiability of blind identification," *Circuits and Systems, IEEE Transactions on*, vol. 38, no. 5, pp. 499–509, 1991.

[2] C. Jutten and J. Karhunen, "Advances in blind source separation (bss) and independent component analysis (ica) for nonlinear mixtures." *Int J Neural Syst*, vol. 14, no. 5, pp. 267–292, October 2004.

[3] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution." *Neural Comput*, vol. 7, no. 6, pp. 1129–1159, November 1995.

[4] Kolenda, *Independent components in text*, ser. Advances in Independent Component Analysis. Springer-Verlag, 2000, pp. 229–250.

[5] E. Bingham, J. Kuusisto, and K. Lagus, "Ica and som in text document analysis," in *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2002, pp. 361–362.

[6] Q. Pu and G.-W. Yang, "Short-text classification based on ica and lsa," *Advances in Neural Networks - ISNN 2006*, pp. 265–270, 2006.

[7] C. J. James and C. W. Hesse, "Independent component analysis for biomedical signals," *Physiological Measurement*, vol. 26, no. 1, pp. R15+, 2005.

[8] B. Azzerboni, M. Carpentieri, F. La Foresta, and F. C. Morabito, "Neural-ica and wavelet transform for artifacts removal in surface emg," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 4, 2004, pp. 3223–3228 vol.4.

[9] F. De Martino, F. Gentile, F. Esposito, M. Balsi, F. Di Salle, R. Goebel, and E. Formisano, "Classification of fmri independent components using ic-fingerprints and support vector machine classifiers," *NeuroImage*, vol. 34, pp. 177–194, 2007.

[10] T. Kumagai and A. Utsugi, "Removal of artifacts and fluctuations from meg data by clustering methods," *Neurocomputing*, vol. 62, pp. 153–160, December 2004.

[11] Y. Zhu, T. L. Chen, W. Zhang, T.-P. Jung, J.-R. Duann, S. Makeig, and C.-K. Cheng, "Noninvasive study of the human heart using independent component analysis," in *BIBE '06: Proceedings of the Sixth IEEE*

Symposium on BionInformatics and BioEngineering. IEEE Computer Society, 2006, pp. 340–347.

[12] J. Enderle, S. M. Blanchard, and J. Bronzino, Eds., *Introduction to Biomedical Engineering, Second Edition*. Academic Press, April 2005.

[13] A. Cichocki and S.-I. Amari, *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. John Wiley & Sons, Inc., 2002.

[14] Q. Zhang, J. Sun, J. Liu, and X. Sun, "A novel ica-based image/video processing method," 2007, pp. 836–842.

[15] Hansen, *Blind separation of noicy image mixtures.* Springer-Verlag, 2000, pp. 159–179.

[16] D. J. C. Mackay, "Maximum likelihood and covariant algorithms for independent component analysis," University of Cambridge, London, Tech. Rep., 1996.

[17] Sorenson, "Mean field approaches to independent component analysis," *Neural Computation*, vol. 14, pp. 889–918, 2002.

[18] KabÂt'an, "Clustering of text documents by skewness maximization," 2000, pp. 435–440.

[19] T. W. Lee, "Blind separation of delayed and convolved sources," 1997, pp. 758–764.

[20] ——, *Independent component analysis: theory and applications*. Kluwer Academic Publishers, 1998.

[21] H. Attias and C. E. Schreiner, "Blind source separation and deconvolution: the dynamic component analysis algorithm," *Neural Comput.*, vol. 10, no. 6, pp. 1373–1424, August 1998.

[22] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations." *Neural Comput*, vol. 12, no. 2, pp. 337–365, February 2000.

[23] A. Hyvarinen, R. Cristescu, and E. Oja, "A fast algorithm for estimating overcomplete ica bases for image windows," in *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, vol. 2, 1999, pp. 894–899 vol.2.

[24] T. W. Lee, M. S. Lewicki, and T. J. Sejnowski, "Unsupervised classification with non-gaussian mixture models using ica," in *Proceedings of the 1998 conference on Advances in neural information processing systems*. Cambridge, MA, USA: MIT Press, 1999, pp. 508–514.

[25] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley-Interscience, May 2001.

[26] J. V. Stone, *Independent Component Analysis : A Tutorial Introduction (Bradford Books)*. The MIT Press, September 2004.

[27] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Cambridge, UK, 2000.

[28] P. Comon, "Independent component analysis, a new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287–314, april 1994.

[29] A. Hyvrinen, "New approximations of differential entropy for independent component analysis and projection pursuit," in *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*. MIT Press, 1998, pp. 273–279.

[30] Olshausen, "Sparse coding of natural images produces localized, oriented, bandpass receptive fields," Department of Psychology, Cornell University, Tech. Rep., 1995.

[31] G. F. Harpur and R. W. Prager, "Development of low entropy coding in a recurrent network." *Network (Bristol, England)*, vol. 7, no. 2, pp. 277–284, May 1996.

[32] B. A. Olshausen, "Learning linear, sparse, factorial codes," Tech. Rep., 1996.

[33] T. W. Lee, M. Girolami, M. S. Lewicki, and T. J. Sejnowski, "Blind source separation of more sources than mixtures using overcomplete representations," *Signal Processing Letters, IEEE*, vol. 6, no. 4, pp. 87–90, 2000.

[34] D. Lathauwer, P. L. Comon, B. De Moor, and J. Vandewalle, "Ica algorithms for 3 sources and 2 sensors," in *Higher-Order Statistics, 1999. Proceedings of the IEEE Signal Processing Workshop on*, 1999, pp. 116–120.

[35] Bofill, "Blind separation of more sources than mixtures using sparsity of their short-time fourier transform," Pajunen, Ed., 2000, pp. 87–92.

[36] O. Shriki, H. Sompolinsky, and D. D. Lee, "An information maximization approach to overcomplete and recurrent representations," in *In Advances in Neural Information Processing Systems*, vol. 14, 2002, pp. 612–618.

[37] F. J. Theis, E. W. Lang, T. Westenhuber, and C. G. Puntonet, "Overcomplete ica with a geometric algorithm," in *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*. Springer-Verlag, 2002, pp. 1049–1054.

[38] F. J. Theis and E. W. Lang, "Geometric overcomplete ica," in *Proc. of ESANN 2002*, 2002, pp. 217–223.

[39] K. Waheed and F. M. Salem, "Algebraic independent component analysis: an approach for separation of overcomplete speech mixtures," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 1, 2003, pp. 775–780 vol.1.

[40] ——, "Algebraic independent component analysis," in *Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on*, vol. 1, 2003, pp. 472–477 vol.1.

[41] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1998.

[42] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[43] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by v1?" *Vision Res*, vol. 37, no. 23, pp. 3311–3325, December 1997.

[44] M. Joho, H. Mathis, and R. Lambert, "Overdetermined blind source separation: Using more sensors than source signals in a noisy mixture," 2000.

[45] G. R. Naik, D. K. Kumar, and M. Palaniswami, "Multi run ica and surface emg based signal processing system for recognising hand gestures," in *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, 2008, pp. 700–705.

[46] A. Ypma, D. M. J. Tax, and R. P. W. Duin, "Robust machine fault detection with independent component analysis and support vector data description," in *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, 1999, pp. 67–76.

[47] Z. Li, Y. He, F. Chu, J. Han, and W. Hao, "Fault recognition method for speed-up and speed-down process of rotating machinery based on independent component analysis and factorial hidden markov model," *Journal of Sound and Vibration*, vol. 291, no. 1-2, pp. 60–71, March 2006.

[48] M. Kano, S. Tanaka, S. Hasebe, I. Hashimoto, and H. Ohno, "Monitoring independent components for fault detection," *AIChE Journal*, vol. 49, no. 4, pp. 969–976, 2003.

[49] L. Zhonghai, Z. Yan, J. Liying, and Q. Xiaoguang, "Application of independent component analysis to the aero-engine fault diagnosis," in *2009 Chinese Control and Decision Conference*. IEEE, June 2009, pp. 5330–5333.

[50] de La, C. G. Puntonet, J. M. Górriz, and I. Lloret, "An application of ica to identify vibratory low-level signals generated by termites," 2004, pp. 1126–1133.

[51] F. Acernese, A. Ciaramella, S. De Martino, M. Falanga, C. Godano, and R. Tagliaferri, "Polarisation analysis of the independent components of low frequency events at stromboli volcano (eolian islands, italy)," *Journal of Volcanology and Geothermal Research*, vol. 137, no. 1-3, pp. 153–168, September 2004.

[52] H. Farid and E. H. Adelson, "Separating reflections and lighting using independent components analysis," *cvpr*, vol. 01, 1999.

[53] M. Yamazaki, Y.-W. Chen, and G. Xu, "Separating reflections from images using kernel independent component analysis," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, 2006, pp. 194–197.

[54] M. Coli, R. Di Nisio, and L. Ippoliti, "Exploratory analysis of financial time series using independent component analysis," in *Information Technology Interfaces, 2005. 27th International Conference on*, 2005, pp. 169–174.

[55] E. H. Wu and P. L. Yu, "Independent component analysis for clustering multivariate time series data," 2005, pp. 474–482.

[56] S.-M. Cha and L.-W. Chan, "Applying independent component analysis to factor model in finance," in *IDEAL '00: Proceedings of the Second International Conference on Intelligent Data Engineering and Automated Learning, Data Mining, Financial Engineering, and Intelligent Agents*. Springer-Verlag, 2000, pp. 538–544.

[57] R. Cristescu, T. Ristaniemi, J. Joutsensalo, and J. Karhunen, "Cdma delay estimation using fast ica algorithm," vol. 2, 2000, pp. 1117–1120 vol.2.

[58] J. P. Huang and J. Mar, "Combined ica and fca schemes for a hierarchical network," *Wirel. Pers. Commun.*, vol. 28, no. 1, pp. 35–58, January 2004.

[59] O. Déniz, M. Castrillón, and M. Hernández, "Face recognition using independent component analysis and support vector machines," *Pattern Recogn. Lett.*, vol. 24, no. 13, pp. 2153–2157, 2003.

[60] S. Fiori, "Overview of independent component analysis technique with an application to synthetic aperture radar (sar) imagery processing," *Neural Netw.*, vol. 16, no. 3-4, pp. 453–467, 2003.

[61] H. Wang, Y. Pi, G. Liu, and H. Chen, "Applications of ica for the enhancement and classification of polarimetric sar images," *Int. J. Remote Sens.*, vol. 29, no. 6, pp. 1649–1663, 2008.

[62] M. S. Karoui, Y. Deville, S. Hosseini, A. Ouamri, and D. Ducrot, "Improvement of remote sensing multispectral image classification by using independent

component analysis," in *2009 First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*. IEEE, August 2009, pp. 1–4.

[63] L. Xiaochun and C. Jing, "An algorithm of image fusion based on ica and change detection," in *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.* IEEE, 2004, pp. 1096–1098.

[64] J.-H. H. Lee, S. Oh, F. A. Jolesz, H. Park, and S.-S. S. Yoo, "Application of independent component analysis for the data mining of simultaneous eeg-fmri: preliminary experience on sleep onset." *The International journal of neuroscience*, vol. 119, no. 8, pp. 1118–1136, 2009. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/19922343

[65] C.-J. Lu, T.-S. Lee, and C.-C. Chiu, "Financial time series forecasting using independent component analysis and support vector regression," *Decis. Support Syst.*, vol. 47, no. 2, pp. 115–125, 2009.

[66] D.-M. Tsai, P.-C. Lin, and C.-J. Lu, "An independent component analysis-based filter design for defect detection in low-contrast surface images," *Pattern Recogn.*, vol. 39, no. 9, pp. 1679–1694, 2006.

[67] F. Castells, J. Igual, J. Millet, and J. J. Rieta, "Atrial activity extraction from atrial fibrillation episodes based on maximum likelihood source separation," *Signal Process.*, vol. 85, no. 3, pp. 523–535, 2005.

[68] H. Safavi, N. Correa, W. Xiong, A. Roy, T. Adali, V. R. Korostyshevskiy, C. C. Whisnant, and F. Seillier-Moiseiwitsch, "Independent component analysis of 2-d electrophoresis gels," *ELECTROPHORESIS*, vol. 29, no. 19, pp. 4017–4026, 2008.

[69] R. Llinares and J. Igual, "Application of constrained independent component analysis algorithms in electrocardiogram arrhythmias," *Artif. Intell. Med.*, vol. 47, no. 2, pp. 121–133, 2009.

[70] E. Niedermeyer and F. L. Da Silva, *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields.* Lippincott Williams and Wilkins; 4th edition , January 1999.

[71] J. C. Rajapakse, A. Cichocki, and Sanchez, "Independent component analysis and beyond in brain imaging: Eeg, meg, fmri, and pet," in *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, vol. 1, 2002, pp. 404–412 vol.1.

[72] J. Wisbeck, A. Barros, and R. Ojeda, "Application of ica in the separation of breathing artifacts in ecg signals," 1998.

[73] S. Calinon and A. Billard, "Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 105–112.

[74] M. Kato, Y.-W. Chen, and G. Xu, "Articulated hand tracking by pca-ica approach," in *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition.* IEEE Computer Society, 2006, pp. 329–334.

[75] G. R. Naik, D. K. Kumar, V. P. Singh, and M. Palaniswami, "Hand gestures for hci using ica of emg," in *VisHCI '06: Proceedings of the HCSNet workshop on Use of vision in human-computer interaction.* Australian Computer Society, Inc., 2006, pp. 67–72.

[76] G. R. Naik, D. K. Kumar, H. Weghorn, and M. Palaniswami, "Subtle hand gesture identification for hci using temporal decorrelation source separation bss of surface emg," in *Digital Image Computing Techniques and Applications, 9th Biennial Conference of the Australian Pattern Recognition Society on*, 2007, pp. 30–37.

[77] M. Scherg and D. Von Cramon, "Two bilateral sources of the late aep as identified by a spatio-temporal dipole model." *Electroencephalogr Clin Neurophysiol*, vol. 62, no. 1, pp. 32–44, January 1985.

[78] R. Phlypo, V. Zarzoso, P. Comon, Y. D'Asseler, and I. Lemahieu, "Extraction of atrial activity from the ecg by spectrally constrained ica based on kurtosis sign," in *ICA'07: Proceedings of the 7th international conference on Independent component analysis and signal separation.* Berlin, Heidelberg: Springer-Verlag, 2007, pp. 641–648.

[79] J. Oster, O. Pietquin, R. Abächerli, M. Kraemer, and J. Felblinger, "Independent component analysis-based artefact reduction: application to the electrocardiogram for improved magnetic resonance imaging triggering," *Physiological Measurement*, vol. 30, no. 12, pp. 1381–1397, December 2009. [Online]. Available: http://dx.doi.org/10.1088/0967-3334/30/12/007

[80] R. Vigário, J. Särelä, V. Jousmäki, M. Hämäläinen, and E. Oja, "Independent component approach to the analysis of eeg and meg recordings." *IEEE transactions on bio-medical engineering*, vol. 47, no. 5, pp. 589–593, May 2000.

[81] J. Onton, M. Westerfield, J. Townsend, and S. Makeig, "Imaging human eeg dynamics using independent component analysis," *Neuroscience & Biobehavioral Reviews*, vol. 30, no. 6, pp. 808–822, 2006.

[82] B. Jervis, S. Belal, K. Camilleri, T. Cassar, C. Bigan, D. E. J. Linden, K. Michalopoulos, M. Zervakis, M. Besleaga, S. Fabri, and J. Muscat, "The independent components of auditory p300 and cnv evoked potentials derived from single-trial recordings," *Physiological Measurement*, vol. 28, no. 8, pp. 745–771, August 2007.

[83] J. C. Mosher, P. S. Lewis, and R. M. Leahy, "Multiple dipole modeling and localization from spatio-temporal meg data," *Biomedical Engineering, IEEE Transactions on*, vol. 39, no. 6, pp. 541–557, 1992.

[84] M. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila, and O. V. Lounasmaa, "Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain," *Reviews of Modern Physics*, vol. 65, no. 2, pp. 413+, April 1993.

[85] A. C. Tang and B. A. Pearlmutter, "Independent components of magnetoencephalography: localization," pp. 129–162, 2003.

[86] J. Parra, S. N. Kalitzin, and Lopes, "Magnetoencephalography: an investigational tool or a routine clinical technique?" *Epilepsy & Behavior*, vol. 5, no. 3, pp. 277–285, June 2004.

[87] K. Petersen, L. K. Hansen, T. Kolenda, and E. Rostrup, "On the independent components of functional neuroimages," in *Third International Conference on Independent Component Analysis and Blind Source Separation*, 2000, pp. 615–620.

[88] T. Ristaniemi and J. Joutsensalo, "the performance of blind source separation in cdma downlink," 1999.

[89] N. Cvejic, D. Bull, and N. Canagarajah, "Improving fusion of surveillance images in sensor networks using independent component analysis," *Consumer Electronics, IEEE Transactions on*, vol. 53, no. 3, pp. 1029–1035, 2007.

[90] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters." *Vision Res*, vol. 37, no. 23, pp. 3327–3338, December 1997.

[91] T.-W. W. Lee and M. S. Lewicki, "Unsupervised image classification, segmentation, and enhancement using ica mixture models." *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 11, no. 3, pp. 270–279, 2002.

[92] G. Antoniol, M. Ceccarelli, P. Petrillo, and A. Petrosino, "An ica approach to unsupervised change detection in multispectral images," in *Biological and Artificial Intelligence Environments*, B. Apolloni, M. Marinaro, and R. Tagliaferri, Eds. Springer Netherlands, 2005, ch. 35, pp. 299–311.

An Identity-Based Mediated Signature Scheme Without Trusted PKG

Xiaofeng Wang and Shangping Wang
School of Science, Xi'an university of technology, Xi'an 710048, P.R.China
E-mail: xfwang66@sina.com.cn

Keywords: ID-based signature, ID-based mediated signature, without trusted PKG, immediate revocation, GDH group

Received: July 14, 2008

Mediated Signature Scheme provides an efficient method for fast revocation of a user's identity in identity (ID)-based cryptosystems. The only ID-based mediated signature scheme was proposed by Cheng et al. from bilinear pairing in [8]. Unfortunately, their scheme has an inherent flaw that the PKG is fully capable to generate a valid mediated signature of some message on behalf of its signers by only utilizing the public information of the system. In this paper, an efficient ID-based mediated signature scheme without trusted PKG is proposed. Compared with the scheme [8], the proposed scheme has other property besides achieving immediate revocation of a signer's ID. That is, proposed scheme is ID-based, but without any assumption of pre-fixed trusted relationship between users and PKG, which effectively solves the problem that exists in some existing ID-based public key cryptosystems in which a trusted PKG and key escrow are needed.

Povzetek: Predstavljena je metoda elektronskega podpisovanja.

1 Introduction

The ID-based public key cryptosystems allow public keys of a user to be computed easily and publicly from a string to correspond with his (her) identity (such as name, telephone number, email address or an IP address). This characteristic avoids the necessity of using certificates and PKI system. Compared with certificate-based cryptosystems, ID-based cryptosystems have simplified key management since there is no need to maintain a great database containing a list of public keys and their respective owners. Therefore, ID-based public key cryptosystems have a wide application foreground in information security field. However, two inherent limitations of ID-based cryptosystems have hindered its development in implementing.

The first limitation is the necessity of a trusted party, referred to as Private Key Generator (PKG) and key escrow. In ID-based cryptosystems, a user's identity (ID) is used as his/her public key, for this work, users cannot generate their own key-pairs. As alternative, this is done by a PKG. The PKG uses a master key to generate private keys for users. Usually, PKG is supposed to be trusty. However, there is not a fully trusted PKG in practice. Since PKG knows private key of each user, a dishonest PKG can impersonate any user and forge their signatures. Recent research shows that this problem can be mitigated by splitting PKG's master key between a numbers of PKGs[1], but this adds extra complexity to key generation.

The second limitation is that current ID-based cryptosystems cannot provide an efficient solution to immediately revoke a user's identity. The typical way of revoking a user's identity is to concatenate a valid period to the identity string. Revocation is achieved by instructing PKG to stop issuing new private keys for revoked identities. This involves the need to periodically re-issue all private keys in the system, and the PKG must be online most of the time, otherwise, the user's identity cannot be immediately revoked using this method.

Boneh et al. introduced a method for obtaining fast revocation of a user's public key privilege in RSA-based cryptosystems. They call it mediated RSA (mRSA)[2]. The main idea behind mRSA is to introduce a special online entity in standard RSA, called Security Mediator (SEM). To sign or decrypt a message, user must first obtain a message-specific token from the SEM, and he(she) cannot use his (her) private key without this token. To revoke user's ability to sign or decrypt, the administrator instructs the SEM to stop issuing tokens for user's public key. Mediated RSA (mRSA) is a simple and practical method of splitting RSA private keys between the user and the SEM. Neither the user nor the SEM can cheat one another since each signature or decryption must involve both parties. mRSA allows fast revocation of user's security privileges. However, mRSA still relies on public key certificates to derive public keys. Boneh et al.[3] and Ding et al.[4] proposed Identity-based mRSA schemes, respectively. The basic idea behind identity-based mRSA is the use of a single common RSA modulus n among all users of a system. This modulus is assumed to be public and contained in a public key certificate, and the certificate is issued, as usual, by a Certificate Authority (CA). This method cannot essentially avoid the necessity of using certificates and CA.

Boneh et al. first gave a practical ID-based encryption scheme from Weil pairing [5] in 2001. Based on this scheme, Libert et al. [6], Baek et al. [7] proposed an ID-based mediated encryption scheme, respectively, using the

similar method given in mRSA. Both schemes provide efficient methods to immediately revoke a user's identity.

Very recently, Cheng et al. proposed an ID-based mediated signature scheme [8]. Their scheme avoids the using of certificates and CA. The main idea behind it is to introduce a SEM, in a general ID-based signature scheme. A signer's private key is split into two parts. One part is hold by himself(herself), and another is given to the SEM. Therefore, only with the help of the SEM, can a signer generate a valid signature. As a result, an immediate revocation of a signer's ID (i.e. a signer's signing privilege) is possible by instructing the SEM not to help the revoked user anymore. Unfortunately, their scheme has an inherent flaw, it is that the PKG is fully capable to generate a valid mediated signature of some message on behalf of its signers by only utilizing the public information of the signers and the SEM.

In this paper, we propose an ID-based mediated signature scheme from bilinear pairing. Proposed scheme has other properties besides achieving immediate revocation of signer's ID. First, our scheme is ID-based, but without any assumption of pre-fixed trusted relationship between users and PKG, it solves the problem that exists in some existing ID-based public key cryptosystems, in which a trusted PKG and key escrow are needed, in certain extent. Second, our scheme is able to prevent the dishonest PKG from impersonating the signer to generate a valid mediated signature. To construct such a scheme, we first improve Cheng's ID-based signature scheme [8], to make it has the property that the PKG is unable to generate a valid signature on behalf of any signers even if it knows the private keys of the signers, then used it to construct an efficient ID-based mediated signature scheme without trusted PKG.

The remaining sections are organized as follows. In Section 2 we briefly introduce some related mathematical knowledge. In Section 3 we recall the Cheng's ID-based mediated signature scheme. In Section 4 we propose an ID-based signature scheme, based on this scheme, we propose a new ID-based mediated signature scheme without trusted PKG and analysis its security in Section 5, then we conclude this paper in Section 6.

2 Preliminaries

2.1 Bilinear pairings

Let q be a prime with l bits length. Let G_1 be an additive cyclic group generated by P, whose order is q. Let G_2 be a multiplicative cyclic group of the same order q. A bilinear pairing is a map $\hat{e}: G_1 \times G_1 \to G_2$ satisfies the following properties:

(1) Bilinear: For any $aP, bP \in G_1$, $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab}$, where $a, b \in Z_q^*$;

(2) Non-degenerate: There exists $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1_{G_2}$;

(3) Computable: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all P, $Q \in G_1$.

2.2 Gap Diffie-Hellman (GDH) group

We consider the following problems in G_1.

(1) Discrete logarithm problem (DLP): Given $Q \in G_1$, to find an integer $x \in Z_q^*$, such that $Q = xP$ (Assuming such an integer exists.).

(2) Computational Diffie-Hellman problem (CDHP): Given $aP, bP \in G_1$, to compute abP.

(3) Decisional Diffie-Hellman problem (DDHP): Given $P, aP, bP, cP \in G_1$, to decide whether $c = ab \bmod q$, if so, (P, aP, bP, cP) is called a valid Diffie-Hellman quaternion.

Definition 2.1 We call G_1 a gap Diffie-Hellman (GDH) group if DDHP can be solved in polynomial time but there is no polynomial time algorithm to solve CDHP on G_1 with non-negligible probability.

Such a group can be found in super-singular elliptic curve or hyper-elliptic curve over finite fields. For more details, see [9,10,11,12,13,14]. An efficient method to solve DDHP is introduced in [15]: assuming there is a bilinear map $\hat{e}$, then (P, aP, bP, cP) is a valid Diffie-Hellman quaternion $\Leftrightarrow \quad \hat{e}(aP, bP) = \hat{e}(P, cP)$.

Schemes in this paper can work on any GDH group. Throughout this paper, we define the system parameters in all schemes as follows: G_1, G_2, P, q and $\hat{e}$ are as described above. These system parameters can be obtained using a GDH Parameters Generator [5,15]. Define two cryptographic hash functions: $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to Z_q^*$.

3 Cheng's ID-based mediated signature scheme and its security analysis

3.1 The scheme

Cheng's ID-based mediated signature scheme (for short CMSS) consists of three entities: PKG, SEM and signers. There are four algorithms: Setup, MeExtract, MeSign and Verify. They are described as follows:

(1) Setup: Sharing the same system parameters with above. PKG picks $s \in Z_q^*$ at randomly as a master key and computes the system public key $P_{pub} = sP$. P_{pub} is published but s is kept secretly.

(2) MeExtract: Given an identity ID, PKG chooses $s_{ID} \in Z_q^*$ at randomly, computes:

$$\begin{aligned} Q_{ID} &= H_1(ID) \\ D_{ID}^{user} &= s_{ID} \cdot Q_{ID} \\ D_{ID}^{SEM} &= (s - s_{ID}) \cdot Q_{ID} \end{aligned}$$

D_{ID}^{user} is sent secretly to the signer whose identity is ID, as the private key of the signer, and (D_{ID}^{SEM}, ID) is sent secretly to the SEM.

(3) MeSign: To sign a message M, the signer interacts with the SEM as follows:

The signer chooses $\tilde{r}_1 \in Z_q^*$ at randomly, and computes: $\tilde{R}_1 = \tilde{r}_1 P$. The triple $(M, \tilde{R}_1, ID)$ is sent to the SEM.

After having received $(M, \tilde{R}_1, ID)$, the SEM first checks the ID of the signer is not revoked. It then picks $\tilde{r}_2 \in Z_q^*$ at randomly, and computes:

$$\begin{aligned} \tilde{R}_2 &= \tilde{r}_2 P \\ \tilde{R} &= \tilde{R}_1 + \tilde{R}_2 \\ \tilde{h} &= H_2(M, \tilde{R}) \\ \tilde{S}_{SEM} &= \tilde{r}_2 P_{pub} + \tilde{h} D_{ID}^{SEM} \end{aligned}$$

Then $(\tilde{R}, \tilde{S}_{SEM})$ is sent back to the signer.

After having received $(\tilde{R}, \tilde{S}_{SEM})$, the signer computes:

$$\begin{aligned} \tilde{h} &= H_2(M, \tilde{R}) \\ \tilde{S}_{user} &= \tilde{r}_1 P_{pub} + \tilde{h} D_{ID}^{user} \\ \tilde{S} &= \tilde{S}_{user} + \tilde{S}_{SEM} \end{aligned}$$

He verifies whether $\hat{e}(P, \tilde{S}) = \hat{e}(P_{pub}, \tilde{R} + \tilde{h} Q_{ID})$ holds. If so, the signature on message M under ID is set to be $\tilde{\sigma} = (\tilde{R}, \tilde{S})$.

(4) Verification: Given a signature $\tilde{\sigma} = (\tilde{R}, \tilde{S})$ on message M under ID, the verifier computes:

$$\begin{aligned} \tilde{h} &= H_2(M, \tilde{R}) \\ Q_{ID} &= H_1(ID) \end{aligned}$$

He accepts the signature if $\hat{e}(P, \tilde{S}) = \hat{e}(P_{pub}, \tilde{R} + \tilde{h} Q_{ID})$.

3.2 Security analysis

The signature on message M under ID is:

$$\begin{aligned} \tilde{S} &= \tilde{S}_{user} + \tilde{S}_{SEM} \\ &= \tilde{r}_1 P_{pub} + \tilde{h} D_{ID}^{user} + \tilde{r}_2 P_{pub} + \tilde{h} D_{ID}^{SEM} \\ &= s(\tilde{R}_1 + \tilde{R}_2) + \tilde{h} s Q_{ID} \\ &= s\tilde{R} + \tilde{h} s Q_{ID} \end{aligned}$$

It is obvious that not only can the dishonest PKG generate every user's private key and impersonate any user to forge their signatures, but also generate a valid mediated signature on message M under ID by only utilizing the public information $(M, \tilde{R}, ID)$.

4 Improved ID-based signature scheme and its security

4.1 Improved ID-based signature scheme

Our ID-based signature scheme is based on GDH groups. It is a variant of the ID-based signature scheme given by Yi [16]. Similar variants can be seen in [8,23,24]. The security analysis of the scheme can be found in [17]. In some ID-based signature scheme such as [12,16,18], the PKG can directly forge signature by using of the signature's public information. Our construction avoids this flaw. Our ID-based signature scheme consists of four algorithms: Setup, Extract, Signing and Verification, which is described as follows.

(1) Setup: Given a security parameter l, PKG runs the GDH Parameters Generator to obtain Params = $\{G_1, G_2, P, q, \hat{e}, H_1, H_2\}$. Then it picks a random number $s \in Z_q^*$ as a master key and computes the system public key $P_{pub} = sP$. P_{pub} is published but s is kept secretly.

(2) Extract: It is a key extraction algorithm engaged by PKG and a user. A user submits and authenticates his/her identity $ID \in \{0,1\}^*$ to PKG, PKG inputs system parameters, master key and the user's identity ID; and outputs the user's public key and private key.

The signer randomly chooses an integer $r_u \in Z_q^*$, sets $R_u = r_u P$, submits (ID, R_u) to PKG, and authenticates his(her) identity to PKG by out-band mechanism. PKG generates signer's public key and private key: $Q_u = H_1(ID, R_u)$, $D_u = sQ_u$, and sends D_u to the signer via a secure channel.

(3) Signing: Given a message M, the signer picks a random number $r_v \in Z_q^*$ such that $r_v r_u \bmod q \neq 1$, and computes:

$$\begin{aligned} R_v &= r_v P \\ h &= H_2(M, ID, R_u, R_v) \\ X &= r_u r_v P + h D_u \end{aligned}$$

The signature on message M is set to be $\delta = (X, R_u, R_v)$.

(4) Verification: Given a signature $\delta = (X, R_u, R_v)$ on message M under ID, the verifier computes:

$$\begin{aligned} h &= H_2(M, ID, R_u, R_v) \\ Q_u &= H_1(ID, R_u) \end{aligned}$$

He accepts the signature if $\hat{e}(X, P) = \hat{e}(R_u, R_v)\hat{e}(hQ_u, P_{pub})$.

4.2 Security analysis

Theorem 4.1 The improved ID-based signature scheme is secure against existential forgery under adaptively chosen message and ID attack in the random oracle model.

Analysis: Generally, an ID-based signature scheme involving two security models[10]. The first is adaptively chosen message and ID attack, the second is adaptively chosen message and given ID attack. The latter is in fact the security notion of a general signature scheme. Using the same methodology as Lemma 1 in [10], we can prove that, if there exists a forger $\mathscr{A}$ who performs an existential forgery under an adaptively chosen message and ID attack against our scheme, then, making use of $\mathscr{A}$, we can construct an algorithm $\mathscr{B}$, with the same advantage as $\mathscr{A}$, against our scheme under adaptively chosen message and given ID attack.

Following certification process shows, if there exists a adversary $\mathscr{B}$ which performs an existential forgery against our scheme under adaptively chosen message and given ID

attack, then we can construct an algorithm $\mathscr{F}$ that solves the CDHP by running the adversary $\mathscr{B}$ as a subroutine.

Proof: We cannot directly reduce the security of our ID-based signature scheme to the hardness of the CDHP because our scheme contains a random value in its signature [13]. We reduce the security of our ID-based signature scheme to the hardness of the CDHP by making use of the oracle replay technology and the forking lemma [20,21].

Given an identity ID, the corresponding public/private key pair is (Q_u, D_u). If there exists an efficient algorithm $\mathscr{B}$ against our scheme under adaptively chosen message and given ID attack, then an algorithm $\mathscr{F}$ can be constructed as follows: inputs P, $P_{pub} = sP$ and $Q_u = tP$ for some $t \in Z_q^*$, if $\mathscr{B}$ chooses a message M, uses the oracle replay method and the forking lemma [20,21], $\mathscr{F}$ can obtain two valid signatures (M, R_u, R_v, h_1, X_1) and (M, R_u, R_v, h_2, X_2) such that $h_1 \neq h_2$, and satisfying equations $\hat{e}(X_1, P) = \hat{e}(R_u, R_v)\hat{e}(h_1 Q_u, P_{pub})$ and $\hat{e}(X_2, P) = \hat{e}(R_u, R_v)\hat{e}(h_2 Q_u, P_{pub})$. That is, $\hat{e}(X_1 - X_2, P) = \hat{e}((h_1 - h_2)D_u, P)$. We have $\hat{e}((X_1 - X_2) - (h_1 - h_2)D_u, P) = 1_{G_2}$. Since $\hat{e}$ has the property of non-degeneracy, we have $(X_1 - X_2) - (h_1 - h_2)D_u = O$ (here O is an ideally defined point, namely the point at infinity, and is also recognized as a point on elliptic curve.), and $D_u = (h_1 - h_2)^{-1}(X_1 - X_2)$ (see [8]). It means that $\mathscr{F}$ can solve an instance of CDHP in G_1 since $D_u = sQ_u = stP = (h_1 - h_2)^{-1}(X_1 - X_2)$.

5 Proposed scheme and its security analysis

The idea behind ID-based mediated signature is to introduce a trusted online party SEM in a general ID-based signature scheme. A private key of the signer is split into two parts. One part is given to the signer, and another is given to the SEM. Therefore, only with the help of the SEM, can a signer generate a valid mediated signature. As a result, an immediate revocation of a signer's signing privilege is possible by instructing the SEM not to help the revoked user anymore.

5.1 Our scheme

Now we give the mediated version of the improved ID-based signature scheme in Section 4.1, and show how can avoid forging signature by dishonesty PKG in our scheme. Our scheme consists of three entities: PKG, SEM and signer, and four algorithms: Setup, MeExtract, MeSign and Verification, they are described as follows:

(1) Setup: Given a security parameter l, PKG runs the GDH Parameters Generator to obtain Params= $\{G_1, G_2, P, q, \hat{e}, H_1, H_2\}$. PKG picks two different random numbers $s_1 \in Z_q^*$ and $s_2 \in Z_q^*$, lets $s = s_1 + s_2$ as the master key, and generates system public key $P_{pub} = sP$. P_{pub} is published but s_1, s_2 are kept secretly.

(2) MeExtract: the signer randomly chooses an integer $r_s \in Z_q^*$, sets $R_s = r_s P$, submits (ID, R_s) to PKG and authenticates his/her identity ID to PKG by out-band mechanism. PKG generates the public key and private key of the signer: $Q_s = H_1(ID, R_s)$, $D_s = s_1 Q_s$. PKG generates the private key of the SEM: $D_{SEM} = s_2 Q_s$. Then sent D_s to the signer, sent (D_{SEM}, ID) to the SEM, respectively, over a confidential and authentic channel.

(3) MeSign: To sign a message M, the signer must present a service requisition to SEM. He interacts with the SEM as follows:

The signer chooses a random number $r_1 \in Z_q^*$ such that $r_1 r_s \bmod q \neq 1$ and $r_1^2 \bmod q \neq 1$, computes:

$$\begin{aligned} R_1 &= r_1 P \\ Z_s &= r_s r_1 P + H_2(M, ID, R_s, R_1) D_s \end{aligned}$$

Signer sends (M, ID, R_s, R_1, Z_s) to the SEM.

After having received (M, ID, R_s, R_1, Z_s), the SEM checks that the ID of the signer is not revoked, then computes:

$$\begin{aligned} v_s &= H_2(M, ID, R_s, R_1) \\ Z &= Z_s + v_s D_{SEM} \end{aligned}$$

and verifies:

$\hat{e}(Z, P) = \hat{e}(R_s, R_1)\hat{e}(v_s H_1(ID, R_s), P_{pub})$

If so, the signer is a legitimate participant, and the SEM provides service for him/her.

SEM then picks a random number $r_2 \in Z_q^*$ such that $r_2^2 \bmod q \neq 1$, and computes:

$$\begin{aligned} R_2 &= r_2 P \\ S_{SEM} &= r_2^2 P + H_2(M, ID, R_s, R_1, R_2) D_{SEM} \end{aligned}$$

The pair (R_2, S_{SEM}) is sent to signer.

After having received (R_2, S_{SEM}), the signer computes:

$$\begin{aligned} v &= H_2(M, ID, R_s, R_1, R_2) \\ S_s &= r_1^2 P + v D_s \\ S &= S_s + S_{SEM} \end{aligned}$$

and verifies:

$\hat{e}(S, P) = \hat{e}(R_1, R_1)\hat{e}(R_2, R_2)\hat{e}(vQ_s, P_{pub})$

If so, the mediated signature on message M under ID is set to be $\sigma = (R_s, R_1, R_2, S)$.

(4) Verification: Given a signature $\sigma = (R_s, R_1, R_2, S)$ on message M under ID, the verifier computes:

$$\begin{aligned} v &= H_2(M, ID, R_s, R_1, R_2) \\ Q_s &= H_1(ID, R_s) \end{aligned}$$

He accepts the signature if and only if $\hat{e}(S, P) = \hat{e}(R_1, R_1)\hat{e}(R_2, R_2)\hat{e}(vQ_s, P_{pub})$.

5.2 Security analysis

Theorem 5.1 In our scheme, the dishonest PKG can not impersonate its signer to generate a valid mediated signature.

Analysis: We discuss the Theorem 5.1 from the following two aspects:

First, dishonest PKG can not generate a valid mediated signature by only utilizing the public information of a signer and the SEM.

For the valid mediated signature on message M under signer U's identity ID :

$$S = S_s + S_{SEM} = r_1^2 P + r_2^2 P + vsQ_s$$

Consider following impersonation attack[19]: PKG wants to impersonate the signer U to forge a mediated signature, it can do as follows:

Chooses r'_s, r'_1, $r'_2 \in Z_q^*$ at randomly, computes:

$$\begin{aligned} R'_s &= r'_s P \\ R'_1 &= r'_1 P \\ R'_2 &= r'_2 P \end{aligned}$$

Lets $Q'_s = H_1(ID, R'_s)$ as the U's public key, then PKG computes:

$$\begin{aligned} v' &= H_2(M, ID, R'_s, R'_1, R'_2) \\ S' &= {r'_1}^2 P + {r'_2}^2 P + v' s Q'_s \end{aligned}$$

Mediated signature is $\sigma' = (R'_s, R'_1, R'_2, S')$. Because $\hat{e}(S', P) = \hat{e}(R'_1, R'_1)\hat{e}(R'_2, R'_2)\hat{e}(v'Q'_s, P_{pub})$, PKG forged a *valid* mediated signature.

However, signer U can provide a proof to convince that the mediated signature is forged by PKG. To do so, he firstly sends $R_s = r_s P$ to an arbiter, and provides a *knowledge proof* that he knows $Q_s = H_1(ID, R_s)$ and private key $D_s = s_1 Q_s$; the arbiter randomly chooses a secret integer $a \in_R Z_q^*$ and sends aP to U; U then computes $B = \hat{e}(D_s, aP)$ and sends B to arbiter. If the equation $B = \hat{e}(H_1(ID, R_s), P_{pub})^a$ holds, i.e., identity ID corresponds to both $R_s = r_s P$ and $R'_s = r'_s P$. The arbiter deduces PKG dishonest because the master-key s is only known to PKG.

Second, dishonest PKG can not generate a valid mediated signature by replacing signer's secret value r_1 and SEM's secret value r_2.

Consider the following impersonation attack: PKG wants to impersonate a signer with identity ID. To do so, PKG chooses $r'_s \in Z_p^*$ at randomly, lets $R'_s = r'_s P$, $Q'_s = H_1(ID, R'_s)$, lets $D'_s = s_1 Q'_s$ as signer's private key. To sign a message M, PKG must firstly present the service requisition to SEM. It interacts with the SEM as follows:

PKG chooses $r'_1 \in Z_q^*$ at randomly, computes:

$$\begin{aligned} R'_1 &= r'_1 P \\ v'_s &= H_2(M, ID, R'_s, R'_1) \\ Z'_s &= r'_s r'_1 P + v'_s D'_s \end{aligned}$$

Then he sends $(M, ID, R'_s, R'_1, Z'_s)$ to the SEM.

The SEM checks that the signer's ID is not revoked, then computes:

$$\begin{aligned} v'_s &= H_2(M, ID, R'_s, R'_1) \\ Q'_s &= H_1(ID, R'_s) \\ Z' &= Z'_s + v'_s D_{SEM} \end{aligned}$$

It is able to find immediately $\hat{e}(Z', P) \neq \hat{e}(R'_s, R'_1)\hat{e}(v'_s H_1(ID, R'_s), P_{pub})$. Therefore, the SEM refuses to provide service for it.

Theorem 5.2 In our scheme, the only functionality of the SEM is to revoke signer's signing privilege. It cannot generate valid mediated signatures of some message on behalf of its signers.

Supposing that an attacker is able to compromise the SEM and expose the secret key D_{SEM}, it enables the SEM to *un-revoke* previously revoked, or blocks possible future revocation of current valid identities. However, the knowledge of D_{SEM} does not enable the attacker to sign messages on behalf of its signers, since the generation of a valid mediated signature needs a cooperation of the SEM and the signer. Let us consider an attacker trying to forge a signer's mediated signature on some message. Recall that the token sent to the signer by the SEM, it is a pair (R_2, S_{SEM}), where $R_2 = r_2 P$ and $S_{SEM} = r_2^2 P + vD_{SEM}$, respectively. We notice that they are all random elements in G_1, which is useless to the attacker.

Theorem 5.3 The proposed ID-based mediated signature scheme is unforgeable under the random oracle model with the assumption that G_1 is a GDH group.

According to the analysis of Theorem 4.1, if we want to proof that our scheme is secure against adaptively chosen message and ID attack, we only need to proof that our scheme is secure against adaptively chosen message and given ID attack [10]. Now we proof the latter under the random oracle model.

Lemma 5.1 If there is a forger $\mathscr{F}$ for an adaptively chosen message and *given* ID attack against our ID-based mediated signature scheme, $\mathscr{F}$ can ask queries to the oracle H_1, H_2, MeExtract, and MeSign, at most $q_{H_1}, q_{H_2}, q_E, q_S$ times, and has running time T_0 and advantage $\varepsilon_0 \geq 10(q_S + 1)(q_S + q_{H_2})/l$, then CDHP can be solved with probability $\varepsilon \geq 1/9$ within running time $T \leq (23 q_{H_2} T_0)/\varepsilon_0$.

Proof: Let G_1 be a cyclic additive group defined in Section 2. We show how to construct an algorithm $\mathscr{B}$ that compute abP for a randomly given instance $P, aP, bP \in G_1$ (where $a, b \in Z_q^*$) by running $\mathscr{F}$ as a subroutine.

During the game, $\mathscr{F}$ will consult $\mathscr{B}$ for answers to the random oracles H_1, H_2. Roughly speaking, these answers are randomly generated, but to maintain the consistency and to avoid collision, $\mathscr{B}$ keeps two lists L_1 and L_2 to store the answers. We assume that $\mathscr{F}$ will ask for $H_1(ID, \cdot)$ before ID is used as an input of any other queries.

Initialization: Fix an identity ID, lets $P_{pub} = aP$ as system public key.

ID-Hash Queries (H_1): When ID_i is submitted to H_1 oracle, $\mathscr{B}$ first scans L_1 of sorted elements (ID_i, Q_{s_i}) (where

$1 \le i \le q_{H_1}$) to check whether H_1 was already defined for that input. If it was, the previously defined value Q_{s_i} is returned. Otherwise, $\mathscr{B}$ picks $r_i \in Z_q^*$ at randomly, defines $Q_{s_i} = \begin{cases} bP, & if \quad ID_i = ID \\ r_iP, & otherwise \end{cases}$, and stores (D_i, Q_{s_i}) in L_1.

Private Key Extraction Queries (MeExtract): When $\mathscr{F}$ requests the private key associated with an identity ID_k (where $1 \le k \le q_E$), $\mathscr{B}$ recovers the corresponding (ID_k, Q_{s_k}) from L_1. Then $\mathscr{B}$ picks $u_k \in Z_q^*$ at randomly, lets $D_k = u_k Q_{s_k}$ as the private key corresponding to ID_k. Note that $\mathscr{F}$ must not ask the private key corresponding to the $ID_k = ID$.

Message-Hash Queries (H_2): When a message (M_j, ID_j) is submitted to the H_2 oracle, $\mathscr{B}$ first scans L_2 of sorted elements $(M_j, ID_j, R_{s_j}, R1_j, R2_j, v_j)$ (where $1 \le j \le q_{H_2}$) to check whether H_2 was already defined for that input. If it was, the previously defined value v_j is returned. Otherwise, $\mathscr{B}$ picks $r1_j, r2_j, v_j \in_R Z_q^*$ at randomly, returns v_j as the answer to $\mathscr{F}$, lets $R_{s_j} = r_jP$, $R1_j = r1_jP$, $R2_j = r2_jP$, and stores $(M_j, ID_j, R_{s_j}, R1_j, R2_j, v_j)$ in L_2.

Signing Queries (MeSign): If $\mathscr{F}$ asks the signature on M_t of ID_t, $\mathscr{B}$ first scans L_1 to recover the previously defined value (ID_t, Q_{s_t}), then scans L_2 to recover the previously defined value $(M_t, ID_t, R_{s_t}, R1_t, R2_t, v_t)$. Then $\mathscr{B}$ lets $S_t = r_{1_t}^2 P + r_{1_t}^2 P + r_t v_t(aP)$, and returns $\sigma_t = Sign(M_t, ID_t) = (M_t, ID_t, R_{s_t}, R1_t, R2_t, v_t, S_t)$ to $\mathscr{F}$ as the answer. Obvious, σ_t is a valid ID-based mediated signature, i.e. it satisfies the verify equation $\hat{e}(S_t, P) = \hat{e}(R1_t, R1_t)\hat{e}(R2_t, R2_t)\hat{e}(v_t Q_{s_t}, P_{pub})$.

Output: We need to take care of a nasty problem of collisions of the query result of **MeSign** and $\boldsymbol{H}_2$, as mentioned in [20] (Proof of Lemma 4). This may cause some "collision"; a query result of **MeSign** may produce a value that is inconsistent with other query results of **MeSign** or $\boldsymbol{H}_2$. In this case, $\mathscr{B}$ just outputs fail and exits. If no collisions have appeared, $\mathscr{B}$ outputs a valid signature $\sigma = (M, ID, R_s, R_1, R_2, v, S)$ with probability ε_0, which is expected to be valid for the fixed ID, without accessing any oracles except H_1, H_2. i.e. it satisfies the verification equation $\hat{e}(S, P) = \hat{e}(R_1, R_1)\hat{e}(R_2, R_2)\hat{e}(vQ_s, P_{pub})$. Considering $P_{pub} = aP$, $Q_s = bP$, we have $\hat{e}(S, P) = \hat{e}(R_1, R_1)\hat{e}(R_2, R_2)\hat{e}(vabP, P)$ ..... (1).

We apply the oracle replay technique which was invented by Pointcheval and Stern in [20,21], in which, $\mathscr{B}$ replays the same random tape but different choices of H_2, as done in the forking lemma [20], we obtain signature $(M, ID, R_s, R_1, R_2, v', S')$ with $v \ne v'$, which are expected to be valid with respect to hash function H_2' on (M, ID, R_s, R_1, R_2). So we have $\hat{e}(S', P) = \hat{e}(R_1, R_1)\hat{e}(R_2, R_2)\hat{e}(v'abP, P)$ .....(2).

From (1) and (2), we have:

$\hat{e}(S - S', P) = \hat{e}((v - v')abP, P)$

Then we obtain $abP = \frac{S - S'}{v - v'}$.

Since the oracle $\boldsymbol{H}_1$, $\boldsymbol{H}_2$, **MeExtract**, and **MeSign** generate random distribution and are indistinguishable from the original scheme, $\mathscr{F}$ learns nothing from query results. Therefore, $\mathscr{B}$ works as expected if no collisions appear in **Output**. Intuitively, since v is random, the possibility of collisions is negligible; in [20] (Proof of Theorem 3), this probability was computed explicitly, and furthermore, it was proved that the oracle replay in **Output** produces valid signatures $(M, ID, R_s, R_1, R_2, v, S)$ and $(M, ID, R_s, R_1, R_2, v', S')$ with the expected properties such that $v \ne v'$ with probability $\varepsilon \ge 1/9$ within the time $T \le (23 q_{H_2} T_0)/\varepsilon_0$.

5.3 Discussion

There are two types of possible attacks against the proposed scheme. The first comes from the choice of the random numbers used in our scheme; the second comes from the attacks on the discrete logarithm of elliptic curves. We show the details as following:

(1) Choosing appropriate random numbers

In our ID-based signature scheme (see the Section 4.1(3)), if $r_v r_u \bmod q = 1$, then $X = r_u r_v P + hD_u$ can be represented as $X = P + hD_u$, thus the signer's private key can be computed from $D_u = h^{-1}(X - P)$.

In our ID-based mediated signature scheme (see the Section 5.1(3)), if $r_1 r_s \bmod q = 1$, then $Z_s = r_s r_1 P + v_s D_s$ can be represented as $Z_s = P + v_s D_s$, thus the signer's private key can be computed by SEM from $D_s = v_s^{-1}(Z_s - P)$. If $r_1^2 \bmod q = 1$, then $S_s = r_1^2 P + vD_s$ can be represented as $S_s = P + vD_s$, thus the signer's private key can be computed by the verifier from $D_s = v^{-1}(S_s - P)$. If $r_2^2 \bmod q = 1$, then $S_{SEM} = r_2^2 P + vD_{SEM}$ can be represented as $S_{SEM} = P + vD_{SEM}$, thus the SEM's private key can be computed by the signer from $D_{SEM} = v^{-1}(S_{SEM} - P)$.

The probability of both $r_v r_u \bmod q = 1$ and $r_1 r_s \bmod q = 1$ are all $1/(q-1)$, and the probability both $r_1^2 \bmod q = 1$ and $r_2^2 \bmod q = 1$ are all $1/(q-1)$. It is neglectable when q is large enough. (e.g., The bit length of q exceed the length that defined in the international standards for elliptic curve cryptography, such as ANSIX9.62, ANSIX9.63, IEEE-P1363, ISO/IEC14888,etc..)

In our scheme, we restrict that $r_v r_u \bmod q \ne 1$, $r_1 r_s \bmod q \ne 1$, $r_1^2 \bmod q \ne 1$, and $r_2^2 \bmod q \ne 1$ to avoid these events taking place. Though similar ID-based signature scheme[16] and its variants [8,23,24] have not any restriction for choosing random numbers, and do not discuss this security flaw, but we especially emphasize such a restriction in order to make our scheme more perfect.

(2) The attacks on the discrete logarithm of elliptic curves

Our scheme is based on elliptic curve cryptography whose security relies on the difficulty to solve the discrete logarithm problem of the elliptic curve abelian group (The Elliptic Curve Discrete Logarithm Problem, ECDLP). The results show, the time complexity is exponential to break the ECDLP using the Pollard rho algorithm that is acknowledged most effective attack method for ECDLP [25,26]. However, not all the elliptic curves are suitable for cryptography. In order to guarantee the security, we must choose

secure elliptic curves whose orders are large prime numbers (e.g. Its bit length exceeds 234[26]) or include large prime factors. The result [27] provided four efficient methods to select secure elliptic curves. As long as select appropriate secure elliptic curves, as far as we know, there are not efficient methods to break ECDLP.

6 Conclusions

We improved an ID-based signature scheme and constructed an efficient ID-based mediated signature scheme from the bilinear pairing. Our ID-based mediated signature scheme has a character that the dishonest PKG can not impersonate signer to generate a valid mediated signature. Our scheme not only provides an efficient method for immediate revocation of a user's identity in ID-based public key cryptosystems, but also solves the problem that exists in some existing ID-based signatures scheme, in certain extent, in which, a trusted PKG and key escrow are needed.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (60873268); the China Postdoctoral Science Foundation (No.20080431238 and No.200902596); the Natural Science Foundation Research Plan of Shaanxi province of China (No.08JK382 and No.2009JM8004-5).

References

[1] T.Candebat, C.R.Dunne and D.Gray. (2005) Pseudonym Management using Mediated Identity-Based Cryptography, *In Advances in 2005 ACM Workshop on Digital Identity Management (DIM'05)*, Fairfax, Virginia, USA, pp.1-10.

[2] D.Boneh, X.Ding, G.Tsudik and C.Wong. (2001) A method for fast revocation of public key certificates and security capabilities, *In Advances in the 10th USENIX Security Symposium*, Washington D.C., pp.297-308.

[3] D.Boneh, X.Ding, G.Tsudik, Identity-based Mediated RSA.(2002) *In Advances in 3rd. International Workshop on Information and Security Applications*, Jeju Island, Korea, pp.192-209.

[4] X.Ding,G.Tsudik.(2003) Simple Identity-Based Cryptography with Mediated RSA. *volume 2612 of LNCS*, Springer-Verlag, pp.192-209.

[5] D.Boneh, M.Franklin.(2001) Identity-based encryption from the Weil pairings, *In Advances in Cryptology-Crypto2001,volume 2139 of LNCS*, Springer-Verlag, pp.213-229.

[6] B.Libert,J.Quisquater.(2003) Efficient revocation and threshold pairing based cryptosystems. *In Advances in 22nd Symposium on Principles of Distributed Computing*, ACM Press, pp.163-171.

[7] J.Baek and Y.Zheng.(2004) Identity-based threshold decryption. *In Advances in PKC'04, volume 2947 of LNCS*. Springer-Verlag, PP.248-261.

[8] X.Cheng, L.Guo, X.Wang.(2006) An Identity-based Mediated Signature Scheme from Bilinear Pairing. International Journal of Network Security, Vol.2, No.1, pp.29-33.

[9] Q.Wu, W.Susilo, Y.Mu, F.Zhang.(2006) Efficient Partially Blind Signatures with Provable Security. *In Advances in ICCSA 2006, volume 3982 of LNCS*,Springer-Verlag, pp.345-354.

[10] J.C.Cha and J.H.Cheon, An identity-based signature from gap Diffie-Hellman groups, *In Advances in PKC 2003, volume 2567 of LNCS*, Springer-Verlag, pp.18-30.

[11] S.D.Galbraith, K.Harrison and D.Soldera.(2002) Implementing the Tate pairings. *In Advances in ANTS 2002, volume 2369 of LNCS*, Springer-Verlag, pp.324-337.

[12] F.Hess.(2003) Efficient identity based signature schemes based on pairings. *In Advances in Select Areas in Cryptography, SAC 2002, volume 2595 of LNCS*, Springer-Verlag, pp.310-324.

[13] J.H.Cheon, Y.Kim, H.J.Yoon.(2004) A New ID-based Signature with Batch Verification, Cryptology ePrint Archive, Report 2004/131.

[14] X.Huang,Y.Mu,W.Susilo,F.Zhang.(2005) Short Designated Verifier Proxy Signature from Pairings. *In Advances in EUC Workshops 2005, volume 3823 of LNCS*. Springer-Verlag, pp.835-844.

[15] D.Boneh, B. Lynn and H. Shacham.(2001) Short signatures from the Weil-pairing. *In Advances in Asiacrypt'01, volume 2248 of LNCS*, Springer-Verlag, pp.514-532.

[16] X.Yi.(2003) An identity-based signature scheme from the Weil pairing, IEEE Communications Letters, vol.7, no.2, pp.76-78.

[17] X.Cheng,J.Liu,X.Wang.(2005) Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. *In Advances in ICCSA 2005, volume 3483 of LNCS*, Springer-Verlag, pp.1046-1054.

[18] K.Paterson.(2002) ID-based signatures from pairings on elliptic curves. Electronics Letters, vol.38, no.18, pp.1025-1026.

[19] X.Chen, F.Zhang, K.Kim.(2002) A new ID-based group signature scheme from bilinear pairings. Cryptology ePrint Archive, Report2002/184, http://eprint.iacr.org/

[20] D.Pointcheval and J.Stern.(2000) Security arguments for digital signatures and blind signatures, Journal of Cryptology, vol.13, no.3, pp.361-396.

[21] D.Pointcheval and J.Stern.(1998) Security proofs for signature schemes. *In Advances in Cryptology-Eurocrypt 96, volume 1163 of LNCS*, Springer-Verlag, pp.387-405.

[22] R.Gennaro, S.Jarecki, H.Krawczyk and T. Rabin.(1996) Robust threshold DDS signatures. *In Advances in Cryptology-Eurocrypt'96, volume 1070 of LNCS*, New York,Springer-Verlag, pp.354-371.

[23] J.Malone-Lee.(2002) Identity-Based Signcryption. Cryptology ePrint Archive, http://eprint.iacr.org /2002/098/.

[24] B.Libert, J.Quisquater.(2003) New identity based signcryption schemes from pairings. IEEE Information Theory Workshop 2003. Paris, France, Available from http://eprint.iacr.org/2003/023.

[25] Ma DaPeng, Huang JianHua.(2007) The elliptic curve cryptosystem and its security analysis. http://www.paper.edu.cn/ downloadpaper.php/serialnumber=200707-432.

[26] Huang BaoQing. The elliptic curve cryptosystem(ECC). http://www.hids.com.cn/ data.asp.

[27] J.H. Silverman.(1986) The Arithmetic of Elliptic Curves. Graduate Texts in Math., vol.106, Springer-Verlag, Berlin, Heidelberg, New York, pp.130-136.

Factors Affecting Acceptance and Use of Moodle: An Empirical Study Based on TAM

Boštjan Šumak, Marjan Heričko, Maja Pušnik and Gregor Polančič
Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, 2000 Maribor, University of Maribor, Slomškov trg 15, 2000 Maribor, Slovenia
E-mail: {Bostjan.Sumak, Marjan.Hericko, Maja.Pusnik, Gregor.Polancic}@uni-mb.si,
tel.: +386 2 220 7378, Fax: +386 2 220 7272

Keywords: e-learning, Moodle, acceptance, TAM

Received: September 19, 2009

Advancements in web technologies and the increased influence of the World Wide Web are leading to new and innovative ways of learning. New e-learning system technologies and services enable activities that allow users to be active learners, actively participating in the on-line learning process. When an e-learning system with new technologies and services is presented, it needs to be adopted by its users. The acceptance and use of an e-learning system can be influenced by different factors. The objective of this research is to examine the factors that have an impact on students' perceptions about the use and acceptance of Moodle - an open source e-learning system. In this study, the technology acceptance model (TAM) was used as an underlying theory. The data, collected from 235 students, was used to test the hypothesized research model. A data analysis was performed using structural equation modelling (SEM). The results of the analysis have revealed that the actual use of Moodle depends on two main factors: behavioural intentions and attitudes toward using Moodle. Perceived usefulness was found as the strongest and the most important predictor of attitudes toward using Moodle. Several practical and theoretical implications of the present study are discussed at the end of the paper.

Povzetek: Analizirana je uporaba sistema Moodle, odprtokodnega sistema za e-učenje.

1 Introduction

An e-learning system is a system that provides services that are necessary for handling all aspects of a course through a single, intuitive and consistent web interface. Such services are, for example: (1) course content management, (2) synchronous and asynchronous communication, (3) the uploading of content, (4) the return of students' work, (5) peer assessment, (6) student administration, (7) the collection and organization of students' grades, (8) online questionnaires, (9) online quizzes, (10) tracking tools, etc. With the advent of Web 2.0 technologies and services (like wikis, blogs, RSS, 3D virtual learning spaces, etc) e-learning systems will provide services that enable students to shift from passive to active learners where they can actively participate in the on-line learning process. E-learning environments that provide access to synchronous and asynchronous learning resources and activities are going to continue growing [1].

In addition to educational organizations, business organizations are also using e-learning technologies and services for cost-effective online training for their employees. In spite of the fact that educational and/or business institutions are investing a lot of money and resources in implementing e-learning systems, such systems will not be fully utilized if the users fail to use the system. When a new e-learning environment is presented, it needs to be adopted by its users. User's perceptions regarding the use and acceptance of an e-learning system can be affected by different factors, which can be combined into two main groups: (a) technological characteristics (like reliability, responsiveness, efficiency, security, etc.) and (b) individual characteristics (like age, gender, e-learning experience, etc.). The main challenge for e-learning system developers is to provide an e-learning system with appropriate services that will positively affect a user's experience. E-learning content providers must attract learners with appropriate e-learning content and they have to adequately incorporate e-learning services and technologies in the e-learning process. For these reasons, developers, designers and purchasers of e-learning systems must carefully consider the needs, trends and values of e-learning users and ensure that the system will meet their demands.

This study aimed to investigate the factors that affect the acceptance and use of an e-learning system, namely Moodle. Moodle provides different activity modules (like Assignments, Forums, Wikis, Blogs, Quizzes, Tracking, etc.), and can therefore be applied in different ways. Moodle can be used as a tool for delivering content to students and assess learning using assignments or quizzes and, more interestingly, it can be used to build rich collaborative learning communities. At the Faculty of Electrical Engineering and Computer Science, students

use Moodle to enroll in courses, download learning materials, communicate with other participants using forums, write blogs, contribute in content creation using wikis, communicate with professors and teaching assistants through a built-in messaging system, finish their activities and upload files, check grades, etc. Professors and teaching assistants use Moodle to manage learning content materials, manage students and their grades, check the uploaded students' work, prepare quizzes, create content using Wikis, aggregate news from different RSS feeds, etc. To understand students' perceptions about using Moodle, the technology acceptance (TAM) research model and hypothesized relationships between TAM constructs were empirically tested using the structural equation modelling (SEM) approach.

This paper is organized as follows: In the next section, theoretical backgrounds and a summary of the literature review in the field of e-learning system acceptance are given. In section three, the research model and the causal hypotheses are stated. In the section that follows, the research methodology that guided this study is described. In section five, data analysis and results are given. The last section concludes the paper with the implications and limitations of the study.

2 Theoretical backgrounds

Adoption of an e-learning system by learners may be treated as technology adoption. The most common theory in the field of IT/IS (information technology/information system) adoption is the Technology Acceptance Model – TAM. Davis [2] proposed TAM to explain the potential user's behavioural intentions when using a technological innovation, because it explains the causal links between beliefs (the usefulness of a system and ease of use of a system) and users' attitudes, intentions, and the actual usage of the system. The principal TAM concepts are (see Figure 1):

- perceived ease of use (PEOU) – the degree to which a person believes that using a particular system would be free of effort,
- perceived usefulness (PU) – the degree to which a person believes that using a particular system would enhance his or her job performance, and
- the dependent variable behavioural intention (BI) – the degree to which a person has formulated conscious plans to perform or not perform some specified future behaviour.

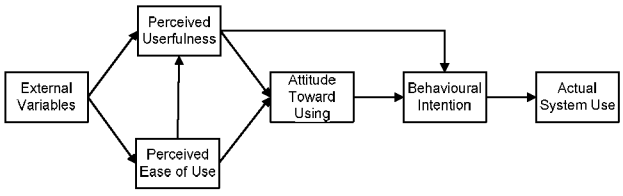


Figure 1: Technology Acceptance Model (TAM) [2].

Davis et al. [3] stated that one purpose of TAM is to serve as a starting point for examining the impact that external variables can have on behavioural intentions. Over time, TAM has progressed through a rigorous development process, since its enormous flexibility allows it to be extended. TAM has become one of the most widely used models in IS research because of its understandability and simplicity [4]. Because of TAM's demonstrated adaptability [5], it can also be used as a model for investigating user requirements and factors important for e-services, or specifically, the usefulness and simplicity of e-learning.

When doing a research about acceptance and use of an information technology, authors usually perform one of the following two types of studies: (1) an empirical validation of TAM (or other theory) in the context of a specific information technology, and (2) an extension of the theoretical model (for example the TAM model) with user specific factors. In existing literature, we can find mixed results (see also Table 1) about the importance of these determinants. For example, in [6] the PEOU was not a significant predictor of attitudes toward use (ATU) and intent of using an e-learning system. Van Raaij and Schepers [7] also did not find a significant connection between PEOU and intention of using the e-learning system. On the other hand, in the study performed by Ngai, Poon, & Chan [8], PEOU demonstrated it to be a dominant determinant of the attitude of students using an e-learning system. The statistical significance of the path between the PEOU and attitudes towards using an e-learning system was also found by Liu, Liao, & Pratt [9] where the authors studied a user's acceptance of streaming media for e-learning. Results across different studies revealed several antecedent factors to PU and PEOU. In TAM2 [10] the subjective norm, image, job relevance, and result demonstrability were found to be significant determinants of PU. It was also shown that subjective norms, PU and PEOU were direct determinants of intentions of use. We performed a meta-analysis of existing literature in the field of e-learning acceptance, where TAM was used as a ground theory. The goal of the literature review was to analyze the findings about the causal links between the main TAM constructs. The literature review in the field of e-learning acceptance showed that PEOU is the factor that mostly affects PU. PU is also an important determinant of BI. ATU is mostly positively affected by PU and PEOU.

Table 1 summarizes the positive and negative causal links between main TAM constructs together with some user specific factors, which were shown to be a direct determinant for a specific TAM construct in several studies. The list of all user-specific factors is actually larger, but because of the space limit we enlisted only those that were discussed in two or more studies. These factors are: computer self efficacy (CSE), confidentiality (CONF), computer anxiety (CANX), self efficacy (SEFF), subjective norms (SN), and enjoyment (ENJ).

Table 1: The results of existing research in the field of e-learning acceptance.

Causal Relationship					
Independent Variable	**Dependent Variable**	**Positive**	**Positive ^{NS}**	**Negative**	**Negative ^{NS}**
PEOU	PU	14	1	0	0
PU	BI	13	0	0	0
PEOU	BI	6	2	0	0
PU	ATU	6	0	0	0
PEOU	ATU	5	1	0	0
ATU	BI	4	0	0	0
BI	U	2	0	0	0
ATU	U	0	1	0	0
User specific or contextual factors affecting TAM constructs					
CSE	PEOU	6	0	0	0
CSE	PU	2	2	0	0
CONF	PU	3	0	0	0
CANX	PEOU	0	1	2	0
SN	PU	3	0	0	0
SEFF	PU	3	0	0	0
ENJ	BI	3	0	0	0
ENJ	ATU	2	0	0	0
ENJ	PU	2	0	0	0

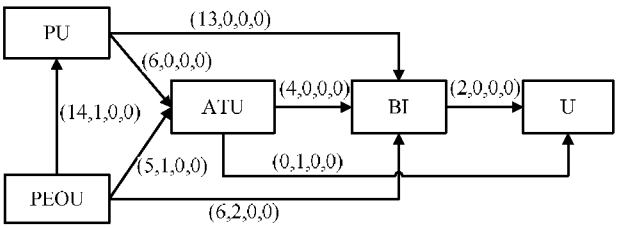


Figure 2: Meta-analysis of e-learning acceptance literature.

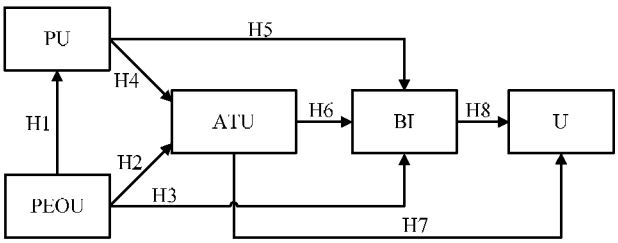


Figure 3: The research model.

Figure 2 is a summary of the literature review where links between TAM constructs are described with four values (x1, x2, x3, x4) indicating: (x1) number of positive relations, (x2) number of non-significant positive relations, (x3) number of negative relations, and (x4) number of non-significant negative relations.

3 Research model and hypotheses

The main reasons why TAM was used as a ground theory in this study are: (1) TAM is focused on information technology, (2) TAM has been used by different researchers, (3) TAM is a simple and generic model that can be used to study initial and continued intention, and (4) so far, TAM has not been used in the context of Moodle.

The main purpose of this study is to empirically validate the model TAM in the context of Moodle, therefore the research model (see Figure 3) was adapted from TAM. In the following sub-sections, the variables, their relationships and consequent causal links are hypothesised.

3.1 E-learning system ease of use

Ease of use refers to the effort required by the user to take advantage of the application. PEOU can have impact on a user's belief about PU when using a system [2]. In case of e-learning, a positive link between PEOU and PU was found in several studies [11-13],[9],[7],[14-16],[8],[6],[17],[18]. However, Ya-Ching Lee [17] showed that this link is not significant when an e-learning system is being used voluntarily. We propose the following hypothesis:

H1: PEOU will have a positive effect on PU.

In existing literature, PEOU was shown to be a positive determinant on ATU [9],[16],[8],[6]. We therefore propose the following hypotheses:

H2: PEOU will have a positive effect on student's attitudes towards using Moodle.

The use of a system is theorised to be influenced by PEOU [2]. BI can be influenced by PEOU as has been demonstrated by different authors [11-15],[17]. In case of the voluntary use of an e-learning system these results can differ, as has been shown by [18] and [17]. We propose the following hypothesis:

H3: PEOU will have a positive effect on Moodle usage intention.

3.2 E-learning system usefulness

Among the many variables that may influence system use, PU has been demonstrated as the main precondition for BI and system acceptance. According to the results of the meta-analysis, PU is the most important predictor of BI. The positive effect between PU and BI was found in different studies [11-13],[9],[19],[14],[15],[6],[17],[18]. It was also shown that PU has a positive effect on ATU [9],[16],[8],[6]. We therefore suggest the following hypotheses:

H4: PU will have a positive effect on student's attitudes towards using Moodle.

H5: PU will have a positive effect on student's intention to use Moodle.

3.3 Attitude toward using an e-learning system

An attitude is "a summary evaluation of a psychological object captured in such attribute dimensions as good-bad, harmful-beneficial, pleasant-unpleasant, and likable-dislikable" [20]. A learner's BI can be caused by their feelings about the system. If the learners don not like an e-learning system or if they feel unpleasant when using it, they will probably want to replace the system with a new one. Liu et al. [9] and Matthew K.O. Lee et al. [6] have demonstrated that ATU is a direct determinant of BI. We propose the following hypotheses:

H6: ATU will have a positive effect on a student's intention to use Moodle.

H7: ATU will have a positive effect on a student's actual use of Moodle.

3.4 Behavioural intentions for using an e-learning system

BI is an indication of an individual's readiness to perform a given behaviour. It is assumed to be an immediate antecedent of behaviour. Existing studies [11],[14] have demonstrated that BI can be a determinant for the actual use of an e-learning system. Thus, we propose the following hypothesis:

H8: Students' BI will have a positive effect on his or her actual use of Moodle.

4 Research methodology

Quantitative research in the form of an online questionnaire based survey was performed to test the stated hypotheses. In this section, the development of the measurement instrument, the sampling process and data analysis approach are described.

Table 2: Profile of the respondents.

Demographic characteristics		Frequency	Percentage
Gender	Male	188	80.0
	Female	47	20.0
Age	18 – 20 years	45	19.1
	21 – 22 years	124	52.8
	23 – 24 years	48	20.4
	25 – 26 years	15	6.4
	more than 26 years	3	1.3
Internet experience	No experience	0	0.0
	Some experience	2	.9
	Experienced	117	49.8
	Very experienced	116	49.4
Moodle experience	No experience	1	0.4
	Some experience	33	14.0
	Experienced	164	69.8
	Very experienced	37	15.7
Number of courses where Moodle is used (for the present academic year)	1	4	1.7
	2	28	11.9
	3-5	127	54.0
	5-8	61	26.0
	8-13	16	6.8
	13-21	3	1.3
Frequency of Moodle use	A couple times a year	2	0.9
	A couple times a month	4	1.7
	Weekly	81	34.5
	Daily	148	63.0

4.1 Instrument development

Empirical data were collected by means of an online questionnaire containing 22 questions. The questions were organized into the following two groups: (1) demographic questions about the respondents' gender, age, years of study, internet experience, Moodle experience, voluntariness, etc. (see Table 1 for the characteristics list); and (2) measures for the TAM constructs. The TAM constructs were adapted into the context of Moodle (see in Appendix A). The TAM measuring items were Likert-like items on a 7-point scale from ''strongly agree" to ''strongly disagree". To reduce measurement error, the development of the online questionnaire involved the following steps. First, a pre-test of the questionnaire was performed. The main goal of the pre-test was to improve the content of the measuring items, therefore colleagues from the faculty were asked to examine the questionnaire for meaningfulness, relevance and clarity. According to the feedback, few measurement items were refined in wording. After the pre-test, a pilot test of the questionnaire was performed with a non-random sample of twenty-five volunteers constituting faculty staff and students. The main goal of the pilot test was to empirically validate the reliability of the questionnaire – to check whether the measurement instrument lacked accuracy or precision. Data collected from the pilot test was analysed using SPSS to conduct internal consistency of the measurement items. The statistical test results confirmed a solid reliability for all measurement items.

4.2 Sampling process

Moodle is an open-source system and can be downloaded, deployed and used for free. Therefore, it is hard to identify the exact number of Moodle users. The number increases by approximately 1,300 new registered users every month. The statistical report from April 2010 [21] indicates that at the time of this report, 44,171 Moodle sites from 207 countries had been registered and validated. So far, over three million online courses using Moodle have been established and there are more than 31 million registered Moodle users. Our sample frame was limited to students that use Moodle at the Faculty of Electrical Engineering and Computer Science in Maribor. Our sample frame covered full-time students of technical studies. At the time of our research, 115 online courses were established and 1,566 users were registered. A systematic random sampling process, where every member of the sample frame had an equal chance of being selected, produced a sample of 800 Moodle users. The students that participated in the pilot test were excluded from the sample frame in the random sampling process. A request form for participation in the online survey was sent to the selected students. 284 online surveys were started, of which 235 were successfully finished and 49 returned incomplete. The usable response rate was thus 29%.

4.3 Statistical analysis

To describe the main features of an average participant in this study, descriptive statistics was used on the respondents' characteristics data. Structural equation modelling (SEM) was used to test the fit of the proposed theoretical model (Fig. 3) with empirical data. The measurement model was estimated using confirmatory factor analysis to test whether the proposed constructs possessed sufficient validation and reliability. To assess the reliability and validity of the measurement instrument used in this study, internal consistency, composite reliability and convergent validity were demonstrated. After assessing the reliability and validity of the measurement instrument, the measurement model was estimated. After the final measurement model passed the goodness-of-fit tests, the structural part of the research model was estimated using SEM on the structural model. The structural model was also tested for a data fit with appropriate goodness-of-fit indices. A statistical analysis was performed using the SPSS statistical package together with AMOS 17.0 software.

5 Data analysis and results

In this section, the data analysis and results are given. In the first subsection, the profile of the respondents is presented using descriptive statistics. In the following subsection, the methods for measurement instrument validity and reliability assessment are explained. Finally, the measurement and structural model analysis are used to explain the results of the study.

5.1 Demographic characteristics

The characteristics of the respondents are presented in Table 2. The typical respondent is a 21-22 year old male with less than a year of 1-2 years of study. The respondent has solid internet experience and already has had experience with Moodle. 63% of the respondents use Moodle daily and more than a half use Moodle for 3-5 courses.

5.2 Measure reliability and validity

Before testing the hypotheses, measurement items in the questionnaire were first assessed for content and construct reliability and validity. The results of the tests for unidimensionality, reliability and convergent validity provided evidence of the internal and external validity of the measurement instrument and scales. Table 3 summarizes the results of internal reliability, composite reliability and convergent validity for measurement instrument constructs. The internal consistency of the constructs was assessed by Cronbach's α, which is used for estimating the extent to which multiple indicators for a latent variable belong together. All the estimated Cronbach's α values for TAM constructs exceeded the cut-off value of 0.70 [22] , thus the constructs showed a reasonable level of reliability. Composite reliability was estimated using the following equation:

$$Cr = \frac{(\sum factor\ loading)^2}{(\sum factor\ loading)^2 + \sum measurement\ error}$$

The composite reliability measures for all of the constructs exceeded the recommended level of 0.70 [23]. As the third indicator of convergent validity, average variance extracted (AVE) was estimated. If the AVE is less than 0.5, then the variance due to measurement error is greater than the variance captured by the respective construct [23]. AVE was estimated using the following equation:

$$AVE = \frac{\sum(factor\ loading)^2}{\sum(factor\ loading)^2 + \sum measurement\ error}$$

Table 3: Instrument reliability and validity.

Construct	Item	Factor loading	Internal consistency Cronbach α >= 0.70	Composite factor reliability >= 0.70	Convergent validity Average Variance Extracted >= 0.50
Perceived Usefulness	PU2	0.77	0.815	0.821	0.605
	PU3	0.84			
	PU4	0.72			
Perceived Ease of Use	PEOU1	0.82	0.899	0.902	0.755
	PEOU2	0.89			
	PEOU3	0.90			
Behavioural Intention	BI1	0.77	0.848	0.865	0.682
	BI2	0.89			
	BI3	0.81			
Attitude Toward Using	ATU2	0.77	0.848	0.850	0.654
	ATU3	0.85			
	ATU4	0.80			

5.3 The measurement model

According to the modification indices provided by AMOS, some indicators (PEOU4, PU1 and ATU1) have been cut off from the initial measurement model and then the overall fit model for the final measurement model was estimated to ensure a good data fit with the model. A variety of fit indices were assessed to identify model goodness-of-fit as proposed by Rainer and Miller [24]. These indices include $\chi 2$, the goodness-of-fit index (GFI), the adjusted goodness-of-fit index (AGFI), the comparative fit index (CFI), the root mean squared residual (RMSR), the root mean square error of approximation (RMSEA), the normed fit index (NFI), the non-normed fit index or Tucker Lewis index (NNFI) and the parsimonious fit index (PNFI).

Table 4 provides a summary of estimated fit indices for the final measurement model.

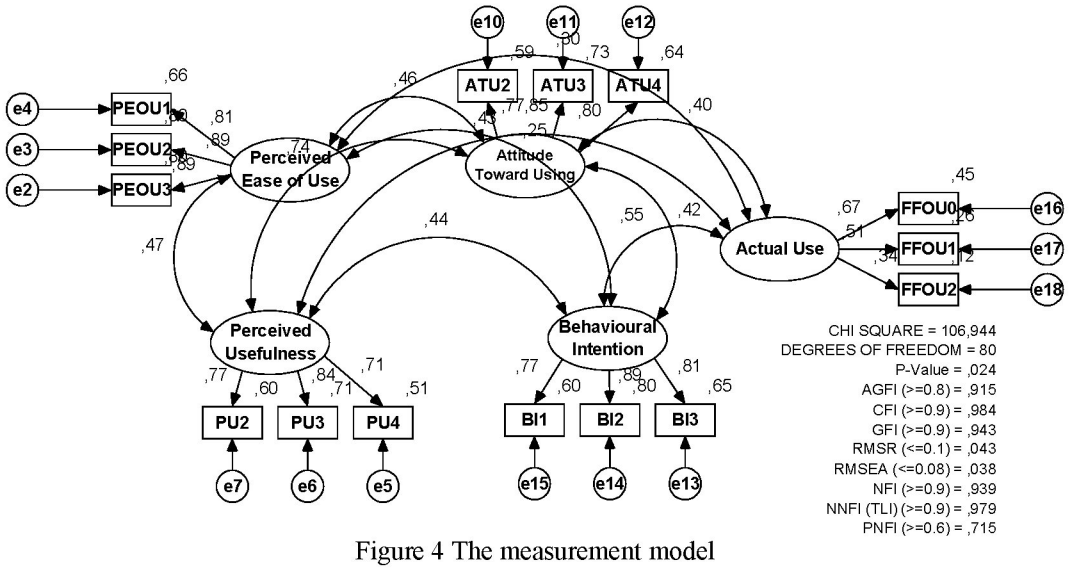


Figure 4 The measurement model

Table 4: Model fit summary for the final measurement and structural model.

Fit index	Recommended value [25]	Measurement model	Structural Model
χ^2	Non-significant	106.944	109.309
Degrees of freedom (df)	n/a	80	82
p		0.024	0.024
χ^2/df	< 3.00	1.337	1.318
Goodness-of-fit index (GFI)	> 0.90	0.943	0.942
Adjusted Goodness-of-fit index (AGFI)	> 0.80	0.915	0.915
Comparative fit index (CFI)	> 0.90	0.984	0.983
Root mean square residuals (RMSR)	< 0.10	0.043	0.044
Root mean square error of approximation (RMSEA)	< 0.08	0.038	0.038
Normed fit index (NFI)	> 0.80	0.939	0.938
Non-normed fit index (NNFI)	> 0.90	0.979	0.979
Parsimony normed fit index (PNFI)	> 0.60	0.715	0.732

5.4 The structural model

The estimated values of fit indices have proven the good structural model fit to the data. The values of fit indices are presented in

Table 4: The results of the final structural model (see

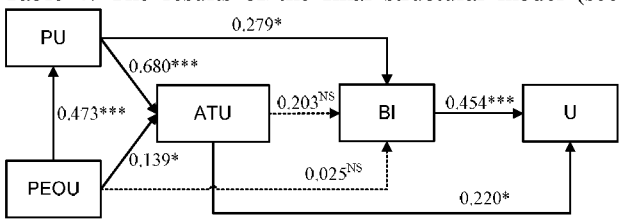


Figure 6) provide support for H1, meaning that PEOU (β=0.473; p<0.001) positively influences the PU. The final structural model results also show that PEOU (β=0.139; p<0.05) and PU (β=0.680; p<0.001) positively affect attitudes toward using Moodle. These results provide support for hypotheses H2 and H4. Students' behavioural intentions using Moodle are also positively affected by perceived usefulness (β=0.279; p<0.05), thus the hypothesis H5 was supported. Actual use of Moodle is positively affected both by attitudes toward using Moodle (β=0.220; p<0.05) and students' behavioural intentions (β=0.454; p<0.001), meaning that hypotheses H7 and H8 were supported. However, there was statistically insufficient evidence regarding the impact of PEOU and ATU on BI. This means, that the results did not provide support for hypotheses H3 and H6. Table 5 summarizes the hypothesis testing results and the results of the multiple-group analysis.

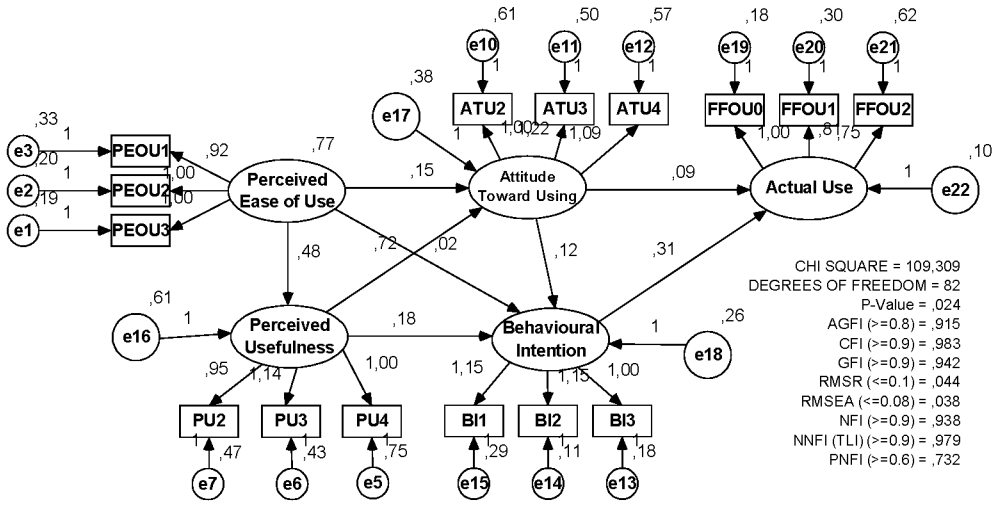


Figure 5: The final structural model.

Table 5: Hypothesis testing results.

Hypothesis	Effects	Path coefficient	Remarks
H1	PEOU → PU	0.473***	Supported
H2	PEOU → ATU	0.139*	Supported
H3	PEOU → BI	0.025NS	Not Supported

H4	PU → ATU	0.680***	Supported
H5	PU → BI	0.279*	Supported
H6	ATU → BI	0.203^{NS}	Not Supported
H7	ATU → U	0.220*	Supported
H8	BI → U	0.454***	Supported

Notes: * p< 0.05; ** p< 0.01; *** p < 0.001; ^{NS} p > 0.05

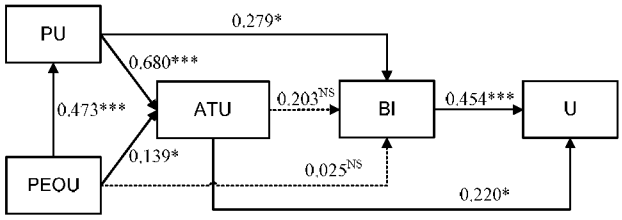


Figure 6: The research model results.

The final hypothesis results are also presented in Figure 6, where the values of size and the significance of individual causal links are written above the arrows between TAM constructs. A dotted arrow between two constructs means that there was no significant relationship found between these two constructs.

Table 6 summarizes the results of the study, where the discovered relationships are added to existing knowledge in the field of e-learning acceptance. The questionnaire and the items were confirmed with adequate discriminant and convergent validity metrics. For the measurement and structural model, several data fit indices were estimated in order to test the fit of the data with the proposed research model.

Table 6: The contribution of the study to existing knowledge.

Causal Relationship					
Independent Variable	**Dependent Variable**	**Positive**	**Positive ^{NS}**	**Negative**	**Negative ^{NS}**
PEOU	PU	14*(+1)*	1	0	0
PU	BI	13*(+1)*	0	0	0
PEOU	BI	6	2*(+1)*	0	0
PU	ATU	6*(+1)*	0	0	0
PEOU	ATU	5*(+1)*	1	0	0
ATU	BI	4	0*(+1)*	0	0
BI	U	2*(+1)*	0	0	0
ATU	U	0*(+1)*	1	0	0

6 Conclusion

The present study resulted in the empirical validation of the TAM research model in the context of Moodle and therefore contributes to the body of research in the field of e-learning acceptance based on the state-of-the-art theory: TAM.

The results of the study revealed that the perceived usefulness and perceived ease of use are factors that directly affect students' attitudes toward using Moodle, whereas perceived usefulness is the strongest and most significant determinant of students' attitudes toward using Moodle. This means that students like to use Moodle if they have good feelings about the usefulness of Moodle in getting better grades and knowledge. Several existing studies have also revealed that perceived usefulness can play an important role in affecting students' attitudes towards using an e-learning system [6],[9],[8],[26]. Students' perceptions regarding the "likeness" of using the system are also reflected by their comprehensions about how easy it is to use the system. The same results were also demonstrated in several other studies [9],[8],[26]. Perceived ease of use has a strong and significant impact on perceived usefulness. Students' intentions for using Moodle is not a result of students' perceptions about how much they like to use it. The SEM analysis also did not show a direct causal link between perceived ease of use and students' intention of using Moodle. The students' intention of using Moodle is mainly prompted by its perceived usefulness, meaning that students will use the e-learning system if they find it useful in their learning process. According to the results, the actual use of Moodle is a result of two factors: attitudes toward using and behavioural intention, where the latter is the most significant and strongest predictor of actual use of Moodle.

The results of this study have implications that are important to different e-learning stakeholders. As was discovered in this research, in the majority of cases, students are experienced internet users that are not worried about dealing with new technologies. Students mostly like to use an e-learning system because they find it useful for their studies, meaning that the e-learning system has to provide all the necessary e-learning services that a modern student needs in his or her learning process. E-learning system developers have to keep up with new web technologies and properly build them into the e-learning system. The usefulness of the e-

learning system is also closely connected to the content of e-learning materials that students are downloading from it. Learning content providers have to take advantage of an e-learning system to make the best of it when providing students with learning materials, news, asynchronous and synchronous communication, etc. We believe that students would find the e-learning system more useful if they would get adequate learning materials. The findings, presented in this study, can also be a direction for researchers in their future work. The research model should be extended in order to find external variables to investigate which factors have a significant influence on students' perceptions regarding ease of use and the usefulness of the e-learning system.

As in all empirical research, this study has limitations that need to be identified and discussed. First, the sample is limited to students at a faculty that is more or less technically oriented. Although the results from this study are useful for describing the characteristics of a large population of students, the generalizations of the results are limited to full-time undergraduate students. The students that participated in this study are mostly obliged to use Moodle in their studies. An average student is male and already possesses technical skills when it comes to internet use. Next, this study is only limited to a particular e-learning system. Although Moodle is a modern and well accepted e-learning system, the generalization of the results is limited to the characteristics and features provided by it. Moodle is an open-source product and therefore extensions can be implemented. The actual implementation and deployment of Moodle can affect different students' perceptions, such as usefulness and easiness. Because Moodle deployments' primary objectives are not the same in every case, this is another variable that will have to be addressed in future work as well.

In our future work, we will try to examine new variables that could be used to extend the TAM model for the e-learning domain. Together with future internet developments, new technologies and services will enable the creation of new and innovative e-learning system extensions and modules. We believe there are many constructs related to the user, technology and service domain characteristics. Such constructs can have a direct or indirect (but significant) impact on users' attitudes and intention for using the system. Our future research will therefore be dedicated to finding and evaluating such potential constructs.

7 References

[1] J. Massy (2005). "The eLearning industry and market in Europe". Available at http://ec.europa.eu/education/archive/elearning/doc/studies/market_annex1_en.pdf. [Accessed April 15, 2010].

[2] F.D. Davis (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, vol. 13, pp. 319-340.

[3] F.D. Davis, R.P. Bagozzi, and P.R. Warshaw (1989). User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. *Management Science*, vol. 35, pp. 982-1003.

[4] W. King and J. He (2006). A meta-analysis of the technology acceptance model. *Information & Management*, vol. 43, pp. 740-755.

[5] J.H. Sharp (2006). Development, Extension and Application: A Review of the Technology Acceptance Model. *Proc ISECON*, vol. 23.

[6] M.K. Lee, C.M. Cheung, and Z. Chen (2005). Acceptance of Internet-based learning medium: the role of extrinsic and intrinsic motivation. *Information & Management*, vol. 42, pp. 1095-1104.

[7] E.M. van Raaij and J.J. Schepers (2008). The acceptance and use of a virtual learning environment in China. *Computers & Education*, vol. 50, pp. 838-852.

[8] E. Ngai, J. Poon, and Y. Chan (2007). Empirical examination of the adoption of WebCT using TAM. *Computers & Education*, vol. 48, pp. 250-267.

[9] S. Liu, H. Liao, and J.A. Pratt (2009). Impact of media richness and flow on e-learning technology acceptance. *Computers & Education*, vol. 52, pp. 599-607.

[10] V. Venkatesh and F.D. Davis (2000). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science*, vol. 46, pp. 186-204.

[11] S. Zhang, J. Zhao, and W. Tan (2008). Extending TAM for Online Learning Systems: An Intrinsic Motivation Perspective. *Tsinghua Science & Technology*, vol. 13, pp. 312-317.

[12] C. Ong, J. Lai, and Y. Wang (2004). Factors affecting engineers' acceptance of asynchronous e-learning systems in high-tech companies. *Information & Management*, vol. 41, pp. 795-804.

[13] C. Ong and J. Lai (2006). Gender differences in perceptions and relationships among dominants of e-learning acceptance. *Computers in Human Behavior*, vol. 22, pp. 816-829.

[14] C. Yi-Cheng, C. Chun-Yu, L. Yi-Chen, and Y. Ron-Chen (2007). Predicting College Student' Use of E-Learning Systems: an Attempt to Extend Technology Acceptance Model. Available at: http://www.pacis-net.org/file/2007/1295.pdf [Accessed April 10, 2010].

[15] P.D. Chatzoglou, L. Sarigiannidis, E. Vraimaki, and A. Diamantidis (2009). Investigating Greek employees' intention to use web-based training. *Computers & Education*, vol. 53, pp. 877-889.

[16] M. Lee (2010). Explaining and predicting users' continuance intention toward e-learning: An extension of the expectation-confirmation model. *Computers & Education*, vol. 54, pp. 506-516.

[17] Y. Lee (2006). An empirical investigation into factors influencing the adoption of an e-learning system. *Online Information Review*, vol. 30, pp.

517 - 541.
[18] J. Shen and L.B. Eder (2009). Intentions to Use Virtual Worlds for Education. *Journal of Information Systems Education*, vol. 20, pp. 225-233.
[19] S. Liaw, H. Huang, and G. Chen (2007). "Surveying instructor and learner attitudes toward e-learning. *Computers & Education*, vol. 49, pp. 1066-1080.
[20] I. Ajzen (2001). Nature and operation of attitudes. *Annual Review of Psychology*, vol. 52, pp. 27-58.
[21] Moodle, Moodle.org: open-source community-based tools for learning. *Moodle.org*, 2010. Available at: http://www.moodle.org [Accessed April 15, 2010].
[22] N. Schmitt (1996). Uses and abuses of coefficient alpha. *Psychological Assessment*, vol. 8, pp. 350-353.
[23] A. Segars (1997). Assessing the unidimensionality of measurement: a paradigm and illustration within the context of information systems research. *Omega*, vol. 25, pp. 107-121.
[24] R.K. Rainer and M.D. Miller (1996). An assessment of the psychometric properties of the computer attitude scale. *Computers in Human Behavior*, vol. 12, Spring. pp. 93-105.
[25] S. Hong, J.Y.L. Thong, and K.Y. Tam (2006). Understanding continued information technology usage behavior: a comparison of three models in the context of mobile internet. *Decis. Support Syst.*, vol. 42, pp. 1819-1834.
[26] T. Teo (2009). Modelling technology acceptance in education: A study of pre-service teachers *Computers & Education*, vol. 52, pp. 302-312.

8 Appendix A. TAM items and scales

Item	Question	I strongly agree	I agree	I somewhat agree	Can't decide	I somewhat disagree	I disagree	I strongly disagree
Perceived Usefulness (PU)								
PU1	I would find Moodle useful for learning.	☐	☐	☐	☐	☐	☐	☐
PU2	Using Moodle enables me to accomplish tasks more quickly.	☐	☐	☐	☐	☐	☐	☐
PU3	Using Moodle for learning increases my productivity.	☐	☐	☐	☐	☐	☐	☐
PU4	If I use Moodle, I will increase my chances of getting knowledge.	☐	☐	☐	☐	☐	☐	☐
Perceived Ease Of Use (PEOU)								
PEU1	My interaction with Moodle would be clear and understandable.	☐	☐	☐	☐	☐	☐	☐
PEU2	It would be easy for me to become skilful at using the system.	☐	☐	☐	☐	☐	☐	☐
PEU3	I would find Moodle easy to use.	☐	☐	☐	☐	☐	☐	☐
PEU4	Learning to operate Moodle is easy for me.	☐	☐	☐	☐	☐	☐	☐
Attitude Toward Using Technology (ATU)								
ATU1	Using Moodle is a bad idea (negative).	☐	☐	☐	☐	☐	☐	☐
ATU2	Moodle makes learning more interesting.	☐	☐	☐	☐	☐	☐	☐
ATU3	Working with Moodle is fun.	☐	☐	☐	☐	☐	☐	☐
ATU4	I like working with Moodle.	☐	☐	☐	☐	☐	☐	☐
Behavioural intention (BI)								
BI1	I intend to use Moodle in the next 6 months.	☐	☐	☐	☐	☐	☐	☐
BI2	I predict I would use Moodle in the next 6 months.	☐	☐	☐	☐	☐	☐	☐
BI3	I plan to use Moodle in the next semester.	☐	☐	☐	☐	☐	☐	☐

Resource Control and Estimation Based Fair Allocation (EBFA) in Heterogeneous Active Networks

K. Vimala Devi
Department of Computer Science and Engineering
Anna University, Trichirappalli,
Trichirappalli, 620024, India.
E-mail: k_vimadevi@yahoo.co.in

C. Thangaraj
Kalasalingam University,
Krishnankoil – 626190, India.
E-mail: thangaraj@akce.ac.in

K.M. Mehata
Anna University,
Chennai - 600025, India
E-mail: mehata@annauniv.edu

Keywords: active network management, resource control, fair allocation, resource estimation, estimation based allocation

Received: July 11, 2009

Active networks perform customized computation on the messages flowing through them. Individual packets carry executable code, or references to executable code. Active networks are changing considerably the scenery of computer networks and consequently, affect the way network management is conducted. In a heterogeneous networking environment, each node must understand the varying resource demands associated with specific network traffic. This paper describes and evaluates an approach to control the CPU utilization of malicious packets and to estimate the CPU demand for good packets in a heterogeneous active network environment. We also describe a new approximation for estimation based fair allocation. The proposed algorithm called Estimation Based Fair Allocation Algorithm (EBFAA) avoids the ill-behaved flows to utilize more CPU time and achieves perfect fairness for all flows during allocation.

Povzetek: Prispevek opisuje obravnavo zlonamernih paketov v aktivnih heterogenih mrežah.

1 Introduction

In classical packet-switched communication networks, when a packet transits through an intermediate node along the path from source to destination, each intermediate node has a measured rating for per-message and per-byte throughput. Thus a linear extrapolation from packet size and arrival rate should provide the node a reasonable estimate for the CPU demand associated with individual packets or with sets of packets. Unfortunately, this simple approach cannot work for active networks because individual packets can require substantially different processing.

In active networks [15], when a packet arrives at an intermediate node, the data may include program code that can be accessed, interpreted, and executed by the node. The code may specify a compression algorithm to be applied on the data if congestion has been detected in the area of the node, or may specify which packets to drop first, or may modify the destination address to route around congestion. Thus, in active networks, some more sophisticated technique is needed to estimate CPU demand associated with active packets.

1.1 Active network architecture

Active networks [15] allow individual user, or groups of users, to inject customized programs into the nodes of the network. "Active" architectures enable a massive increase in the complexity and customization of the computation that is performed within the network

- Node operating system (node os)

A NodeOS [7,9] is a special-purpose operating system that runs on the routers of an active network and supports active network execution environments (A router in an active network is called an **active node**, and hence the name NodeOS). In order to prevent active applications

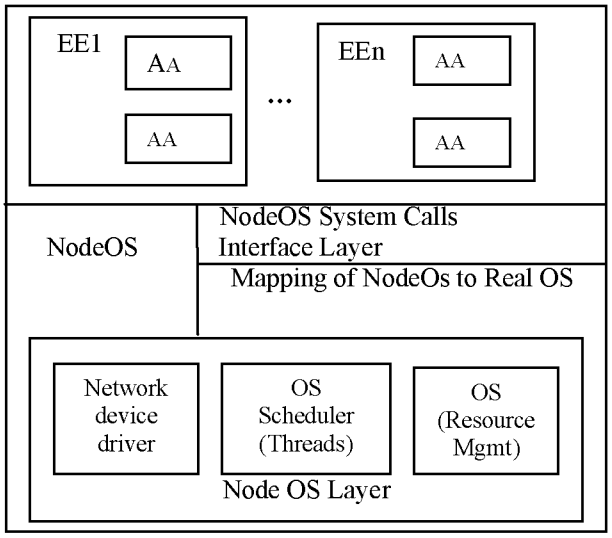


Figure 1: Active-network architecture [3, 5].

from misbehaving, active network execution environments enforce fine-grained control over the resources consumed by active applications. For example, an execution environment may restrict the number of CPU cycles an active application can consume, or it may enforce a limit on the number and type of packets an active application can receive and send. The interface/API provided by traditional operating systems is inadequate for such needs of execution environments. For example, in traditional Unix, where all resources are associated with a **process**, it is very difficult to enforce an absolute limit on resources consumed by an active application if it is not a process, and active networks would be very slow if each active application is run in a separate Unix process. A NodeOS provides the exact interface needed by active network execution environments. A NodeOS is also different from a traditional OS in terms of the overhead it imposes to do its job. Further, a NodeOS should be capable of handling as many network packets per second as possible. Therefore, the NodeOS should impose minimum overhead to perform operating system functions. The above requirements raise interesting operating system design issues, primarily in the areas of API design and effificent resource control.

The Node operating System (NodeOS) provides the basic functions from which Execution Environments build the abstractions that make up the network APIs. The NodeOS isolates EEs from details of resource management and the existence of other EEs. The EEs in turn, hide most of the details of their interaction with the end user from the NodeOS. The NodeOS defines four primary abstractions: threads pools, memory pools, channels and flows. The first three encapsulate a system's three types of resources: computation, storage and communication. The fourth is used to aggregate control and scheduling of the other three in a form that more closely resembles network application programs[11].

Examples of NodeOS: -Scout and Amp. A number of other NodeOS implementations, such as xbind and EROS, are also under development and testing.

- ***Execution environment (ee)***

Active networks rely on the ability to add programs easily to the network infrastructure, so the choice of the Execution Environment's runtime environment and programming language is critical. Below are some of the Execution Environments for setting up of Active Networks.

1. ***Ants: Active Node Transfer System***

The Massachusetts Institute of Technology's ANTS aims at standardizing on a communication model rather than individual communication protocols, such as IP, UDP etc. The major design goal is to build a system that allows rapid transfer and deployment of protocol code across the network. ANTS uses Java as its programming language, and the Java Virtual Machine as its runtime environment. Java's features make ANTS suitable for a variety of applications [19].

2. ***Magician***

Magician [1], a toolkit for creating a prototype Active Network was developed at the University of Kansas. In an Active Network, program code and data is placed inside specialized packets called SmartPackets. The nodes of an Active Network are called active nodes and they are programmable in the sense that when a SmartPacket reaches an active node, the code inside the SmartPacket is extracted and executed. Depending on the nature of the code inside the SmartPacket, the SmartPacket either modifies the behavior of the active node or transforms the data it is carrying. The basic implementation uses UDP/IP combination for transport and routing.

1.2 Current network management and its limitations

Currently, networks are monitored and controlled mainly through SNMP commands that read or set variables in the MIBs of the elements. Current MIB implementations, which defined by their manufacturers, have several significant limitations.

1. A well-known limitation of SNMP is related to its inability to handle high volumes of processed network data.
2. Another limitation of the current management techniques is that all management decisions are usually made centrally. This approach is inefficient when the network is congested, or when a part of it is inefficient when the network is congested, or when a part of it is unreachable, since the management commands may arrive late or get lost. Active nodes can be programmed to make such decisions, thus allowing the distribution of the decision centers across the network [8, 12].

1.3 Suitability of active networks for network management

The use of Active Networks technologies to network management has the following advantages:

- The information returned can be controlled and managed according to needs.
- The management rules can be shifted from the management centers to the active nodes.
- The monitoring and control loop is shortened.

Active networks for the functional areas of network management

The functional areas of Network Management are Fault Management, Configuration Management, Accounting Management, Performance Management and Security Management (FCAPS).

- ***Fault management***

In fault management as well as other areas of Network Management such as configuration and performance management, predicting and preventing undesirable situations is important. Current predictive algorithms take into consideration only a few parameters. However active network technologies enable deployment of efficient predictive management, since the computations can be distributed to the whole network. Each node predicts and transmits to its neighbors its future state and also the prediction of each node depends on its current state and the predictions of its neighbors. Congestion can also be predicted with satisfactory accuracy [12].

- ***Configuration management***

Configuration management techniques may be enhanced in an AN environment. For instance, MAs can be used for inventory management. Those MAs can be used to discover and report changes to the existing configuration. For example agents could be programmed to propagate DNS updates to the entire network.

AN can also facilitate VPN deployment. VPNs are independent private networks built over a shared public network. Practically this means that network resources are partitioned and allocated (dynamically or statically) to each group. In AN access to the resources of active nodes can be controlled; hence partitioning of resources can easily be implemented. An attempt in this direction is Virtual Active Network (VAN) architecture [12].

- **Accounting management**

One of the important tasks accounting management tools carry out is monitoring network usage. Most AN architectures, for security and safety reasons, authenticate the users before any resources are allocated to them to access any service. Thus the monitoring of the resources is integrated to the network architecture, rather than being an additional function. With AN, all resource usage, such as bandwidth, CPU, memory, or scheduling priorities, can be accounted.

Finally, AN may be manageable even when some areas cannot be reached by the management stations. This is crucial for accounting management, because those situations usually lead to unreported network use, and therefore loss of profit. Such situations can be prevented in active environments [12].

- **Performance management**

With AN, the way devices handle traffic can easily be customized on a per-device and per-user basis. Hence scheduling and routing, traffic shaping, admission control, and priorities can easily be controlled in order to manipulate traffic. The deployment of QoS services can easily be achieved in AN, since protocols that perform the necessary reservations and computation can be installed on active nodes. AN is also flexible in installing protocols. Complex QoS protocols, such as Resource Reservation Protocol (RSVP) or qGSMP, could be deployed easily too: the reservation of resources and scheduling algorithms of active nodes can be manipulated in any desired way. The ability to implement QoS protocols without relying on legacy and rigid protocols (e.g., IP) makes those protocols lightweight and efficient [12, 19].

- **Security management**

The AN architectures implement modules that relate to security and safety. These modules authenticate access to resources hence; several of the current security management tasks are architecturally integrated in to these modules. This relieves NM tools from the enforcement of policies and SLAs. Apart form traditional policing; intrusion detection can become much easier and effective by agents that reside on sensitive nodes. Attacks such as TCP SYN attack can also be effectively detected and prevented. For instance Phonix framework allows the existence of MAs that are programmed to perform specific tasks, such as safeguarding the network. [12]

1.4 The resource demand in active networks

Performance management aims to keep network performance within predefined levels. It is strongly related to resource management and QoS provisioning [12, 17] and to the parameter, resource utilization. Performance management tools measure various parameters, such as network throughput, delays, and CPU and bandwidth utilization, and attempt to control them. To use the Active Network technology safely and efficiently, individual nodes must understand the varying resource demands associated with specific network traffic.

Three types of resources in active networks: Computation, Storage and Communication (Network)

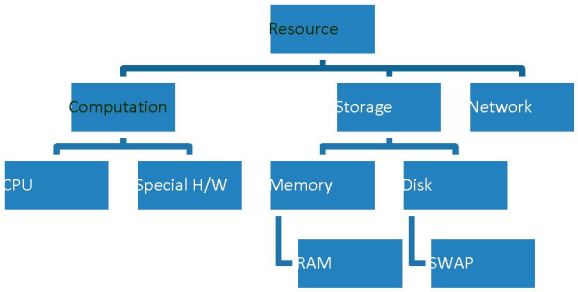


Figure 2: Resource management (QoS- parameters).

Inability to estimate the CPU demands of active packets can lead to some significant problems. First, a maliciously or erroneously programmed active packet might consume excessive CPU time at a node, causing the node to deny services to valid active packets. Alternatively, a node might terminate a valid active packet prematurely, wasting the CPU time used prior to termination, and ultimately denying service to a correctly programmed application. Second, an active node may be unable to schedule CPU resources to meet the performance requirements of packets. Third, an active packet may be unable to discover a path that can meet its performance requirements. Devising a method for active packets to specify their CPU demands and fair resource allocation can help to resolve these problems, and can open up some new areas of research. Unfortunately, there exists no well-accepted metric for expressing CPU demands in a platform independent form. This is the problem that motivated our research.

The paper is organized as follows: In section 2, the existing solutions to the problem are presented. Section 3 discusses the problem with the applied models. The implementation of a heterogeneous active network setup using Magician, a tool for active networks is discussed in section 4 and the evaluation and comparison of the results with various active applications is also presented. The resource estimation methods are presented in section 5. Section 6 proposes the Estimation Based Fair Allocation algorithm (EBFAA) and evaluates the performance of the algorithm. Section 7 draws conclusions on the effectiveness of our solution and suggests some possible future work.

2 A survey of existing approaches

While the outlines of our solution appear complex, we believe that success along these lines will enable more effective control of CPU usage by mobile programs and will enable node operating systems to more efficiently manage CPU resources. Others also see a need to provide such capabilities. In this section we present the existing solutions to prevent excessive CPU resource consumption in active networks and in mobile agent systems. Next we examine the research conducted outside of active networks that could help to provide effective resource management in active-network nodes.

2.1 Existing solutions to control the CPU usage

In order to prevent malicious or erroneous active packets from consuming excessive CPU time, most execution environments implement specific control mechanisms. In this section, we discuss the most common mechanisms.

Limit fixed by the packet

Some execution environments, such as ANTS [23], assign a timeto-live (TTL) to each active packet. An active node decreases this TTL as a packet transits the node, or whenever the node creates a new packet. In this way, each active packet can only consume resources on a limited number of nodes, but individual nodes receive no protection. The current TTL recommendation for the Internet protocol (IP) is 64 hops [13], which is supposed to roughly correspond to the maximum diameter of the Internet. This value might prove large enough for an active packet that propagates a configuration from node to node between two videoconferencing machines. But if the active packet creates numerous additional packets (to which it delegates a part of its own TTL), then the assigned TTL could prove insufficient. And it is usually difficult to predict how many new packets will be generated since these predictions might depend on network parameters, such as congestion and topology, which can rarely be known in advance. This TTL mechanism could contribute to protect individual nodes if the TTL is given in CPU time units instead of hop count. But the problem remains: how to choose the initial value for the TTL?

In the related context of mobile agents, Huber and Toutain [7] propose to enable packets that did not complete their "mission" to request additional credits. The decision to grant more credit would be taken by the originating node for its packets, or by the generating packet for packets created while moving among nodes. The decision must be made after examining a mission report included with the request for more credits. The proposed solution remains unimplemented, perhaps because the reports proved difficult to generate and evaluate.

Limit fixed by the node

In some execution environments (e.g., ANTS), a node limits the amount of CPU time any one packet can use. This solution protects the node but does not allow optimal management of resources. For instance, imagine that a node limits each packet to 10 CPU time units. Suppose that a packet requiring 11 CPU time units arrives when the node is not busy. In this case, the node will stop the execution of the packet just before it completes.

Use a restricted language

The SNAP language [10] is designed with limited expressiveness so that a SNAP program uses CPU in linear proportion to the packet's length. While this approach supports effective management of resource usage, it could prove too restrictive for expressing arbitrary processing in active applications. For instance,

only forward branches are allowed; as a result, if repetitive processing is required, the packet must be resent repeatedly in loop-back mode until the task is completed.

Market based approach

Yamamoto and Leduc [23] describe a model for trading resources inside an active-network node, based on the interaction between a "reactive user agents" included in the packet and resource manager agents that reside in the network nodes. The manager agents propose resources (such as link bandwidth, memory, or CPU cycles) to the user agents at a price that varies as a function of the demand for the resource (the higher the demand, the higher the price). Packets carry a budget that allows them to afford resources on active nodes. Based on the posted price of the resources and on its remaining credit, the user agent of a packet makes decisions about the processing to apply. For instance, if the CPU is in high demand and thus expensive to use, then a packet may decide to apply a simple compression algorithm to its data, instead of a more efficient but more costly algorithm, which the packet would have applied if the resource were more affordable. This approach, which might prove appropriate for mobile agent platforms, could increase the packet complexity too much to be used efficiently in active networks.

The two most common approaches to resource control in active networks apply a fixed limit on the CPU time allocated to an active packet. In one approach, each node applies its own limit to each packet, while in the other approach each packet carries its own limit, a limit that might prove insufficient on some nodes a packet encounters and overly generous on other nodes.

Neither approach provides a means to establish an appropriate limit for a variety of active packets, executing on a variety of nodes. Our research aims to solve this problem, while at the same time we intend to develop a solution that does not reduce the expressiveness of an active packet, nor make a packet too complex.

2.2 Existing attempts to quantify the CPU demand units

The survey of research related to quantify the CPU requirements initiates us to devise an effective solution. The following sections outline and discuss some of the ideas we found.

RISC cycles

The active-network architecture documents specify that a node is responsible to allocate and schedule its resources, and more particularly CPU time. Calvert [4] emphasizes the need to quantify the processing demands of an active application in a context where such demands can vary greatly from one node to another, and he suggests using RISC (Reduced Instruction Set Computer) cycles as a unit to express processing demands. He does not address two crucial questions. First, for a given active application, how can a programmer evaluate the number of RISC cycles required to execute a packet on a given node? Second, how can this number be converted into a meaningful unit for non-RISC machines?

Extra information provided by the programmer

In the AppLeS (application-level scheduling) project [3], the programmer provides information about the application that she wishes to execute on a distributed system. She must indicate for instance whether the application is more communication oriented or computation-oriented or balanced, the type of communication (e.g., multicast or point-to-point), and the number of floating-point operations (in millions) performed on each data structure. Using this information, a scheduling program produces a schedule expected to lead to the best performance for the application. This method can yield acceptable predictions only if the programmer is both willing and able to provide the required characteristics of the program. Discussions with software performance experts led us to think this is rarely the case.

Combined node-program characterization

Saavedra-Barrera and colleagues [14] attempted to predict the execution time of a given program on various computers. To describe a specific computer, they used a vector to indicate the CPU time needed to execute 102 well-defined FORTRAN operations. In addition, they provided a means to analyze a FORTRAN program, reducing it to the set of well-defined operations. The program execution time can then be predicted by combining the computer model with the program model. The approach yielded good results for predicting the CPU time needed to execute one specific run of a program on different computer nodes. These results encouraged us to model platforms separately from applications; however, we need to capture multiple execution paths through each application, rather than a single path. We are pursuing a separate thread of research, discussed under future work, which aims to apply insights from Saavedra-Barrera to the active-network environment.

Use acyclic path models

To measure, explain, or improve program performance, a common technique is to collect profile information summarizing how many times each instruction was executed during a run. Compact and inexpensive to collect, this information can be used to identify frequently executed code portions. Unfortunately, such profiles provide no detail on the dynamic behavior of the program (for instance, these techniques do not capture and report iterations). To solve this problem a detailed execution trace must be produced, listing all instructions as they are executed. But as program runs become longer, the trace becomes larger and more difficult to manipulate. Ball and Larus [2] propose an intermediate solution: to list only loop-free paths, along with their number of occurrences. Among other things, the authors demonstrate how the use of these acyclic paths can improve the performance of branch predictors. We might be able to exploit such algorithms to efficiently capture looping behaviors; however, to collect acyclic path

information we would need to instrument the program code for each application to be modeled. Given the variety of execution environments and active applications being devised by researchers, we decided to first evaluate some simpler approaches.

3 CPU control and demand prediction models

Any effective model of CPU demand by a mobile program, which we call an active-application model, seems likely to require delineating the processing paths through the program in terms of elements of a platform independent abstraction that the program will invoke on every node. We refer to such platform-independent abstractions as node models. In the context of active networks, two types of node model seem feasible: (1) white-box models and (2) black-box models. White-box models specify the functions offered to active applications by a specific execution environment. Black-box models specify the system calls offered to execution environments by a standard node operating system interface. While we are investigating both approaches, in this paper we focus mainly on a white-box model because, if successful, such models can be developed for each execution environment that a node intends to support. In addition to seeking techniques to improve black-box models, we have begun to investigate white-box models as an alternative approach. In our conception, white-box models represent the processing logic within an active application as it invokes services offered by an execution environment.

3.1 Proposed approach and significance

Now, we illustrate how our CPU demand models can be used in two sample applications. In one application, we decide when to terminate an active packet based on its consumption of CPU time. In a second application, we predict the CPU demand for nodes in an active network. In both applications, we compare results obtained using our white-box models (without considering the system calls) against results obtained using CPU control and estimation techniques typically available in execution environments. As active packets traverse a series of nodes along a path from source to destination, each active node will wish to enforce CPU usage limits on each packet. This permits a node to protect itself from malicious or erroneously programmed active packets.

While innovative and radical when considered for use inside networks, active-network execution environments share much in common with virtual machines used in Internet-based software architectures, and active applications appear quite similar to other forms of dynamically injected software, such as applets, scripts, servlets, and dynamically linked libraries. These similarities encourage us to believe that our model can be applied generally to the problem of specifying CPU demand in distributed applications that rely on the use of mobile code.

4 Implementation of white-box model

4.1 A heterogeneous active network setup

For the test setup, a three node heterogeneous active network is constructed: the machine *"AH-1"* is the sending node and *"AN-1"* is the destination. The following figure (Figure 3) represents this topology:

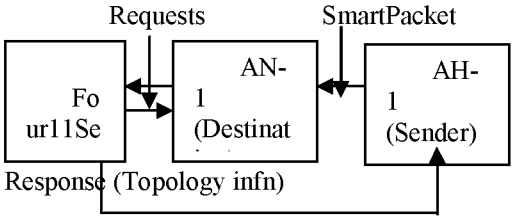


Figure 3: Test network setup.

The Smartpacket is transmitted from the Sender AH-1 to the Destination AN-1.

Network setup using MAGICIAN:
MAGICIAN was loaded in Linux environment. The tool provides an additional Execution Environment for setting up the Active Network and for sending the active packets using ANEP (Active Network Encapsulation Protocol). A Network Environment was created by giving specific host names to the machines forming the network.

One server (Four11) and two nodes have been setup:

Host name	**IP address**
magicserver	192.168.1.60 - Server
magicclient1	192.168.1.168 – (AN-1)
magicclient2	192.168.1.169 – (AH-1)

The tool was installed in all nodes. A network configuration (topology) file was created with the filename as similar to netname:magicserver

magicserver.conf consists of the following topology information:

```
(net:magicserver
    (node: AN-1
     host:magicserver
     IP:192.168.1.60
     gateway: (AN-2)
     ports: (3325 3322 3324)
    nbors; ((AN-2 192.168.1.168 3325 10000))

    node: AN-2
    host:magicclient1
    IP:192.168.1.168
    gateway: (AN-2)
    ports: (3323 3324)
    nbors: ((AN-1 192.168.1.60 3324 10000))
    ……..
```

A configuration file is created for Four11 server which is to be read by all the clients to know where the server is

running and to setup a connection from the client side to the server.

Four11.conf consists of:

host: magicserver port: 9411

Starting and testing the active network:

Starting the Four11 server:

The server has to be started from the directory where the tool has been stored

Java magician.Four11.Four11 magicserver

arguments:

magicserver - the netname-which in turn the name of the network config file-magicserver.conf

Starting the active node from the magicclient1 (host m/c):

The Node has also to be started from the directory where the tool has been stored in client machine.

Java magician.Node.NetworkNode magicclient1 magicserver log.txt

arguments:

magicclient1- host name where the node has to be setup – must be present in the n/w config file

magicserver - netname

log.txt - trace filename is the name of some file where we want the results to be stored

4.2 Evaluations and comparison

The white box model is implemented over *Magician*, a tool for implementing the Active networks. Magician is modified in order to incorporate CPU usage control. The CPU time needed for the execution of each packet in the first node is found out and stored. Here, the execution time of each packet in the first node is taken as the predicted value. SmartPing and SmartRoute are the applications which send the active packets. One malicious packet is intruded in between 5 good packets. Each malicious packet is programmed to consume as much CPU time as possible on each node. The EE monitors the execution of each active packet, interrupting them on a regular basis to query their execution time. If this execution time is below the predicted, then the packet continues its execution. Otherwise the EE kills it. Once the packet completes its execution, or when it's killed, the EE writes the information about the packet in the MIB (trace file): increments the number of packets killed or completed, and modifies the average CPU time used (computed over the last 20 packets) and all these information about the packet are stored. The average CPU time, the mean time and the variance CPU time are calculated and also stored in the trace file. The percentage of error, (i.e) the difference between the predicted time and the executed time is found out. The CPU time wasted for identifying and killing the malicious packet is also stored in MIB.

The characteristics of the heterogeneous platform selected for the control demo is presented in TABLE I.

Characteristics of three computer platform selected:

Table 1: CPU control and prediction demo-platform.

	Platform Description		
Node Name	Server	Node1 – AN-1	Node2 – AH-1
Processor Speed	3 GHz	2.4 GHz	1.2 GHz
Processor Architecture	Pentium IV	Pentium III	Celeron
O.S / Version	Red hat Linux / 7.0	Red hat Linux / 7.0	Red hat Linux / 7.0
Java Virtual Machine / Version	Jdk1.3.1_02	Jdk1.3.1_02	Jdk1.3.1_02
Memory size(Mega bytes)	512 MB	256 MB	128 MB

The results from the control demo and prediction are again analyzed and compared against two applications and two nodes. TABLE II gives the node-wise CPU utilization between two applications. The time taken for executing the two applications like Smartping and SmartRoute are given in the table II. The CPU time spent in Node1 and Node2 resembles the predicted value, which is shown in Table III. CPU predicted timings-report is presented in TABLE III. The predicted value is the time taken for executing the packet in the first node. If any packet with the active application executes beyond the predicted time in other nodes, the packet is identified as the malicious packet and it is killed. The average CPU time and variance in CPU time, calculated and stored in a log file by the Execution Environment is presented in TABLE III. The average CPU time and the variance in CPU time almost resembles the predicted time.

Comparison between applications

Table 2: CPU time usage-node wise report.

	CPU time Used (instruction cycles)		
Application	In Source	During Transition	
		Node1	Node2
SmartPing	13,500	13,700	13,740
SmartRoute	65,084	65,110	65,169

Table 3: CPU predicted timings-report (instruction cycles).

EE EE	AA	Predicted value	Avg CPU Time	Var CPU time
Magician	SmartPing	13,750	13,587	13,587
	SmartRoute	65,200	65,104	65,104

The percentage of error is presented in TABLE IV and the wasted CPU time for identifying and killing the malicious packets is also found. Percentage error is calculated as: Percentage Error = 100 * (prediction – actual) / actual. The actual CPU time is the one measured

in the first node. The percentage error between the predicted and actual CPU time is presented in TABLE IV. The variation is minimum between the predicted and actual timings.

Table 4: Error report.

EE	AA	Node	Percentage Error
Magician	SmartPing	Source	1.85
		Node1	0.36
		Node2	0.07
	SmartRoute	Source	0.18
		Node1	0.14
		Node2	0.05

Control demo – results: - 30 packets were sent and out of which 6 malicious packets were identified between 5 good packets and discarded.

The malicious packets were distinguished by evaluating their execution time, which goes beyond the estimated. It was found that the avg-wasted time for identifying and killing the malicious packets is 8.29 ms per packet. The total time taken per node is 49.74 ms. The CPU demand is calculated and reported for a heterogeneous setup.

5 Estimation of CPU time

The simplest estimation scheme is to measure the actual computation time offline as done in the above models, and include this value in all packets. The Estimation Based Fair Allocation algorithm can use this value for the estimation. This scheme has some drawbacks. The execution time of a program is dependent of the data and also dependent on the particular machine where it is executed. Different cache sizes, for example, can cause a program to take different amount of times, although the same sequence of instructions is executed. Additionally, a protocol is required to include the estimates in the packets, which is a considerable overhead. To avoid these problems, we have focused on estimation schemes [16, 18] that use local results to predict the next packet's execution time. We identified the estimation techniques for CPU estimation:

Constant

The constant estimate is the simplest estimator. The estimated computation time for queue i in round n, $estimate_{i,n}$, is always the same for all packets. If queues correspond to different traffic classes, this information can be used to select the constant.

$$estimate_{i,n} = estimate_{i,n-1} = const.$$

Exponential average

The exponential average is a common method for an adaptive estimation [15, 18] that combines the most current execution time, $actual_n$, with the previous results. It is defined as:

$$estimate_{i,n} = \alpha \,.\, actual_{i,n} + (1-\alpha) \,.\, estimate_{i,n-1}$$

The parameter α specifies how much of the previous history is preserved. This scheme is used in many of practical applications, e.g TCP round-trip delay estimation.

Packet size dependent estimate

While the exponential average works well in practice, it ignores the size of the packet [14] that is going to be processed. The packet size dependent estimate is defined as:

$$estimate_{i,n} = f_n(size(P_n));$$

where the function f_n maps the packet size p_n, to a processing time. The function f_n is adopted by the estimator E as

$$f_n = E(f_{n-1}, actual_{i,n})$$

The estimator E maps a packet size dependent estimation function to a new estimation function under consideration of the actual processing time. Any function can be used for estimation but polynomial functions seem to be most suitable, especially since a polynomial of order 0 can be represented with 0+1 variables. Depending on the precision of the required estimation, higher or lower order polynomials can be used.

6 Estimation based fair allocation

6.1 Estimation based fair allocation algorithm

We propose an allocation algorithm based on Adaptive estimations and DRR for servicing flows (queues) in an active node. For each queue, a deficit counter and an estimate is maintained. The deficit represents the amount of processing that this queue can use. The estimate represents the amount of processing that is expected for the next packet of this queue. The scheduler forwards the packets of a queue to the processor as long as the deficit is larger than the estimate of the next packet. When a packet uses excessive processing, the packet is interrupted by the timer. When the processing is finished or terminated, the actual processing time is used to adjust the deficit, as well as the estimate that is used for the next packet. The architecture is shown in Figure 4.

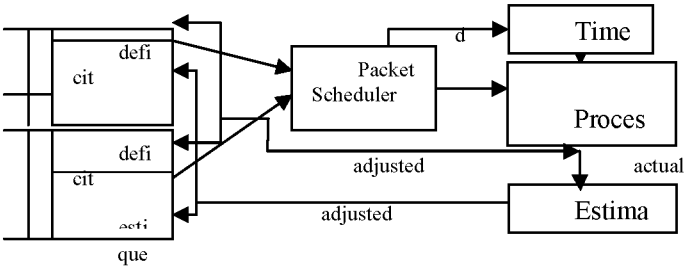


Figure 4: Estimation based fair allocation architecture.

Each network node stores packets coming from different flows in different queues. There are n queues. Each queue has initially no deficit and the estimated

processing time set to a default. The calculation of the estimation time is done using the exponential average method given under section V. The scheduling algorithm at the node selects a packet from the input queue, assigns it to CPU and runs the program associated with it until completion and then deposits it in the output queue. The algorithm (Figure 5) defines *round* to be a state in which the maximum number of packets allowable has been processed from all flows. The algorithm associates *Quantum* units with each flow *i* in each round. Each flow maintains a state variable d*eficit* that is initialized to *Quantum* before the start of each round of processing. The variable *cpu_estimate* maintains the number of CPU cycles required for a packet *p* in a flow during a round.

The main loop checks whether the deficit is positive, the deficit and the estimated processing time for the next packet is compared. If the cpu_estimate is less than the deficit, the packet is processed by p*rocess_packet_p*. If the current packet was previously interrupted, the old state is restored. A timer is also set to the deficit and started. The processing ends by means of two possibilities:

- If the packet used more time than it was permitted (the timer expires), the processing is preempted. The state of the processing is saved and the packet is pushed back into the head of the queue. The processing can then continue with that packet in the next round.
- If the processing is finished before the time expires, the packet is sent on and the next packet in the queue is considered.

The processing function returns the actual time that was used by the packet. The actual time is subtracted from the deficit. The estimator uses the actual time in adjust_estimate(), to adjust the estimation for the queue. If there is remaining deficit for the computation, then the next packet is considered for processing, otherwise the next queue is considered.

The total number of CPU cycles consumed by a flow in a round is maintained in a variable *total_cpu_con*. During each round, after a packet is processed from each flow, *cpu_estimate* is added to *total_cpu_con*. The number of packets processed from each flow in a round is within the restriction that d*eficit* > *total_cpu_con.* At the start of every new round, d*eficit* of the previous round is added to *quantum*. The ratio of q*uantum* given to any flows i, j is equal to the ratio of resource allocations for flows i, j. Also the algorithm only examines non-empty and backlogged flows.

6.2 Fairness

The algorithm is fair based on the following properties:

- The deficit counter is increased only once per round by the allotted quantum.
- If a queue does not make use of its entire share in a round, the amount is carried over to the next round in the deficit counter.
- The difference in total number of CPU cycles consumed between any two backlogged flows is bounded by a small constant.
- No queue receives more processor time than the deficit counter indicates. If the processing is interrupted by the timer, then the queue used its whole deficit and has to wait for the next round to receive more. If the processing terminates earlier, the deficit was not exceeded either.
- The deficit is charged only for the actual time that the processor was used.

```
For each flow i, get quantum
Assign Deficit for flow i as Quantum of i + Deficitof i
Calculate cpu_estimate_i
While (deficit_i >0 && deficit > total_cpu_con_i )
    If (deficit_i >=cpu_estimate_i),
       Start_timer(deficit_i);
          P = head(queue);
          If ( is_interrupted_packet(p)) then
            Restore_state(i);
          end if;
            Start_timer(deficit);
            actual_time = process_packet_p;
           if (process_interrupted) then
              save_state(i);
              enqueue_at_head(p,queue);
          end if;
         deficit_i = deficit_i - actual_time;
         Cpu_estimate =
                adjust_estimate(estimate,actual_time);
     else
      break;
     end if;
end while;
 if (empty(queue)) then
   deficit_i = 0;
end if;
total_cpu_con[i]= cpu_estimate+total_cpu_con[i];
increment totalround with total_cpu_con[i]
increment the round by one
assign prevround as totalround
end For
```

Figure 5: Estiamtion based fair allocation algorithm.

6.3 Results of EBFAA

To find out the performance of the EBFAA, we implemented the algorithm. In Figure 6, we used a single active node and one host configuration with twenty flows sending packets. The only exception is that Flow 10 is a misbehaving flow. While the ill behaved flow grabs an arbitrary share of bandwidth when the EBFAA is not used. While in EBFAA, there is nearly perfect fairness.

The quantum / timeslice ranges from .1 times to 100 times the average processing time of a packet. The algorithm is implemented using three estimates 'e' (e = .1, e = 1 or e = 10 times the average actual processing time). Here 'e' is the scalar constant chosen as the

estimates. The estimation time is closer to actual processing time when e = 1. The algorithm incurs fewer context switches for quantum sizes in the range of the actual processing time.

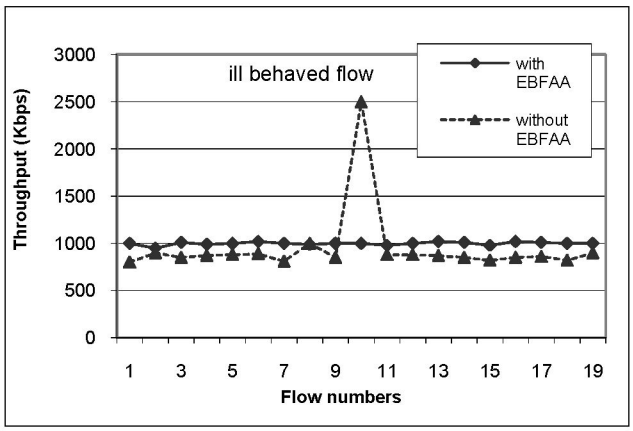


Figure 6: Control of the malicious flow.

We measured the delay rates for 20 flows. Each flow reserves the same processing rate, and sends packets randomly at specified average time intervals to just saturate its share. The sum of average processing rates of all flows is just under the processor capacity. Therefore, the delays measured are mainly due to scheduling not due to queuing backlog. The results in Figure 7 show that EBFAA provides lower maximum delays to all flows, when compared with WFQ, SFQ and SWFQ. EBFAA also gives smaller delay standard deviations than SFQ and SWFQ for all flows. Reduction in delay standard deviations would reduce the delay jitters. We expect that EBFAA would give better delay behavior due to its more accurate system virtual time, especially, where variations in processing requirements of packets are large. Figure 6 shows that WFQ provide smaller maximum delays than SWFQ, SFQ and EBFAA for application flows that have low processing time per packet to reserved rate ratio. However, EBFAA can provide lower maximum delays for all packets in flows when compared with WFQ. We propose to use EBFAA for processing resource scheduling in programmable networks to support QoS in two categories: processing resource reservation, and best-effort. We believe that EBFAA is also applicable for processor scheduling in operating systems in general.

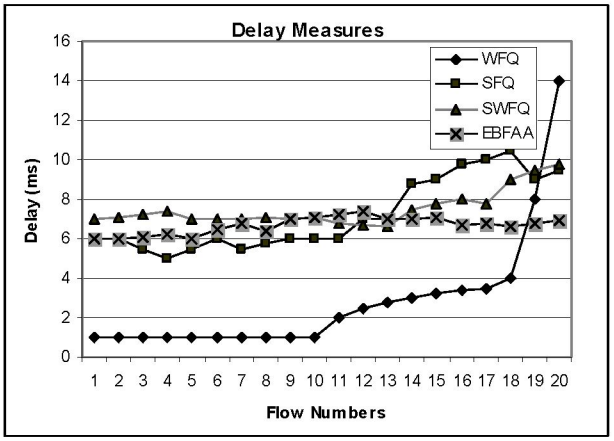


Figure 7: Delay measures

7 Conclusion and future work

This paper examines a way to analyze the CPU resource control and Fair Resource Allocation to improve the Quality of Service (QoS) in a heterogeneous active network environment. It is discussed that some means are needed to accurately specify the CPU demand in order to safely and efficiently deploy mobile code among heterogeneous platforms in a network. This paper has described an approach (White Box model) and an algorithm (EBFAA) to control the CPU utilization of malicious packets and to estimate the CPU demand for good packets in a heterogeneous active network environment and evaluated the approach. In the control application, it is demonstrated to identify the malicious packets, when malicious or erroneous code is injected in to a node and that the amount of CPU time stolen or wasted has been found out and can be reduced. The results from the control demo and prediction model are again analyzed and compared against various applications and nodes. The percentage error between the predicted and the actual CPU time is also less for this prediction model. The algorithm (EBFAA) also provides near-perfect fairness during resource control and allocation. Thus using this resource control model, the network management systems can allocate the capacity better by anticipating varying demands and the network operators can better estimate the quality of service (QoS) that customers can expect.

This work can be extended and can be compared with different Execution environments. White-box models could be combined with histograms and Monte-Carlo simulations to yield reasonably accurate estimates. In the case of white-box models, the histograms would represent the CPU usage observed during calibration for each primitive provided by the execution environment. We have future plans to investigate these ideas in the context of resource management for mobile code loaded into call-processing servers. The issue of determining the CPU requirement for active packet can also be resolved by introducing a policy base [6, 18] at the active node. Combined scheduling algorithms which could schedule both CPU and bandwidth resources adaptively and fairly among all the competing flows can be applied. This work can also be extended for the prediction of the resources in wireless and sensor networks.

Acknowledgment

This work is supported by Technology Information Forecasting and Assessment Council (TIFAC) under the Department of Science and Technology (DST), Government of India and the Centre of Relevance and Excellence (CORE) in Network Engineering at Kalasalingam University, Krishnankoil-626190, Tamil Nadu, India.

References

[1] Amit. B. Kulkarni, "Magician–An Active NetworkingToolkit", http://ittc.ukans.edu/projects/-Magician, 2000.

[2] Ball. T and Larus. J.R, "Using paths to measure, explain, and enhance program behavior", IEEE Computer, July 2000.
[3] Berman. F, Wolski. R., Figueira. S, Schopf. J, and Shao. G, "Application-level scheduling on distributed heterogeneous networks", in Supercomputing '96, September, 1996.
[4] Calvert K. L., Griffioen J., Mullins B., Sehgal A. and Wen S., "Concast: "Design and Implementation of an Active Network Service", IEEE Journal on Selected Area in Communications (JSAC), Volume 19, Issue 3, 2001.
[5] Calvert. K. L., "Architectural Framework for Active Networks", Version 1.0 Draft July 27, 1999.
[6] Christos Tsarouchis et.al. , "A Policy-Based Management Architecture for Active and Programmable Networks", IEEE Network, May/June 2003.
[7] Huber O.r J. and Toutain. L, "Mobile Agents in Active Networks", ECOOP'97, Workshop on Mobile Object Systems, June 1997.
[8] Rohan De Silva, ," A Security Architecture for Active Networks", Proceedings of the 4th ACM WSEAS International Conference on Applied Informatics and Communications, 2004.
[9] Larry Peterson and AN Node OS Working Group, "NodeOS Interface Specification", January 2000.
[10] Moore J.T., Hicks M., and Nettles S. "Practical programmable packets", *In IEEE InfoCom 2001*.
[11] Peterson. L, Gottlieb. Y, Schwab. S, Rho. S, Hibler. M, Tullmann. P, Lepreau. J, and Hartman. J, "An OS Interface for Active Routers", IEEE Journal on Selected Areas in Communications, 2001.
[12] Rauf Bautaba, University of Waterloo, Andreas Polyrakis, University of Toronto, "Projecting Advanced Enterprise Network and Service Management to Active Networks", IEEE Network, Jan/Feb 2002.
[13] Reynolds. J and Postel. J. "RFC 1700 Assigned Numbers", October 1994.
[14] Saavedra-Barrera R. H. Smith A. J., and Miya. E., "Machine characterization based on an abstract high-level language machine". *IEEE Transactions on Computers*, December 1989.
[15] Sohil Munir, "A survey of Active Network Research", IEEE Communications Magazine: vol. 35, no.1, pp 80-86, Jan 1997.
[16] VimalaDevi K. & Mehata K.M., "Resource Estimation and Policy Based Allocation for Qualityof Service in Active Networks", IEEE workshop on Coordinated Quality of Service in Distributed Systems (COQODS-II), held in conjunction with 14th IEEE ICON2006, in Singapore from September 13 to 15, 2006.
[17] VimalaDevi K. &. Mehata, K.M "Advancing Performance Management using Active Network Technology", NCCN'03, S.R.M.Engg. College, Chennai, India, Feb 2003.
[18] Cen, et al. "A distributed real-time MPEG video audio player", In Proceedings of Internal Workshop on Network and Operating System support for Digital and video (NOSSDAV), Lecture Notes in Computer Science, pages 151-162, Durham, New Hampshire, April 1995, Springer.
[19] Wetherall,. D, Guttag. J and Tennenhouse. D, "ANTS: Network Services without the Red Tape", *IEEE Computer*, pp. 42-48. (8), April 1999.
[20] Yamamoto. L and Leduc. G, "An agent-inspired active network resource-trading model applied to congestion control". *In MATA 2000*, pages 151–169, Sep 2000.

Identification and Prediction Using Neuro-Fuzzy Networks with Symbiotic Adaptive Particle Swarm Optimization

Cheng-Jian Lin and Chun-Cheng Peng
Department of Computer Science and Information Engineering
National Chin-Yi University of Technology, Taichung County, Taiwan 411, R.O.C.
E-mail: cjlin@ncut.edu.tw, goudapeng@gmail.com

Chi-Yung Lee
Department of Computer Science and Information Engineering
Nankai University of Technology, Nantou, Taiwan 542, R.O.C.
E-mail: cylee@nkut.edu.tw

Keywords: particle swarm optimization, symbiotic evolution, neuro-fuzzy network, identification, prediction

Received: October 16, 2009

This study presents a novel symbiotic adaptive particle swarm optimization (SAPSO) for neuro-fuzzy network design. The proposed SAPSO uses symbiotic evolution and adaptive particle swarm optimization with neighborhood operator (APSO-NO) to improve the performance of the traditional PSO. In APSO-NO, we combine the neighborhood operator and the adaptive particle swarm optimization to tune the particles that are most significant. Simulation results have shown that the proposed SAPSO performs better and requires less computation time than the traditional PSO.

Povzetek: Razvita je nova metoda nevronskih mrež z uporabo roja delcev.

1 Introduction

Neuro-fuzzy networks (NFNs) have been demonstrated to be successful [1]-[9]. Two typical types of NFNs are the Mamdani-type and TSK-type models. In Mamdani-type NFNs [3]-[4], the minimum fuzzy implication is used in fuzzy reasoning. In TSK-type NFNs [5]-[8], the consequent of each rule is a function input variable. The generally adopted function is a linear combination of input variables plus a constant term. Many researchers [6]-[7] have shown that using TSK-type NFNs achieve superior performance in network size and learning accuracy than using Mamdani-type NFNs.

Training parameters is a problem in the design of a NFN. To solve this problem, back-propagation (BP) training is widely used [3]-[8]. It is a powerful training technique that can be applied to networks. Nevertheless, the steepest descent technique is used in BP training to minimize the error function. The algorithm may allow the local minima to be reached very quickly, yet the global solution may never be found. In addition, the performance of BP training depends on the initial values of the system parameter, and for different network topologies, new mathematical expressions for each network layer have to be derived. Considering the disadvantages mentioned above, one might be faced with suboptimal performances, even for a suitable NFN topology. Hence, techniques capable of training network parameters and finding a global solution while optimizing the overall structure are needed.

In this respect, a new algorithm, called particle swarm optimization (PSO), appears to be a better algorithm than the BP algorithm. It is an evolutionary computation technique developed by Kennedy and Eberhart in 1995 [10]. The underlying motivation for the development of the PSO algorithm was the social behavior of animals, such as birds flocking together, fish swimming in schools, and insects swarming together. Several researchers have used the PSO method to solve some optimization problems, like control problems [11]-[13] and neural network design [14]-[15].

The performance of most stochastic optimization algorithms, including the PSO and genetic algorithms (GAs), declines as the dimensionality of the search space increases. These algorithms stop when they generate a solution that falls in the optimal region, a small volume of the search space surrounding the global optimum. The probability in stochastic optimization algorithms decreases exponentially as the dimensionality of the search space increases. It is clear that, in a similar topology, it is harder to find the global optimum in a high-dimensional problem than it is in a low-dimensional problem. One way to overcome this exponential increase in difficulty is to partition the search space into lower dimensional subspaces, as long as the optimization algorithm can guarantee that it will be able to search every possible region of the search space.

In this paper, a novel learning algorithm, called symbiotic adaptive particle swarm optimization (SAPSO), that tunes the parameters of NFNs is proposed. The proposed SAPSO is different from the traditional PSO. In the traditional PSO, each particle represents a fuzzy system. But in SAPSO, each particle represents only one fuzzy rule. A R-rule fuzzy system is constructed by

selecting and combining R particles from a given swarm. The proposed SAPSO consists of symbiotic evolution and adaptive particle swarm optimization with neighborhood operator to improve the performance of the traditional PSO.

The advantages of the proposed SAPSO are summarized as follows: (1) SAPSO can reduce population sizes; (2) the computation request of SAPSO is less than that of the traditional PSO in each generation; (3) in the learning process, the relative parameters in a fuzzy system are searched; they prevent interference from other parameters that can find what the best parameter values are; (4) the adjusting parameter strategy of SAPSO is more significant than the traditional PSO.

The rest of this paper is organized as follows. After reviewing of training algorithms for NFNs in Section 2, Section 3 illustrates the structure of the TSK-type fuzzy model. An overview of PSO is given in Section 4. A novel symbiotic adaptive particle swarm optimization (SAPSO) is proposed in Section 5. Sections 6 and 7 respectively present the simulation results and discussion. Finally, the conclusion is given in the last section.

2 Related works

Besides the most-applied BP algorithm, some other traditional optimization approaches had been applied to training NFNs, such as the Broyden-Fletcher- Goldfarb-Shanno (BFGS) [16]-[17], conjugate gradient (CG) [18]-[19], and Levenberg-Marquardt (LM) [20]-[21] methods.

In the context of deterministic unconstrained optimisation, quasi-Newton (QN) methods, sometimes called variable metric methods, are well-known algorithms for finding local minima of specific functions. QN methods are based on Newton's method to find the stationary point of a function, where the gradient is zero. Newton's method assumes that the function can be locally approximated as a quadratic in the region around the optimum, and requires the first and second derivatives [22], i.e. the gradient vector and the Hessian matrix, to find the stationary point. Moreover, the Newton's method and its variants require that the Hessian is positive definite - a condition that is difficult to guarantee in practice.

Conjugate Gradient methods are in principle approaches suitable for large-scale problems [23]. The basic idea of CG methods is to find the stepsize along a linear combination of the current gradient vector and the previous search direction. On the other hand, equipped with a damping factor, the LM (so-called *damped Gauss-Newton*) methods are capable of relaxing the difficulties of Hessian-based training, i.e. the ill-conditioning of the Hessian matrix. In addition, when the damping factor is zero, the LM methods become identical to the Gauss-Newton approach; while as the damping factor gets close to infinity, the LM methods are then get equivalent to the steepest descent method.

As indicating in the Introduction section, although the traditional second-order approaches generally have faster convergent speeds, they are still in the situation of local optimization. Evolutional approaches such as particle swarm optimization (PSO) [10], differential evolution (DE) [21], and symbiotic evolution (SE) [25] have been developed for training NFNs [26]-[28], respectively. Since this paper focuses on the PSO approach, the concepts of DE and SE methods are omitted here and suggested to refer the relevant literature for further details, whereas the overview of the PSO and our proposed symbiotic adaptive PSO are presented in this paper.

3 Structure of a TSK-type neuro-fuzzy network (TNFN)

A fuzzy model is a knowledge-based system characterized by a set of rules that model the relationship between the control input and output. The reasoning process is defined by means of the inference method, aggregation operators, and fuzzy connectives. The fuzzy knowledge base contains the definitions of fuzzy sets, which are stored in a fuzzy database, and a collection of fuzzy rules.

Fuzzy rules are defined by their antecedents and consequents, which relate an observed input state to a desired control action. Most fuzzy systems employ the inference method proposed by Mamdani, in which the consequent parts are defined by fuzzy sets [1]. A Mamdani-type fuzzy rule has the form:

$$\textit{IF } x_1 \textit{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textit{ and } x_2 \textit{ is } A_{2j}(m_{2j}, \sigma_{2j}) \ldots \textit{and } x_n \textit{ is } A_{nj}(m_{nj}, \sigma_{nj})$$
$$\textit{THEN } y' \textit{ is } B_j(m_j, \sigma_j) \quad (1)$$

where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation, respectively, of the ith dimension and the jth rule node. The consequents Bj of the jth rule are aggregated into one fuzzy set for the output variable y'. Crisp action is obtained through defuzzification, which calculates the centroid of the output fuzzy set. The more common fuzzy inference method proposed by Mamdani, Takagi, Sugeno, and Kang introduced a modified inference scheme [5]. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. A Takagi-Sugeno-Kang (TSK)-type fuzzy model employs different implications and aggregation methods than the standard Mamdani controller. Instead of fuzzy sets being used, the conclusion part of a rule is a linear combination of the crisp inputs.

$$\textit{IF } x_1 \textit{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textit{ and } x_2 \textit{ is } A_{2j}(m_{2j}, \sigma_{2j}) \ldots \textit{and } x_n \textit{ is } A_{nj}(m_{nj}, \sigma_{nj})$$
$$\textit{THEN } y' = w_{0j} + w_{1j}x_1 + \ldots + w_{nj}x_n \quad (2)$$

Since the consequent of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Instead, the control output is computed as the weighted average of the crisp rule outputs. This computation is less expensive than calculating the center of gravity.

The structure of the TSK-type neuro-fuzzy network (TNFN) is shown in Fig. 1, where n and R are, respectively, the number of input dimensions and the number of rules. It is a five-layer network structure. The functions of the nodes in each layer are described as follows:

Layer 1 (*Input Nodes*): No function is performed in this layer. The node only transmits input values to layer 2.

$$u_i^{(1)} = x_i, \quad i = 1 \cdots n \tag{3}$$

Layer 2 (*Membership Function Nodes*): Nodes in this layer correspond to one linguistic label of the input variables in layer 1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is calculated in this layer. For an external input x_i, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right), \quad j = 1 \cdots R \tag{4}$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i.

Layer 3 (*Rule Nodes*): The output of each node in this layer is determined by the fuzzy AND operation. Here, the product operation is utilized to determine the firing strength of each rule. The function of each rule is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{5}$$

Layer 4 (*Consequent Nodes*): Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output delivered from layer 3, and the other inputs are the input variables from layer 1, as depicted in Fig. 1. For this kind of node, we have

$$u_j^{(4)} = u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i\right) \tag{6}$$

where the summation is over all the inputs, and w_{ij} are the corresponding parameters of the consequent part. w_{ij} can be any real value. If w_{ij}=0, i >0, the TNFN model in this case will be called the zero-order TNFN model.

Layer 5 (*Output Node*): Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i\right)}{\sum_{j=1}^{R} u_j^{(3)}} \tag{7}$$

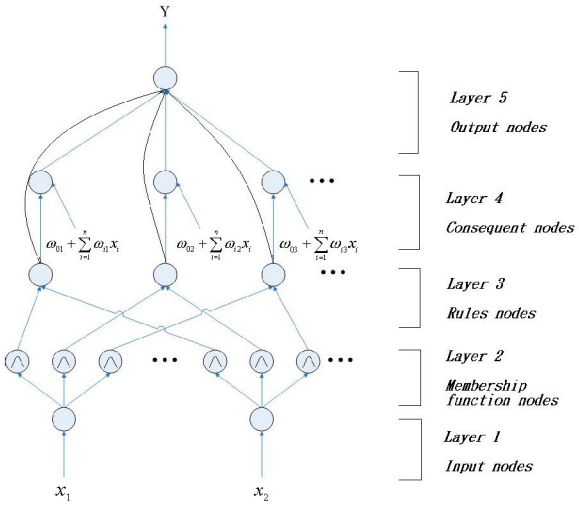


Figure 1: Structure of the TNFN model.

4 An overview of particle swarm optimization

Particle swarm optimization (PSO) [10] is a recently invented high performance optimizer that possesses several highly desirable attributes, including a basic algorithm that is easy to understand and implement. This algorithm is similar to genetic algorithms and evolutionary algorithms, but requires less computational memory and fewer lines of code. The PSO conducts searches using a population of particles which correspond to individuals in GA. Each particle has a velocity vector $\vec{v}_i$ and a position vector $\vec{x}_i$ to represent a possible solution.

Consider an optimization problem that requires the simultaneous optimization of variables. A collection or swarm of particles are defined, where each particle is assigned a random position in the N-dimensional problem space so that each particle's position corresponds to a possible solution of the optimization problem. The particles fly rapidly, moving at their own respective velocities, and search the space. PSO has a simple rule, namely, that each particle has three choices in evolution: (1) insist on itself; (2) move towards its own current best position; each particle remembers its own personal best position that it has found, called the local best; (3) move towards the current best position of the population; each particle also knows the best position found by any other particle in the swarm, called the global best. PSO reaches a balance among these three choices.

At each time step, each of the particle positions is scored to obtain a fitness value based on how well it solves the current problem. Using the local best position (*Lbest*) and the global best position (*Gbest*), a new velocity for each particle is updated using

$$\vec{v}_i(k+1) = \omega * \vec{v}_i(k) + \phi_1 * rand() * (Lbest - \vec{x}_i(k)) + \phi_2 * rand() * (Gbest - \vec{x}_i(k)) \quad (8)$$

where ω, ϕ_1, and ϕ_2 are called the coefficient of inertia, the cognitive study, and the society study, respectively. The term *rand()* refers to uniformly distributed random numbers in [0, 1]. The term $\vec{v}_i$ is limited to the range $\pm \vec{v}_{\max}$. If the velocity violates this limit, it will be set to its proper limit. The concept of the updated velocity is illustrated in Fig. 2.

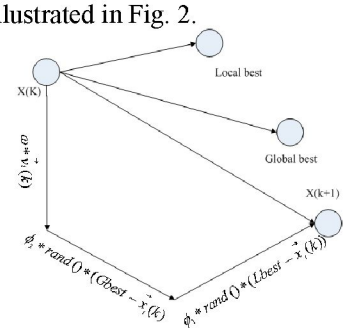


Figure 2: The diagram of the updated velocity in the PSO.

A variable velocity enables every particle to search around its individual best position and the global best position. Based on the updated velocities, each particle changes its position according to the following:

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \quad (9)$$

When every particle is updated, the fitness value of each particle is calculated again. If the fitness value of the new particle is higher than those of the local best/global best, then the local best/global best will be replaced with the new particle. When the above updating processes are repeated step-by-step, the whole population will evolve toward the optimum solution. A detailed flowchart is shown in Fig. 3.

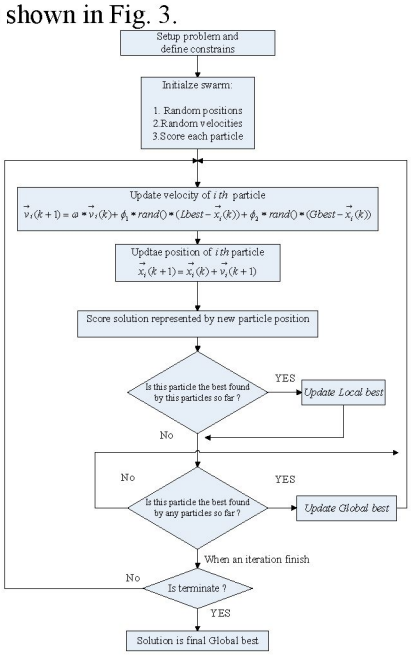


Figure 3: The typical PSO flowchart illustrates the steps and update equations.

5 The symbiotic adaptive particle swarm optimization (SAPSO)

In this section, we will introduce the symbiotic adaptive particle swarm optimization (SAPSO) for NFN design. SAPSO uses symbiotic evolution and adaptive particle swarm optimization with neighborhood operator. The detailed process is described below.

5.1 The Design of Neuro-Fuzzy Network Using SAPSO

Symbiotic evolution was first proposed in an implicit fitness-sharing algorithm that was used in an immune system model [31]. Unlike the traditional PSO that uses each particle in a swarm as a full solution to a problem, in symbiotic evolution, each individual in a population represents only a partial solution to a problem. The goal of each individual is to form a partial solution that will be combined with other partial solutions currently in the population to build an effective full solution. In a normal evolution algorithm, a single individual is responsible for the overall performance, and is assigned a fitness value according to its performance. In symbiotic evolution, the fitness of an individual (a partial solution) is calculated by summing up the fitness values of all possible combinations of that individual with other current individuals (partial solutions) and then dividing the sum by the total number of combinations. The representation of a fuzzy system using SAPSO is shown in Fig. 4.

In Fig. 4, we can see that if we need R rules to construct a fuzzy system, we will have R sub-swarms. Each sub-swarm produces its own sub-particles. The current best parameters, called the cooperative best (Cbest), of the fuzzy system are recorded. As with the traditional PSO, the velocities and sub-particles in every sub-swarm need to be updated.

The evolution process of SAPSO includes coding,

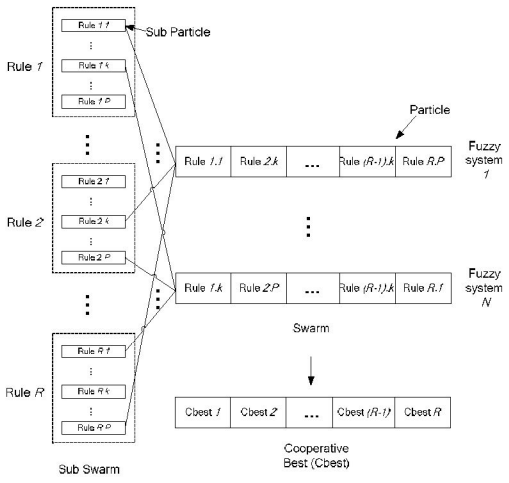


Figure 4: The representation of a fuzzy system by SAPSO.

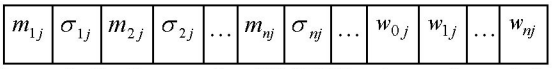


Figure 5: Coding a rule of SAPSO into a sub-particle.

initialization, fitness assignment, and sub-particle updating. The coding step is concerned with the membership functions and the fuzzy rules of a fuzzy system that represent the particles in SAPSO. The initialization step assigns the sub-swarm values before the evolution process. The fitness assignment step gives a suitable fitness value to each fuzzy system during the evolution process. The complete learning process is described step by step below.

A. Coding step: The first step in SAPSO is to code a fuzzy rule into a sub-particle. Figure 5 shows a fuzzy rule that is given by Eq. (2), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation in the ith dimension and the jth rule node, respectively.

B. Initialization step: Before SAPSO is designed, an initial sub-swarm should be generated. As in the traditional PSO, an initial sub-swarm is generated randomly within a fixed range.

C. Fitness assignment step: As mentioned before for SAPSO, the fitness value of a rule (a sub-particle) is calculated by summing up the fitness values of all the possible combinations, which are randomly selected, and then dividing the sum by the total number of combinations. The details for assigning the fitness value are described step-by-step as follows.

Step 1: Randomly choose one sub-particle from each sub-swarm and assemble it to form a particle. This particle represents a fuzzy system derived from these sub-particles.

Step 2: Evaluate the performance of every fuzzy system that is generated from Step 1 to obtain a fitness value.

Step 3: The fitness records are initially set to zero. Accumulate the fitness value of the selected sub-particles to the fitness records.

Step 4: Repeat the above steps until each rule (sub-particle) in a sub-swarm has been selected a sufficient number of times, and record how many times each sub-particle has participated in the fuzzy systems.

Step 5: Divide the accumulated fitness value of each sub-particle by the number of times it has been selected. The average fitness value represents the performance of a rule. In this paper, the fitness value is designed according to the follow formulation:

$$\text{Fitness Value} = 1/(1+\sqrt{E(y,\bar{y})/T}) \tag{10}$$

$$\text{where } E(y,\bar{y}) = \sum_{i=1}^{T} (y_i - \bar{y}_i)^2 \tag{11}$$

where y_i represents the true value of the ith output, $\bar{y}_i$ represents the predicted value, $E(y,\bar{y})$ is an error function, and T represents the number of the training data of each generation.

D. Updating velocities and sub-particles: When the fitness value of each sub-particle is obtained from the fitness assignment step, the *Lbest* of each sub-particle and the *Gbest* of each sub-swarm are updated simultaneously using adaptive particle swarm optimization with neighborhood operator (APSO-NO). The algorithm of APSO-NO is described in subsection 4.2.

E. Updating cooperative best (Cbest): When the fitness value of every fuzzy system is obtained, we can find the best fuzzy system in each generation. If the fitness value of any fuzzy system is higher than the best cooperative one, the cooperative best will be replaced.

The steps mentioned above are repeated until the predetermined condition is achieved.

5.2 The Adaptive Particle Swarm Optimization with Neighborhood Operator (APSO-NO)

In recent years, many researchers [30], [32] have proposed using stability analysis on dynamic PSO in order to obtain an understanding on how it searches for a global optimal solution and the strategy it uses to tune parameters.

In this paper, the velocity of a particle at the $(k+1)$-th iteration is redefined for SAPSO as follows:

$$\begin{aligned}\vec{v}_i(k+1) = {} & \omega * \vec{v}_i(k) + \phi_1 * rand() * (Lbest - \vec{x}_i(k)) \\ & + \phi_2 * rand() * (Gbest - \vec{x}_i(k)) \\ & + \phi_3 * rand() * (Cbest - \vec{x}_i(k))\end{aligned} \tag{12}$$

where ω, ϕ_1, ϕ_2, and ϕ_3 are called the coefficient of inertia, cognitive study, group study, and society study, respectively. We hope to accelerate every sub-particle in a direction toward the best self (Gbest), the best of a partial solution (Lbest), and the best of the full solution (Cbest). The particle will be reduced to one dimension for easy analysis. Thus, Eq. (12) is rewritten as follows:

$$v(k+1) = \omega * v(k) + \alpha * (Z - x(k)) \tag{13}$$

$$\text{where } \alpha = \phi_1 * rand() + \phi_2 * rand() + \phi_3 * rand() \tag{14}$$

$$Z = \frac{\phi_1 * rand() * Lbest + \phi_2 * rand() * Gbest + \phi_3 * rand() * Cbest}{\alpha} \tag{15}$$

The reduced formulas of APSO-NO can be expressed as follows:

$$\begin{aligned} v(k+1) &= \omega * v(k) + \alpha * y(k) \\ y(k+1) &= -\omega * v(k) + (1-\alpha) * y(k) \\ where\ \ y(k) &= Z - x(k) \end{aligned} \tag{16}$$

The reduced system can then be written into a form using matrix algebra.

$$P_{k+1} = MP_k \tag{17}$$

$$where\ P_k = \begin{bmatrix} v(k) \\ y(k) \end{bmatrix},\ M = \begin{bmatrix} \omega & \alpha \\ -\omega & 1-\alpha \end{bmatrix}$$

More generally, $P_k = M_k P_0$. Thus, the system is defined completely by M. The eigenvalues of M are

$$\begin{cases}\lambda_1=\dfrac{\omega+1}{2}-\dfrac{\alpha}{2}+\dfrac{\sqrt{(\omega+1-\alpha)^2-4\omega}}{2}\\ \lambda_2=\dfrac{\omega+1}{2}-\dfrac{\alpha}{2}-\dfrac{\sqrt{(\omega+1-\alpha)^2-4\omega}}{2}\end{cases} \quad (18)$$

According to stability theory, the behavior of a particle is stable if and only if $|\lambda_1|<1$ and $|\lambda_2|<1$. Since the eigenvalues λ_1, λ_2 are a function of the parameters ω, ϕ_1, ϕ_2, and ϕ_3, eigenvalue analysis will be carried out under the following four conditions to find the stable condition of the system. The detailed proofs refer to [30].

$$\begin{cases}(1)\ \omega=0, & \Rightarrow 0<\alpha<2\\ (2)\ \alpha<\omega+1-2\sqrt{\omega}, & \Rightarrow 0\le\omega<1\\ (3)\ \omega+1-2\sqrt{\omega}<\alpha<\omega+1+2\sqrt{\omega}, & \Rightarrow 0\le\omega<1\\ (4)\ \omega+1+2\sqrt{\omega}<\alpha, & \Rightarrow 0\le\omega<1\end{cases} \quad (19)$$

Based on the above analysis and with the use of the parameters α and ω, the criterion of convergence $|\lambda_1|<1$ and $|\lambda_2|<1$ can be written as follows:

$$\begin{aligned}&0<\alpha<2\omega+2\\ &0\le\omega<1\end{aligned} \quad (20)$$

The analysis of vibration for a period is also investigated in [30]. In condition (3) of Eq. (29), since $(\omega+1-\alpha)^2-4\omega<0$ when $\omega+1-2\sqrt{\omega}<\alpha<\omega+1+2\sqrt{\omega}$, λ becomes a complex number, and α can be described as follows:

$$\begin{aligned}&\alpha=\omega+1-2\sqrt{\omega}+4\delta\sqrt{\omega}\\ &where\ 0<\delta<1\end{aligned} \quad (21)$$

Because λ is a complex number, it can be expressed as a polar coordinate $\lambda=|\lambda|e^{\theta}$. If T is to be the period of the reduced system,

$$\begin{aligned}T&=\frac{2\pi}{\theta}=\frac{2\pi}{\tan^{-1}\dfrac{\mathrm{Im}}{\mathrm{Re}}}\\ &=\frac{2\pi}{\tan^{-1}\dfrac{\dfrac{\sqrt{-(\omega+1-\alpha)^2+4\omega}}{2}}{\dfrac{\omega+1-\alpha}{2}}},\quad \alpha<\omega+1\end{aligned} \quad (22)$$

If Eq. (21) is used instead of α, then Eq. (22) can be written as follows:

$$\begin{aligned}T&=\frac{2\pi}{\tan^{-1}\dfrac{2\sqrt{\delta-\delta^2}}{1-2\delta}} \quad (\delta<0.5)\\ &=\frac{2\pi}{\pi+\tan^{-1}\dfrac{2\sqrt{\delta-\delta^2}}{1-2\delta}} \quad (\delta>0.5)\end{aligned} \quad (23)$$

It was shown from Eq. (14) that is a random number distributed in [0, $(\phi_1+\phi_2+\phi_3)$] and that its average value is $\frac{\phi_1+\phi_2+\phi_3}{3}$. We use three parameters, κ_1, κ_2 and κ_3, where $0<\kappa_1+\kappa_2+\kappa_3<1$. Therefore, ϕ_1, ϕ_2, and ϕ_3 is

$$\phi_1:\phi_2:\phi_3=\kappa_1:\kappa_2:\kappa_3 \Rightarrow \begin{cases}\phi_1=3\kappa_1\alpha\\ \phi_2=3\kappa_2\alpha\\ \phi_3=3\kappa_3\alpha\end{cases} \quad (24)$$

We can redefine Eq. (12) based on the above results as follows:

$$\begin{aligned}\vec{v}_i(k+1)=&\omega*\vec{v}_i(k)+3\kappa_1\alpha*rand()*(Lbest-\vec{x}_i(k))\\ &+3\kappa_2\alpha*rand()*(Gbest-\vec{x}_i(k))\\ &+3\kappa_3\alpha*rand()*(Cbest-\vec{x}_i(k))\end{aligned} \quad (25)$$

where $\alpha=\omega+1-2\sqrt{\omega}+4\delta\sqrt{\omega}$ (26)

The qualitative relation between the search trajectory of the reduced system and the parameters is summarized as follows:

(1).If $0\le\omega<1$ and ω closes to 0, the convergence tendency of the system will increase. If $1<\omega$ and ω faces to ∞, the divergence tendency of the system will increase.

(2).If $0<\alpha<1$ and α closes to 1, the dynamics of the system becomes a vibration.

(3). If the parameters $0<\kappa_1+\kappa_2+\kappa_3<1$ and any of the other coefficients closes to 1, Z moves toward the corresponding best (*Lbest, Gbest or Cbest*).

In addition to the above method, another concept, called the neighborhood operator, is introduced in [15]. It prevents a swarm from tending to the global best prematurely in the search process. If it commits early to the global best in the search process, it may be trapped in a poor local minimum. In the neighborhood operator, the Gbest is not a fixed value. Therefore, Eq. (25) can be rewritten as follows:

$$\begin{aligned}\vec{v}_i(k+1)=&\omega*\vec{v}_i(k)+3\kappa_1\alpha*rand()*(Lbest-\vec{x}_i(k))\\ &+3\kappa_2\alpha*rand()*(Vbest-\vec{x}_i(k))\\ &+3\kappa_3\alpha*rand()*(Cbest-\vec{x}_i(k))\end{aligned} \quad (27)$$

where *Vbest* is defined as the best solution in the neighborhood around the sub-particles that are waiting to be updated. The neighborhood is identified by calculating the distances between the candidate sub-particles and the other sub-particles. The pseudo code is shown in Fig. 6. We can see that the number of neighborhoods gradually increases according to the generations. When the generations are close to terminal conditions, *Vbest* tends toward *Gbest*.

1. Get *dist[i]* by calculating distances between the candidate sub-particle and all other sub-particle.
2. Find the $dist_{\max}$ from *dist[i]*.
3. Define a threshold $\ell=0.5+0.5*(iteration_{now}/iteration_{\max})$
4. if ℓ <0.8
   if $\ell>dist[i]/dist_{\max}$

Figure 6: The pseudo code for finding Vbest in every iteration.

6 Simulation results

In this section, the proposed SAPSO is applied to the TNFN design and compared with the traditional PSO. Both SAPSO and the traditional PSO are used to adjust

the antecedent and consequent parameters of fuzzy rules in TNFN.

We use three different simulations for all methods. The first simulation uses the example given by Narendra and Parthasarathy [33]. The second simulation predicts the chaotic time series [34], and the third example approximates a piecewise function [35]. In our simulations, the numbers of swarms and sub-swarms are set to 50 and 10. The initial parameters of the traditional PSO and SAPSO are given in Table 1. In SAPSO, we use different parameter values to observe the effect on performance. All the programs are developed using MATLAB 6.1 software, and each problem is simulated on a Pentium III 1GHz desktop computer. Each experiment is run 20 times.

Parameter / Model	ω	$\kappa_1(\phi_1)$	$\kappa_2(\phi_2)$	κ_3	δ
Traditional PSO	0.4	2.0	2.0	NA	NA
SAPSO*1*	0.4	0.3	0.3	0.3	0.5
SAPSO*2*	0.4	0.5	0.5	0	0.5
SAPSO*3*	0.4	0.5	0	0.5	0.5
SAPSO*4*	0.4	0.2	0.4	0.4	0.5
SAPSO*5*	0.4	0.2	0.4	0.4	0.6-0.4
SAPSO*6*	0.4	0.1	0.3	0.6	0.3
SAPSO*7*	0.4	0.1	0.3	0.6	0.5
SAPSO*8*	0.4	0.1	0.3	0.6	0.7
SAPSO*9*	0.4	0.1	0.3	0.6	0.6-0.4

Table 1: The initial parameters of the traditional PSO and the SAPSO.

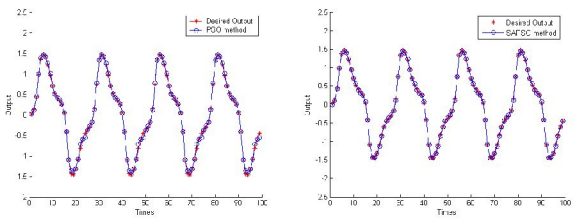
Example 1-Identification of Nonlinear Dynamic System

The first example used for identification is described by the difference equation

$$y(k+1)=\frac{y(k)}{1+y^2(k)}+u^3(k) \qquad (28)$$

The output of this equation depends nonlinearly on both its past value and the input, but the effects of the input and output values are not additive. The training input patterns are randomly generated in the interval [-2, 2] for the training data. In this problem, we use five fuzzy rules, and evolution progressed for 1000 generations.

After 1000 generations, the average best root mean square error (RMSE) of the output approximates 0.016. Figures 7 (a)-(b) show the outputs of the two methods for the input $u(k)=sin(2\pi k/25)$. Figure 8 and Table 2 show the learning curves and the performance of PSO and SAPSO with different parameter values.



(a) (b)

Figure 7: Results of the desired output and the model output of (a) the PSO method, and (b) the SAPSO method.

RMSE / Model	RMSE(Ave)	RMSE(Best)
Traditional PSO	0.023	0.011
SAPSO*1*	0.100	0.059
SAPSO*2*	0.068	0.024
SAPSO*3*	0.066	0.035
SAPSO*4*	0.031	0.012
SAPSO*5*	0.038	0.024
SAPSO*6*	0.099	0.057
SAPSO*7*	0.037	0.020
SAPSO*8*	0.119	0.108
SAPSO*9*	**0.016**	**0.012**

Table 2: The performance comparison with two methods.

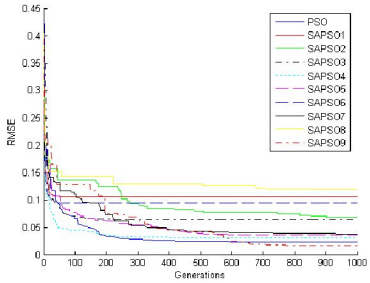


Figure 8: The learning curves of the PSO and the SAPSO with different parameter values.

Example 2-Prediction of the Chaotic Time Series

The Mackey-Glass chaotic time series $x(t)$ in consideration here is generated from the following delay differential equation:

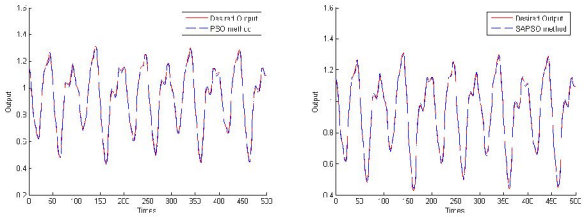
$$\frac{dx(t)}{dt}=\frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)}-0.1x(t) \qquad (29)$$

Crowder [34] extracted 1000 input-output data pairs $\{x, y^d\}$ which consisted of four past values of $x(t)$, i.e.

$$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)] \qquad (30)$$

where τ=17 and $x(0)$=1.2. There are four inputs into the model, corresponding to the values of $x(t)$, and one output representing the value $x(t+\Delta t)$, where Δt is a time prediction into the future. The first 500 pairs (from $x(1)$ to $x(500)$) are the training data set, while the remaining 500 pairs (from $x(501)$ to $x(1000)$) are the testing data set used for validating the proposed method. The number of fuzzy rules is set to 6. The average best RMSE of the prediction output approximates 0.009 after 1000 generations.

Figures 9 (a) and (b) show the prediction results of PSO and SAPSO. Table 3 shows the comparison results of the prediction performance of all methods. Figure 10 shows the RMSE curves of the two models.



(a) (b)

Figure 9: The prediction results of (a) the PSO and (b) the SAPSO.

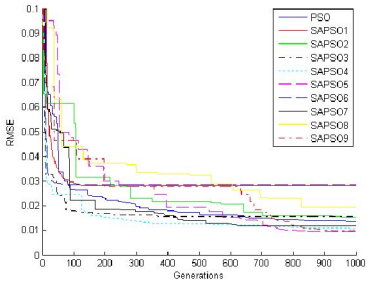


Figure 10: The learning curves of the PSO and the SAPSO with different parameter values for prediction problem.

Model \ RMSE	RMSE(Ave)	RMSE(Best)
Traditional PSO	0.012	0.006
SAPSO*1*	0.025	0.013
SAPSO*2*	0.015	0.010
SAPSO*3*	0.015	0.012
SAPSO*4*	**0.010**	**0.006**
SAPSO*5*	**0.010**	**0.007**
SAPSO*6*	0.029	0.012
SAPSO*7*	**0.011**	**0.008**
SAPSO*8*	0.019	0.010
SAPSO*9*	**0.009**	**0.006**

Table 3: The performance comparison with two methods

Example 3-Approximation of the Piecewise Function

The piecewise function was studied by Zhang [35] and Xu [36] and is defined as:

$$f(x)=\begin{cases} -2.186x-12.864 & -10\le x<-2 \\ 4.246x & -2\le x<0 \\ 10e^{-0.05x-0.5}\sin[(0.03x+0.7)x] & 0\le x\le 10 \end{cases} \quad (31)$$

over the domain D = [-10, 10]. The piecewise function is continuous and can be analyzed. However, traditional analytical tools are inadequate and often fail. This failure may be due to two reasons, namely, the wide-band information hidden at the turning points and the amalgamation of linearity and nonlinearity.

In this example, 200 training input patterns are uniformly generated from Eq. (31). Seven fuzzy rules are generated in this example. The RMSE curve is shown in Fig. 11 with all methods. Figures 12 (a)-(b) show the outputs of the function f with the PSO method and the SAPSO9 method. The solid line represents the output of function f, and the dotted line represents the approximation of various methods. The results comparing our model with PSO are tabulated in Table 4.

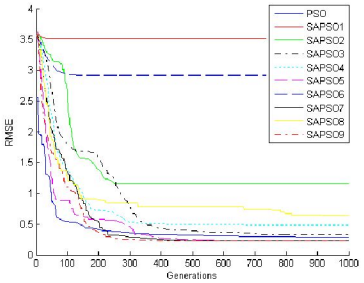
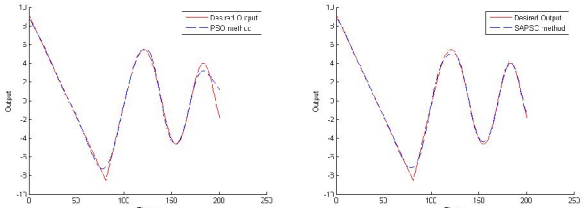


Figure 11: The learning curves of the PSO and the SAPSO with different parameter values for piecewise problem.



(a) (b)

Figure 12: The results of approximation using (a) the PSO method and (b) the SAPSO9 method.

Model \ RMSE	RMSE(Ave)	RMSE(Best)
Traditional PSO	0.28	0.12
SAPSO*1*	3.35	3.15
SAPSO*2*	1.15	0.45
SAPSO*3*	0.33	0.16
SAPSO*4*	0.43	0.14
SAPSO*5*	**0.24**	**0.13**
SAPSO*6*	2.96	2.18
SAPSO*7*	**0.21**	**0.09**
SAPSO*8*	0.64	0.36
SAPSO*9*	**0.20**	**0.09**

Table 4: The performance comparison with two methods.

The average computation time per generation for three examples with the PSO and SAPSO is tabulated in Table 5. We only update the value of sub-particles in each sub-swarm for SAPSO, and the total adjusted parameters of SAPSO are less than that of PSO. Therefore, the computation time required by SAPSO is less than that by PSO.

Example / Model	Identification of Nonlinear Dynamic System	Prediction of the Chaotic Time Series	Approximation of the Piecewise Function
PSO	1.21	7.5	3.15
SAPSO	0.35	1.70	0.65

Table 5: The average computation time of three examples for the PSO and the SAPSO (Unit: sec).

7 Discussion

From the above experimental results, we find that the parameters $\kappa_1, \kappa_2, \kappa_3$ and δ affect the performance of SAPSO. In order to test the relationship between the search trajectory and the parameter δ, we use the same value of κ_1, κ_2, κ_3 and ω in SAPSO6, SAPSO7, SAPSO8, and SAPSO9. We define the collection degree (CD) of a sub-swarm for each generation as follows:

$$CD = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \| particle(i) - particle(j) \| \qquad (32)$$

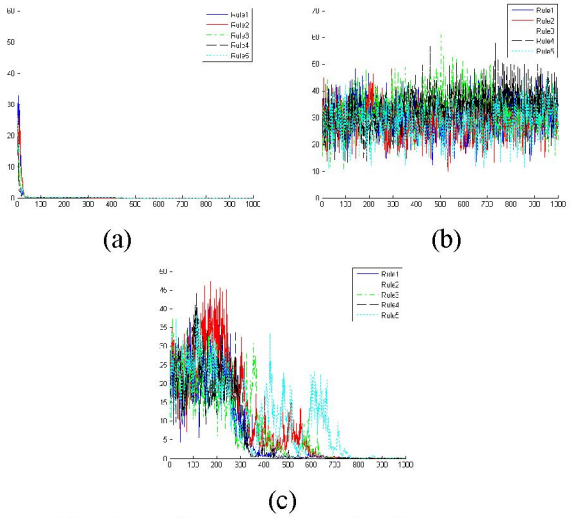
where N is the number of sub-swarms and $\| \|$ is the 2-norm (the Euclidean one for a vector). When CD is small, the particles are close to each other.

Figures 13 (a)-(c) and figures 14 (a)-(c) show the simulation results of example 1 when δ is 0.3, 0.7, and 0.6, diminishing to 0.4 gradually by increasing the learning steps. It is observed that the convergence speed of sub-particles in five sub-swarms is fastest when δ is 0.3, and the dynamics of five sub-swarms are always vibration when δ is 0.7, as seen in Fig. 13 (b).

The two situations mentioned are detrimental to the performance of the system. In Fig. 14 (a), the RMSE curves are sharp in the beginning and do not move afterwards. In Fig. 14 (b), the RMSE curves are like a ladder and drop slowly. Therefore, we hope to combine the advantages of the two situations mentioned and find the appropriate method for setting the parameters. We make the value of δ big enough to increase the chance of search in the beginning and then gradually reduce generations in number. We find that the performance of SAPSO5 and SAPSO9 is better than the others from the experiments. Moreover, from the above experimental results, the performance of the system is better when $\kappa_1 < \kappa_2 \leq \kappa_3$.

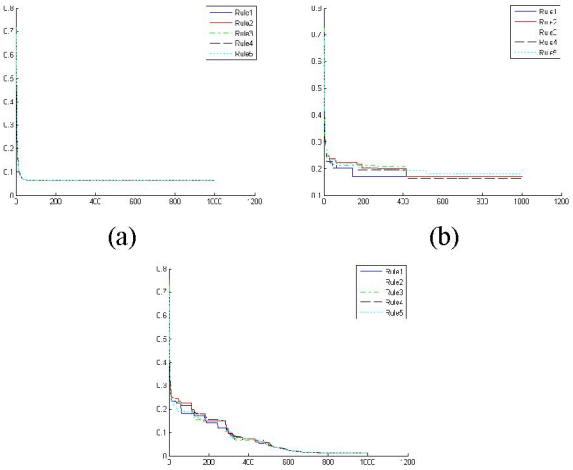
8 Conclusion

In this paper, a novel symbiotic adaptive particle swarm optimization (SAPSO) that adjusts the parameters of fuzzy systems was presented. The proposed SAPSO approach allows control over the dynamic characteristics of particles. The efficiency of the proposed SAPSO was demonstrated by its application to identification, prediction, and approximation of function problems. Simulation results show that the proposed SAPSO has better learning performance and less computation time requirements than the traditional PSO method.



(a) (b) (c)

Figure 13: The collection degree for five sub swarm of example1 in learning processes when δ is (a) 0.3, (b) 0.7, and (c) 0.6~0.4.



(a) (b) (c)

Figure 14: The RMSE curves for five sub-swarm of example1 in learning processes when δ is (a) 0.3, (b) 0.7, and (c) 0.6~0.4.

References

[1] C. T. Lin and C. S. G. Lee (1996), Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System, NJ: Prentice-Hall.

[2] G. G. Towell and J. W. Shavlik (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning*, vol. 13, pp. 71-101.

[3] C. J. Lin and C. T. Lin (1997). An ART-based fuzzy adaptive learning control network. *IEEE Trans. on Fuzzy Systems*, vol. 5, no. 4, pp. 477-496.

[4] L. X. Wang and J. M. Mendel (1992). Generating fuzzy rules by learning from examples. *IEEE Trans. on Systems, Man, Cybern.*, vol. 22, no. 6, pp. 1414-1427.

[5] T. Takagi and M. Sugeno (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Systems, Man, Cybern.*, vol. SMC-15, pp. 116-132.

[6] J.-S. R. Jang (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. on Systems, Man, and Cybern.*, vol. 23, pp. 665-685.

[7] C. F. Juang and C. T. Lin (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Trans. on Fuzzy Systems*, vol. 6, no.1, pp. 12-31.

[8] F. J. Lin, C. H. Lin, and P. H. Shen (2001). Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. *IEEE Trans. on Fuzzy Systems*, vol. 9, no. 5, pp. 751-759.

[9] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima (1992). Neural networks designed on approximate reasoning architecture and their application. *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 752-759.

[10] J. Kennedy and R. Eberhart (1995). Particle swarm optimization. *Proc. IEEE Int'l Conf. Neural Networks*, pp. 1942-1948.

[11] Z. L. Gaing (2004). A particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE Trans. on Energy Conversion*, vol. 19, Issue: 2, pp. 384-391.

[12] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi (2000). A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Trans. on Power Systems*, vol. 15, Issue: 4, pp. 1232-1239.

[13] M. A. Abido (2002). Optimal design of power-system stabilizers using particle swarm optimization. *IEEE Trans. on Energy Conversion,* vol. 17, Issue: 3, pp. 406-413.

[14] C. F. Juang (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 34, Issue: 2, pp. 997-1006.

[15] R. Mendes, P. Cortez, M. Rocha, and J. Neves (2002). Particle swarms for feedforward neural network training. *Proc. Int'l Joint Conf. Neural Networks*, pp. 1895-1899.

[16] A. Savran (2007). an adaptive recurrent fuzzy system for nonlinear identification. *Applied Soft Comp.*, vol. 7, pp. 593-600.

[17] Y. Oysal and S. Yilmaz (2010). an adaptive wavelet network for function learning. *Neural Comp. Appli.*, vol. 19, pp. 383-392.

[18] B. Cetisli and A. Barkana (2010). Speeding up the scaled conjugate gradient algorithm and its application in neuro-fuzzy classifier training. *Soft Comt.*, vol. 14, no. 4, pp. 365-378.

[19] B. Cetisli (2010). Development of an adaptive neuro-fuzzy classifier using linguistic hedges: part 1. *Expert Syst. Appli.*, vol. 37, pp. 6093-6101.

[20] Y. Bodyanskiy, V. Kolodyazhniy, and A. Stephan (2001). An Adaptive Learning Algorithm for a Neuro-fuzzy Network. *Proc. Int'l Conf. 7th Fuzzy Days*, Dortmund, Germany, pp. 68-75.

[21] A. K. Palit and R. Babuska (2001). Efficient Training Algorithm for Takagi-Sugeno Type Neuro-Fuzzy Network. *Proc. IEEE Int'l Conf. Fuzzy Systems*, Vol. 3, pp. 1367-1371.

[22] R. Fletcher (reprinted 2006). *Practical methods of optimization*. Edn. 2nd, West Sussex: Wiley.

[23] P. E. Gill, W. Murray, and M. H. Wright (1981). *Practical Optimization*. London: Academic Press.

[24] R. Storn and K. V. Price (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Opt.*, vol. 11, no. 4, pp. 341-359.

[25] A. E. Douglas (1994). *Symbiotic Interactions*. Oxford: Oxford University Press.

[26] C. J. Lin and C. F. Wu (2009). An Efficient Symbiotic Particle Swarm Optimization for Recurrent Functional Neural Fuzzy Network Design. *International Journal of Fuzzy Systems*, vol. 11, no. 4, pp. 262-271.

[27] C. H. Chen, C. J. Lin, and C. T. Lin (2009), Nonlinear System Control Using Adaptive Neural Fuzzy Networks Based on a Modified Differential Evolution. *IEEE Trans. on Systems, Man, and Cybernetics--Part C: Applications and Reviews,* vol. 39, no. 4, pp. 459-473.

[28] C. H. Chen, C. J. Lin, and C. T. Lin (2009). Using an Efficient Immune Symbiotic Evolution Learning for Compensatory Neuro-Fuzzy Controller. *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 3, pp. 668-682.

[29] P. N. Suganthan (1999). Particle swarm optimizer with neighborhood operator. *Proc. Congress on Evolutionary Computation*, vol. 3.

[30] K. Yasuda, A. Ide, and N. Iwasaki (2003). Adaptive particle swarm optimization. *Proc. IEEE Int'l Conf. Systems, Man and Cybernetics*, vol. 2, pp. 1554-1559.

[31] D. E. Moriarty and R. Miikkulainen (1996). Efficient reinforcement learning through symbiotic evolution. *Mach. Learn.*, vol. 22, pp. 11-32.

[32] M. Clerc and J. Kennedy (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. on Evolutionary Computation*, vol. 6, Issue: 1, pp. 58-73.

[33] K. S. Narendra and K. Parthasarathy (1990), Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 4-27.

[34] R. S. Crowder (1990). Predicting the Mackey-Glass timeseries with cascade-correlation learning. *Proc. of the 1990 Connectionist Models Summer School*, Carnegie Mellon University, pp. 117–123.

[35] Q. Zhang and A. Benveniste (1992). Wavelet Networks. *IEEE Trans. on Neural Networks,* pp. 889-898.

[36] D.W.C. Ho, P. A. Zhang, and J. Xu (2001). Fuzzy Wavelet Networks for Function Learning. *IEEE Trans. on Fuzzy Systems*, vol.9, pp. 200-211.

Order Statistics Bayesian–Mining Agent Modelling for Automated Negotiation

Samir Abdel Rahman, Reem Bahgat and George M. Farag
Computer Science Department,
Faculty of Computers and Information, Cairo University, Egypt
E-mail: {s.abdelrahman, r.bahgat}@fci-cu.edu.eg, gmf.deary@gmail.com

Keywords: bayesian mining, order statistics, automated negotiation, multi-issue, multi-session, opponent modelling

Received: May 27, 2010

The availability of qualitative knowledge has been recently used to simulate human negotiations accurately. During real-life negotiation sessions, people accumulate their knowledge to opt for most adequate bids by which both negotiating parties reach a win-win agreement. Unfortunately, existing research mainly concentrates on few negotiation bids. This paper proposes order statistics Bayesian-mining agent approach to automate bilateral multi-issue multi-session win-win negotiation problems. The proposed agent applies a real-life social bid ranking based on historical bids of all previous negotiation sessions to dynamically update all issues' weights and preferences. Moreover, it uses our proposed deterministic Trade-Off counter offer method, rather than the existing haphazard estimation method, to estimate precisely the next bid. Experiments are conducted on 3-issue, 5-issue, 6-issue and 10-issue having 27, 3169, 3122 and 13219200 bids respectively. The selected evaluation analysis methods are mainly Pareto optimality, utility, cost and step-wise measurements. Compared with existing agent sorts, such as ABMP, Trade-Off, Bayesian and Mining agents, the proposed agent approach is proved that it is more efficient, effective, scalable and sensitive (adaptable to the opponent steps). Also, it works better to maximize its utilities and to minimize the negotiation costs (the number of rounds).

Povzetek: Opisana je agentna metoda pogajanj, ki se odloča na osnovi Bayesovske statistike.

1 Introduction

The paper aims to automate bilateral multi-issue multi-session win-win human negotiation. Negotiation is the process in which two or more parties, having conflicts in their interests, can mutually reach a beneficial agreement on the related set of issues by exchanging some bids. In any bilateral negotiation [20], there are only two parties who exchange their bids using some negotiation protocols. In multi-issue negotiations [20], each bid has many issues such that each issue consists of several discrete items. The negotiator goal is to adjust all issues' preferences to maximize his bid utility. The essential assumption of win-win multi-issue bilateral negotiation is that the two negotiators are rational and they are eager to find a solution of bid utilities that is acceptable to both parties [19]. So, each party has to know the preferences of its opponent to reach an agreement. In reality, the negotiators are hardly willing to disclose their private preferences. Consequently, both parties have incomplete information about each other and so may hardly reach an optimal deal. A typical real-life negotiation may need more than one session to reach a successful deal such that each party commences the new session having some gained knowledge about his opponent from previous sessions, where a session is defined as the time duration in which the negotiators decide to communicate with each other to reach a satisfied agreement, if any.

Automated negotiation recently has become a disputed solution to compensate the human disability to do complicated negotiation calculations accurately. Automated negotiation applications range from simple auctions, in which agents merely have to bid truthfully [29], to complex strategic models, in which agents argue for positions and aim to persuade their opponents of the particular course of action [24].

Some multi-agent research [2, 9, 12] presented strategies to automate multi-issue negotiation. Such research was usually motivated with the high complexity of multi-issue negotiation calculations executed with the lack of available knowledge about the opponent. Recently, some research [7, 12] investigated the use of available knowledge about the environment's bids, issues, or opponent preferences while negotiation sessions move forward. However, they either concentrated on cases having one issue or they depended on few bids from previous/current single session negotiation. Moreover, their assumptions of issue ranking and the data distribution are theoretical rather than experimental; real-life applications are complex in which the agreement is met after successive negotiations having massive hidden information about the environment's preferences and issues that could be mined, i.e. extracted and accumulated, while the negotiation sessions ensue.

This paper presents non-parametric Bayesian-mining agent modelling that depends on all historical successive sessions to solve the complexity of real-life applications. It proposes the following two crucial ideas.

First, while negotiation sessions advance, the

proposed model gradually learns how to reduce the number of session rounds and to maximize the expected utility upon the ratio of accepted bids. To do that, the model weighs the bids, similar to the human ranking, by which the first bid from each session is the most significant bid and the current session is the most vital session. The model then utilizes order statistics, non-parametric, Bayesian learning to model the opponent preferences and profiles which deals with any unknown data distribution.

Second, a proposed trade-off counter-offer method is used to estimate the next bid more precisely. This is done by replacing existing randomness trade-off method with a proposed partial derivative utility function.

Our experiments are set up against some well-known existing agent approaches using 3-issue, 5-issue, 6-issue and 10-issue applications. We use some evaluation criteria, namely Pareto optimality, utility, cost (number of rounds), step-wise (sensitivity analysis and class studies), and confidence interval calculations. It is experimentally proved that any agent following our proposed model is efficient, effective and sensitive to the opponent steps. Also, the proposed agent scalability is verified as the agent guarantees these features on 10-issue applications.

In comparison with current negotiation approaches, the contributions of the proposed non-parametric Bayesian-mining approach are as follows:

1-It works with any negotiation data distribution; all current approaches assume normal distribution of data, which is not necessarily true.

2-The agent outcomes are more effective and sensitive as the agent can benefit from the historical data of all previous negotiation sessions.

3-When our agent is involved in the negotiation, better agreement is reached fast.

4-When our agent is involved in the negotiation, both negotiators tend to maximize their utilities.

5- Our approach is scalable for large data set, while authors of the other approaches state that they have to adapt their models, if possible, to make them acquire such a feature.

The remainder of this paper is organized as follows. The next section discusses other negotiation approaches. Section 3 outlines the evaluation methods stated in the literature. Section 4 presents the overall proposed approach with its assumptions and parameters. Section 5 demonstrates the proposed Bayesian-mining approach. Section 6 presents the proposed counter-offer enhancement. Section 7 shows the experimental results. The paper is concluded with Section 8.

2 Related work

ABMP strategy [15, 16] takes the agent's own utility space in which the next bid utility has less value than the previous one. Unfortunately, the strategy does not use any domain or opponent knowledge. Also it does not search through the negotiation outcome space for results that are mutually beneficial for both parties. Therefore, this strategy is inefficient in complex negotiation domains although it is shown that it outperforms humans in small domains [1].

Trade-off strategy [9] is based on similarity and iso-curve criteria. In this strategy, the agent tries to find a bid similar to his previously proposed one and to be simultaneously suitable for his opponent. However the random nature of its search impacts on its efficiency. Another disadvantage is its difficulty to determine the bid suitability for the opponent's utility without having any knowledge about his preferences. So, it always needs a complement strategy to detect the opponent preferences.

Bazaar model [33] is a learning approach for sequential decision making in a single session single issue (the price) negotiation. It works by generating random numbers of upper and lower limits for the agent's reservation price to ensure the existence of agreement zone. However, the negotiation model is dedicated only to the price issue which is already known earlier to both agents.

A Bayesian Markov chain model [21] was presented to learn the opponent preferences in single issue negotiation. Its major defect is that it does not have a state-memory to save all negotiation movements since it depends only on the negotiation current state to predict the future bids. Moreover, it works only on a single issue.

Kernel Density Estimator model [7], based on [9], provided a kernel estimation method that depends on the difference between two bids to predict the issue weight. Therefore, the estimator doesn't use its whole available negotiation history to define its kernel. Moreover, it does not provide a learning method for estimating the issue weights, hence, it may be used effectively only with single issue negotiations.

A Bayesian learning modelling [12] was presented to learn an opponent model, i.e. the issue preferences and priorities of the opponent. Unfortunately, the model uses single session only to know the opponent preferences. In most cases, the session has few bids to learn and thus the gained knowledge is often imperfect. Also, it enumerates all possible issue-weight ranks to form the weighted issue hypothesis space which makes the space considerably huge. Moreover, as most strategies do, it assumingly considers the negotiation data to follow the normal distribution which contradicts with real human negotiations.

[18] proposed a theoretical means to acquire negotiation knowledge from a batch of previous bids in previous sessions of negotiations within the e-Marketplace field. The model gives weights for each issue and related items based on the accept/reject probabilities. Then it sums these probabilities to weigh the related bid and finally it ranks the bids' weights to select the bid with the highest weight given that it was not previously selected. The authors report that they theoretically open the door for mining negotiation research. Unfortunately, the issue weight calculation doesn't consider the shape of issue evaluation function which yields some wrong bid selections. Moreover, the selected bid may not be the optimum choice for both

buyers.

As shown above, current automated negotiation strategies don't exploit all historical bids to enhance the negotiation outcomes. Moreover, they follow some theoretical assumptions which make the negotiation relatively far from reality.

We compare our work with the above mentioned approaches to prove our approach's efficiency, effectiveness, sensitivity to the opponent steps, and scalability. Fortunately, some research, such as [12, 23], tried to prove the scalability of their approaches. They report that it is not easy for any model to sustain high dimensional specifications. The authors of Bayesian Learning model [12] show that they modify their model to make it scalable for 10-issue applications and when compared with the Trade-Off agent, both performances were similar to each other as the agents stay close to the Pareto frontier. In [23], a system of artificial adaptive agents (AAAs), is created and tested for 10-issue against the human agents to evaluate their performance. The authors find that in high dimensions, such as 10-issue, it is very complicated to make a suitable comparison of the behavior and the performance between AAA and human agents since neither the two agents may outperform each other.

Before presenting the proposed framework strategy (Sections 4, 5 and 6), it is also worth exploring the real-life win-win bilateral game strategy. Three cases are possible. The first case is when both players are unprofessional or not able to accumulate knowledge about each other, then the game outcomes are weak. The second case is when one of them is skilful to gain experience from his opponent tactics, then he outperforms his opponent even if he commences the game weakly. Moreover, the game outcomes are relatively high. The last case is when both players are professional, i.e. they can benefit or learn knowledge about the tactics of each other, which may positively affect both of them on the general results of the game. If such game is negotiation, then people benefit from the past/current sessions to achieve the best, even if they have little initial data about their opponents. People always give the highest priority to the first bid in each session and lower priorities to successive bids. Also, they give weights to previous sessions and their bids but the weights reduce as the sessions are further away from the current session.

3 Evaluation methods

Most existing automated negotiation approaches consider mainly Pareto optimality [6, 8, 27], utility [17, 31], cost (number of rounds) [33] and step-wise (sensitivity analysis and class studies) [13] criteria as their evaluation methods. They consider the negotiation of an agent efficient if its outcomes have rapidly reached the Pareto Frontier with the maximum utilization and the minimum number of rounds. Also, they prove the agent effectiveness using Pareto optimality and sensitivity analysis. People, then, apply sensitivity analysis and class studies to test the negotiator adaptability to the opponent preferences. We use all these methods to evaluate our proposed model.

3.1 Pareto Optimality

This method is to measure the distance of the negotiation outcome to the Pareto Frontier. A deal is Pareto optimal (or Pareto efficient), if it is not dominated by any other deal. In other words, a deal is Pareto optimal if it is the best agreement among all negotiation agents. When the deal is Pareto optimal, the negotiation should end with such an agreement [17, 31].

3.2 Cost Analysis

The cost of a negotiation process is measured by the number of proposals (rounds) exchanged before reaching an agreement [33]. The agent is efficient if it does fewer rounds to reach optimum agreement with its opponent.

3.3 Utility Analysis

An automated negotiation strategy should guarantee its agent to reach the maximum utilities when the agreement deal is committed. An efficient negotiation strategy models its agent such that it swiftly increases the outcomes while the negotiation sessions advance.

3.4 Sensitivity Analysis

In the sensitivity analysis [13], not only the study of negotiation outcomes is essential, but also the investigation of the agent faults which are realized throughout the negotiation activities. Another useful sensitivity analysis of the opponent preferences is to dynamically find out the negotiation characterizations. It can be defined by comparing the percentage (Equation 1) of fortunate, nice and concession steps that increases the opponent's utility to the percentage of selfish, unfortunate and silent steps that decreases it. Spontaneously, an agent which only achieves steps increasing its opponent utility can be said to be sensitive to its opponent requirements.

A single negotiation step between an agent current bid and the previous one for that agent which is written as [13] based on utilities as follows:

$$\Delta_a(s)=U_a(b')-U_a(b) \quad (1)$$

For a step $s = b_a \rightarrow b'_a$, $a \in \{S, O\}$, for the Agent S and its Opponent O, to denote the utility difference of two bids b and b' in the utility space of agent A. From the point of view of the agent S, the negotiation step s is classified as [13]:

- *Fortunate Step*, denoted by (S+, O+), iff: $\Delta_S(s)>0$, and $\Delta_O(s)>0$.
- *Selfish Step*, denoted by (S+, $O_{\leq}$), iff: $\Delta_S(s)>0$, and $\Delta_O(s)\leq 0$.
- *Concession Step*, denoted by (S-, $O_{\geq}$), iff: $\Delta_S(s)<0$, and $\Delta_O(s)\geq 0$.
- *Unfortunate Step*, denoted by ($S_{\leq}$, O-), iff: $\Delta_S(s) \leq 0$, and $\Delta_O(s) <0$.

- *Nice Step*, denoted by (S=, O+), iff: $\Delta_S(s)=0$, and $\Delta_O(s)>0$.
- *Silent Step*, denoted by (S=, O=), iff: $\Delta_S(s)=0$, and $\Delta_O(s)=0$.

The measure for sensitivity of the agent A (Equation 2) to its opponent's preferences is defined for a given trace t [13] which includes all session bids for both agents.

$$sensitivity_a(t) = \left(\frac{\left(\left(\%_{Fortunate}(t_a)\right)+\left(\%_{Nice}(t_a)\right)+\left(\%_{concession}(t_a)\right)\right)}{\left(\left(\%_{Selfish}(t_a)\right)+\left(\%_{Unfortunate}(t_a)\right)+\left(\%_{Silent}(t_a)\right)\right)} \right) \quad (2)$$

If $sensitivity_a(t)<1$, then an agent is more or less insensitive to opponent preferences. If $sensitivity_a(t)>1$, then an agent is more or less sensitive to the opponent's preferences. The sensitivity notion is *asymmetric*, i.e. one agent may be sensitive to the other's preferences, but not vice-versa.

3.5 Class Studies

Given a trace (session offers) $t = \langle b_s^1, b_o^2, b_s^3 \ldots \rangle$ of offers, t^i denotes the i^{th} element of this sequence, $t_s(t_o)$ denotes the sequence of steps from t that are made by the agent himself (opponent), t_c denotes the subsequent steps that belong to a class c and finally $t_{\langle a,c \rangle}$, written t_{ac}, denotes the subsequent steps by $a \in \{S,O\}$ that belong to class c; where the class $c \in$ *{Fortunate, Nice, Concession, Selfish, Unfortunate, Silent}* (Section 3.4). In this research, we are interested in two essential metric measures namely:

- **Total utility difference per class**

The pair $Total_c(t)$ of sums of utility differences in all steps of class c in a sequence t of steps is defined by:

$$Total_c(t) = Total_{Sc}(t) \quad (3)$$

Where for any agent

$$a \in \{S,O\}: Total_{ac}(t) = \sum_i \Delta_a(t_c^i)$$

- **Average Utility Difference per Class**

The pair $u\text{-}Ave_c(t)$ of average differences in utility in all steps in class c in a sequence t of steps is defined by:

$$u\text{-}Ave_c(t) = u\text{-}Ave_{Sc}(t) \quad (4)$$

Where for any agent

$$a \in \{S,O\}: Total_{ac}(t) = \sum_i \Delta_a(t_c^i)/\#t_c$$

Here $\#t_c$ is the number of steps of class c in trace t.

This metric measures the average utility conceded per negotiation step [8]. Negotiation strategies could be observable as negotiation dance patterns. For example, the success of a strategy that is supposed to learn its opponent's preferences can be verified by checking whether the frequency and/or the size of unfortunate steps over a negotiation trace decreases. Such patterns can be seen as a measure of adaptability of a party to its opponent.

4 The proposed approach

The proposed framework handles a bilateral multi-issue multi-session win-win negotiation (Section 1); the agents are rational to be involved in such negotiation. All negotiating agents work independently to maximize their utilities such that all of them win. The negotiation bids are independent which means that the values of bid preferences and issues are not derived from the other bid values. One essential assumption is that the data distribution is unknown. Two other crucial assumptions are related to bids' ranking and bid selection (Section 5). Finally, the model utility function is assumed to be a linear summation (Equation 5)

$$U = \sum_{i=1}^{n} w_i \; e_i \; [8, 12] \quad (5)$$

Where w_i is the issue weight and e_i is the issue evaluation function; $e_i \in [0,1]$.

Beside the model utility function, the main model equations are issue hypothesis space function (Equation 13) and order statistics Bayesian-mining conditional probability function (Equation 14). Using the evaluation methods (Section 3) to test the proposed model, the following steps are orderly done:

1. The model utility function is calculated.
2. Using the above assumptions, bid weighted issues W is calculated (Sections 5.2 and 5.3).
3. Issue hypothesis space (Equation 13) of all bids is defined as the Cartesian product of the shapes of the issue evaluation functions e_i (Equation 5) and W (Step 2).
4. At each bid arrival, the prior probability $P(H_j); H_j \in H$ is updated using Bayesian-Mining Learning approach (Section 5.4) and the Bayesian rule of historical bids order statistics (Equation 14).
5. The proposed counter-offer b_{t+1} (Equation 22) using expected utility $\bar{u}(b_t)$ (Equation 21) is updated using (Equations 20 and 5).

Throughout the Steps (2-5), all possible bids transactions are recorded including session *id*, bid <issues, items, weights, utility, rank>, and opponent acceptance status (Yes/No).

5 Mining the opponent profiles

The order statistics Bayesian-mining strategy is to minimize the number of session rounds of an agent as well as to maximize its utility. The agent rationality is increased whilst the negotiation sessions advance. Also, it is proportionally boosted with the opponent rationality. The opponent results depend on its preferences and behaviour. Fortunately, any opponent always gains from playing against the proposed agent since the game session rounds are extremely diminished, hence, the agreement is reached faster.

In order to apply skilfully its strategy, the proposed agent should have twofold essential properties. First, it mines all historical data from all previous sessions regardless of the underlying data distribution. Second, it uses order statistics Bayesian-mining approach to learn successfully the opponent profiles/preferences and to make good weight estimation for all negotiation issues based on the proposed ranking and bid weighing.

5.1 Ranking Bids

In a multi-session negotiation, the current session is considered the most important session. All current (last) session bids take higher ranks than other previous sessions bids. Also, each session first bid is considered the most important one [22] so it takes higher rank than the consequent bids in the same session. Thus the bids are proposed orderly in sequence such that no bid is proposed twice in the same session. The session ranks (the first bid in each session) r_i^s is assigned and to a negotiation session i according to:

$$r_i^s = \text{int}\left(e^{|session|+x}\right)^y \quad (6)$$

Where x, y is user defined according to the importance of the session, taking a reasonable x for the starting session ranking; in this research y=1, x=3.

In addition, the rank of each bid j, r_{ij}, inside the session is ranked in sequential order as follows:

$$r_{ij} = r_i - (j-1) \times \frac{r_i^s - r_{i-1}^s}{|N_i|} \quad [18] \quad (7)$$

Where $|N_i|$ is the total number of bids in the negotiation session i.

5.2 Bid Selection

All framework agents are assumed to be rational such that the agent selects the bid once to allow win-win situation for both negotiating parties. Through the session activities, the utilities of bids are calculated to gauge the bid acceptance/rejection status. Hence, if the proposed bid utility is less than or equal to the opponent expected utility, then the agent has to make an agreement and the related session is accordingly stopped. Otherwise, the agent rejects the opponent bid and the agent bid is proposed.

5.3 Bid Weighing

As follows, the weight factor can be estimated using the historical sequence of bids, each bid items, the status of the opponent acceptance, and the rank of each bid.

$$P(accept) = \sum bid\ ranks\ (opponent\ accept = YES) / N \quad (8)$$

$$P(reject) = \sum bid\ ranks(opponent\ accept = NO) / N \quad (9)$$

Where N is the summation of all ranks. Thus the weight of each item in the issues can be estimated using [30, 5] as follows:

$$w_i = P(item(i)|accept) = \sum bid\ ranks\ containing\ item\ i\ of\ (opponent\ accept = YES)\ /\ total\ ranks\ of\ accepted\ bids\ in\ all\ sessions\ for\ all\ issues. \quad (10)$$

If each issue is composed of more than one item, then the bigger item weight is considered as the issue weight with the normalization as follows:

$$w_i = w_i / \sum_{i=1}^{n} w_i \quad (11)$$

$$W = (w_i)_{i=1}^{n} \; ; \sum_{i=1}^{n} w_i = 1 \quad (12)$$

Where n is the number of issues. Then, the remaining item set in each issue is adjusted.

5.4 Issue Hypothesis Space

[12] utilizes the issue weighted hypothesis space H as a Cartesian Product of $H^w \times H_1^e \times H_2^e \times ... \times H_n^e$. H^w presents all combinations of issues' weights related to each possible issue ranks and H_i^e is the shape of issue evaluation function; the function shape may be downhill, uphill, or triangular. However, since H^w is calculated based on all enumerations of issue weights and the related ranks, its size is relatively huge.

Fortunately, the proposed weighting issue mechanism (Section 5.3) estimates the issue weights based on the accepted ranked bids and hence the combinations of H^w weights are limited to the estimated issue weights W (Equation 12). Therefore, the size of the proposed H (Equation 13) becomes smaller, the number of session bids is reduced, and the bid acceptance likelihood is increased.

$$H = W \times H_1^e \times H_2^e \times ... \times H_n^e \quad (13)$$

5.5 Bayesian-Mining Opponent Modelling

In the proposed Bayesian-Mining approach, it is needed to update the probability associated with all hypotheses given to the new bid, i.e. the posterior probability given by Eq.14. In reality, the negotiation information about bids may be too imperfect or hidden to be easily estimated, i.e. there is no specific distribution for the historical negotiation data. So, it is decided to utilize the nonparametric and order statistics techniques.

Let b_{t-n}, b_{t-n+1}, $b_{t-n+2,...}$, b_t denote the order statistics of size n of a total number of opponent bids; the utilities of these bids are previously estimated. Let H be the class in which some of these utilities exist. In the proposed Bayesian approach, the most likely class value H_j is the one that maximizes Equation 14:

$$P\left(H_j \middle| (u(b_{t-n}),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t))\right) = \frac{P(H_j)P\left((u(b_{t-n}),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t)) \middle| H_j\right)}{P(u(b_{t-n},),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t))} \quad (14)$$

Using similar thinking as presented in [10], where the conditional probability [14] is:

$$P\left(\left((u(b_{t-n},),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t))\middle|H_j\right)\right) = P\left(\left(u(b_{t-n})\middle|H_j\right)\right)\times P\left(\left(u(b_{t-n+1})\middle|H_j\right)\right)\times ...\times P\left(\left(u(b_t)\middle|H_j\right)\right) \quad (15)$$

Equation 14 dominator could be calculated from Equation 16, which is the joint probability density function using the equation used in [4, 10, 14] and defined as follows

$$P(u(b_i),u(b_j))= \frac{n!}{(i-1)!(j-i-1)!(n-j)!}\times (u(b_i))_i^{i-1}\left(u(b_j)-u(b_i)\right)^{j-i-1}\left(1-u(b_j)\right)^{n-j}, \quad (16)$$

$$0<u(b_j)<u(b_i)<1$$

$$=0 \ elsewhere$$

The conditional probability $P\left(\left(u(b_t)\middle|H_j\right)\right)$ (Equation 15) can be estimated effectively using M-estimate approach [26] as follows:

$$P\left(\left(u(b_t)\middle|H_j\right)\right)=\frac{n_c+m\,p}{n+m} \quad (17)$$

Where n_c is the number of transactions from class H_j that takes the value $u(b_t)$; n is the total number of transactions from class H_j. p is a user-defined parameter and can be computed as the prior probability $p(u(b_t))$. m is a parameter known as the equivalent sample size and it determines the trade-off between the prior probability p and the observed probability n_c/n [26]; the parameter m is set to 2.0 (this setting is usually used as a default and experimentally it gives satisfactory results) [32].

$u(b_t)$ is estimated as $p(u(b_t))$ (Equation 16) which could be calculated as follows using the equation used in [4, 14] :

$$P(u(b_t)_{accepted(b_t))}= \frac{\left(P(accept)\times\prod P(I_{ij}\mid accept)\right)}{\left[(P(accept)\times\prod P(I_{ij}\mid accept))+(P(reject)\times\prod P(I_{ij}\mid reject))\right]} \quad (18)$$

Where I_{ij} is the item of the issue I_i, $P(accept)$ calculated by (Equation 8), and $P(I_{ij}\mid accept)$ calculated by (Equation 10).

$P\left(H_j \middle| u(b_{t-n}),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t)\right)$ normalization [21, 25] is as follows:

$$P\left(H_j \middle| (u(b_{t-n}),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t))\right) = \frac{P(H_j)P\left((u(b_{t-n}),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t)) \middle| H_j\right)}{\sum_{k=1}^{c}P(H_k)P\left((u(b_{t-n}),u(b_{t-n+1}),u(b_{t-n+2}),...,u(b_t)) \middle| H_k\right)} \quad (19)$$

$P(H_j)$ is updated proportional to Equation 8 to get:

$$\sum_{j=1}^{|H|}P(H_j) \quad (20)$$

The expected utility $\overline{u}(b_t)$ *is updated* when the current opponent counter-offer is proposed during the negotiation process, as follows, using Equation 5:

$$\overline{u}(b_t)=\left(\sum_{j=1}^{|H|}P(H_j)\right)U \quad \text{[12, 21]} \quad (21)$$

Where

- $H_j \in H$ is a hypothesis, and b_t is the new bid.
- $P(H_j)$ is the *prior probability* of H_j: the probability that H_j is correct before the new bid b_t is seen or it is the current probability of hypothesis H_j [12].
- $P\left(\left(u(b_t)\middle|H_j\right)\right)$ is the conditional (likelihood) probability of the new bid b_t that its utility might be proposed given that the hypothesis H_j is true.
- $p(u(b_t))$ is the marginal probability of $u(b_t)$

6 The proposed counter-offer

It is assumed that any agent starts any negotiation session by proposing the offer (bid) which has the maximum utility for his owner. The opponent can accept the proposed offer if the utility of that proposal is higher than the offer he last proposed or the offer he intends to propose, else he rejects and proposes a counter-offer. The trade-off algorithm [9], based on the iso-curve concept, starts at the opponent's last bid with a utility $\overline{u}(b_t)$ (Equation 21); where the process is performed in S steps and E is the utility difference between steps. In each step, N children are generated which is closer to the agents iso-curve with small tolerance $\pm\delta$, the most similar to the opponent's bids is selected as a stating bid for another step until the S steps are completed. This last selected one is sent to the opponent. Thus the counter-offer is estimated as follows:

$$b_{t+1} = \underset{b\in\{x\mid\ |u_{own}(x)-u_{t\,\arg et}|\le\delta\}}{\arg\max}\ \overline{u}(b) \text{ [9]} \quad (22)$$

This is similar to what was mentioned in [12, 9]. However, in [9] the children are generated by distributing the utility gain randomly among the issues under negotiation as being mentioned in the algorithm [9] line (5):

$$r_i = \min(random(\overline{E}_i), \frac{E-E_n}{w_i}) \text{ [9]} \quad (23)$$

Where E_i is the maximum evaluation gain for the issue i at this step and E_n is the total amount of consumed utility. $\frac{E-E_n}{w_i}$ is used to limit the final gain to E .

The weak point in this algorithm is that it may increase the utility of a certain issue that has less effect on the opponent utility due to randomization.

Thus it needs to insert a third item to ensure that the increased utility E is distributed fairly among the issues, and to increase the search effectiveness by performing a more directed search for the children at each step in the direction that causes the smallest amount of satisfaction loss to the opponent while increasing the agent's own utility. To do so, the proposed solution is as follows:

Consider a vector $v \in I; v = \{v_i \mid i = 1..j;\ j < n\}$, the set of issues under negotiation, where n is the total number of issues, and v_i is computed by normalizing the partial derivatives $\frac{\partial U(b_t)}{\partial I_i}$. Thus, the proposed enhancement to (Equation 23) is written as:

$$r_i = \min\left(random\left(\frac{E_i}{v_i}\right), (\overline{E_i}), \frac{E - E_n}{w_i}\right) \quad (24)$$

This equation is repeated for all issues similar to the algorithm in [9].

7 Experimental work

The experimental environment is built as follows. First, all the previously mentioned aspects (Sections 3, 4 and 5) are implemented. Second, four agent types are selected and implemented to compete namely, ABMP (A) [15, 16], Trade-Off (T) [9], Bayesian (B) [12], Mining (M) [18] and the proposed Bayesian-Mining (BM) Agent. Finally, three data sets are generated with 3-issue [33], 5-issue [12], 6-issue[1] and 10-issue [8] having respectively 27, 3169, 3122 and 13219200 bids. For each data set, the following bilateral experiments (Opponent vs. Me)/(A vs. B) are carried out:

- Single-Session Experiments
  - ABMP vs. ABMP
  - ABMP vs. Trade-Off
  - ABMP vs. Bayesian
  - ABMP vs. Mining
  - Trade-Off vs. ABMP
  - Trade-Off vs. Trade-Off
  - Trade-Off vs. Bayesian
  - Trade-Off vs. Mining
  - Bayesian vs. ABMP
  - Bayesian vs. Trade-Off
  - Bayesian vs. Bayesian
  - Bayesian vs. Mining
  - Mining vs. Mining
  - ABMP vs. Bayesian-Mining
  - Trade-Off vs. Bayesian-Mining
  - Bayesian vs. Bayesian-Mining
  - Mining vs. Bayesian-Mining
  - Bayesian-Mining vs. Bayesian-Mining
- Multi-Session Experiments(1, 3, and 5 sessions)
  - ABMP vs. Mining
  - Trade-Off vs. Mining
  - Bayesian vs. Mining
  - Mining vs. Mining
  - ABMP vs. Bayesian-Mining
  - Trade-Off vs. Bayesian-Mining
  - Bayesian vs. Bayesian-Mining
  - Mining vs. Bayesian-Mining
  - Bayesian-Mining vs. Bayesian-Mining

[1] We use the online-site: http://interneg.concordia.ca/

7.1 Enhanced Trade-Off Experiments

(3-issue) (1 session)	Enhanced-TRADE-OFF	Basic TRADE-OFF
Mean	0.658	0.510
std dev.	0.080	0.087
Confidence	0.070	0.077
confidence interval 1	0.728	0.587
confidence interval2	0.588	0.4328

Table 1: Statistical Results for 3-issues

(5-issue) (1 session)	Enhanced-TRADE-OFF	Basic TRADE-OFF
Mean	0.4132	0.2168
std dev.	0.1084	0.1200
Confidence	0.0487	0.0539
confidence interval 1	0.4620	0.2708
confidence interval2	0.3645	0.1628

Table 2: Statistical Results for 5-issues

(6-issue) (1 session)	Enhanced-TRADE-OFF	Basic TRADE-OFF
Mean	0.560	0.421
std dev.	0.145	0.155
Confidence	0.079	0.096
confidence interval 1	0.639	0.517
confidence interval2	0.481	0.325

Table 3: Statistical Results for 6-issues

(10-issue) (1 session)	Enhanced-TRADE-OFF	Basic TRADE-OFF
Mean	0.560	0.421
std dev.	0.145	0.155
Confidence	0.079	0.096
confidence interval 1	0.639	0.517
confidence interval2	0.481	0.325

Table 4: Statistical Results For 10-issues

Sensitivity	Enhanced-TRADE-OFF	Basic TRADE-OFF
3-issue	2	1.5
5-issue	1.5	0.923077
6-issue	1.06666	0.380952
10-issue	3.083333	2.466667

Table 5: Sensitivity for The Basic/Enhanced Trade-Off

	Agent	Trade-off vs. Trade-off	Trade-off vs. Enhanced trade-off	Performance increased
3-issue	A	0.6733	0.7566	12.376%
	B	0.7	0.76	8.51%
5-issue	A	0.555	0.6675	20.27%
	B	0.81	0.8125	0.309%
6-issue	A	0.82	0.85	3.659%
	B	0.861	0.8656	0.455%
10-issue	A	0.703	0.707	0.624%
	B	0.6436	0.7345	14.126%

Table 6: Performance for Both Algorithms for All issues

In order to weigh the enhanced trade-off contribution on the negotiation process, experiments are done on the mentioned data to compare both the enhanced trade-off and the basic trade-off algorithms (Tables 1, 2, 3 and 4) for (3-issue, 5-issue, 6-issue and 10-issue) respectively. 95% confidence interval is used and the increase in the confidence interval is found that it is between 24%-36%. It is noticed that the standard deviation, the data population variability measure and the confidence intervals, of the enhanced algorithm is smaller than the basic algorithm standard deviation. This means that the data is spread in smaller range of values leading to less marginal error (confidence). It is found that the agreement offers become more closely to the Pareto frontier raising the utilities for both the agent and his opponent (Table 6).

In 3-issue experiments, while the performance (the utility) of the agent is increased by 8.571%, it is increased by 12.376% for the opponent. In 5-issue experiments, while the performance of the agent is increased by 0.309%, it is increased by 20.270% for the opponent. In 6-issue experiments, while the performance of the agent is increased by 0.455%, it is increased by 3.659% for the opponent. In 10-issue experiments, while the performance of the agent is increased by 14.126%, it is increased by 0.624% for the opponent. Also the enhanced algorithm is more sensitive than the basic algorithm (Table 5).

7.2 Bayesian Mining Experiments

Experiments were run based on 3-issue, 5-issue, 6-issue, and to test the scalability of the approach, 10-issue is used. To compare the performance of the Bayesian mining approach, the agents using opponent modelling were compared with agents using the ABMP, Trade-off, Bayesian and mining strategies. All agents played against the same opponent to compare both negotiation trace (intra-transaction) and the final agreement (inter-transaction). Negotiation takes place between agents *A* and *B* assuming that the latter is the experiment target.

7.2.1 Pareto analysis

The main objective for any automated negotiation is to stay as close as possible to the Pareto efficient frontier. However in current automated negotiation strategies, no player has prior information about the preferences of the negotiating parties, and so all players don't know where the Pareto efficient frontier is located. It thus remains a challenge to stay close to that Frontier. In this research, the Bayesian–mining approach tries to predict the opponent preferences and to select a suitable bid near the Pareto Frontier. Figures 1, 2, 3 and 4 conclude that the Bayesian-mining approach often makes the best prediction to the opponent preferences compared with other strategies; hence, it selects the bids which are preferable to the opponent reaching an agreement close to the Pareto frontier. It may also be concluded that the Bayesian-mining approach gets the shortest distance between the final agreement and the Pareto Efficient Frontier. This is because the accumulated knowledge regarding the opponent behaviour and preferences shortens the distance between the final agreement and the Pareto Efficient Frontier. In 3-issue experiments (Figure 1), after 5 sessions, when the agent B applies Bayesian–mining strategy to negotiate with the opponent A following the same strategy, Bayesian, Trade-off, Mining or ABMP, it gets the distances of the final agreement to the Pareto Frontier equal to 0.020, 0.192, 0.192, 0.170 or 0.209 respectively. Compared with these results, when a Mining strategy agent has opponents, Bayesian, Trade-off, Mining or ABMP, its distances would be 0.2618, 0.1828, 0.3753 or 0.261 respectively. Also, in 10-issue experiments (Figure 4), 5 sessions, when the agent B, having our proposed strategy, negotiates with its opponent A which follows the same strategy, Bayesian, Trade-off, Mining or ABMP, the distances of the final agreement to the Pareto Frontier are 0.009, 0.028, 0.072, 0.042, 0127 respectively. Comparing these results with the agent using Mining strategy having the opponents, Bayesian, Trade-off, Mining or ABMP, the distances become 0.144, 0.164, 0.1266 or 0.129 respectively.

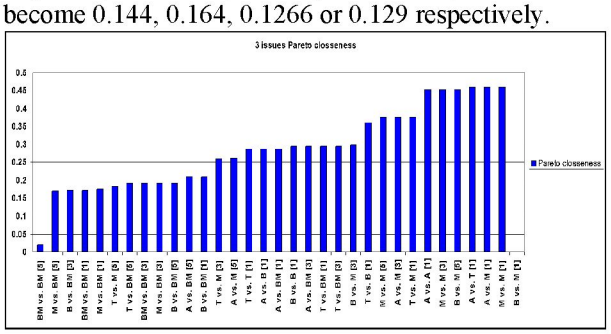


Figure 1:3-issue Pareto Frontier Closeness Outcomes

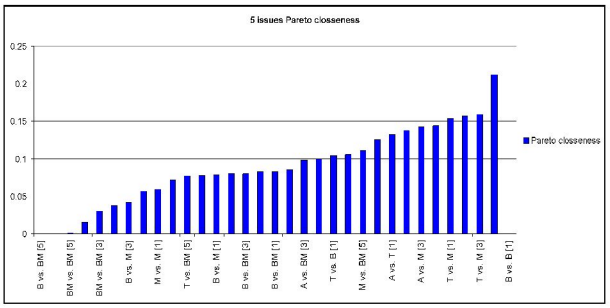


Figure 2:5-issue Pareto Frontier Closeness Outcomes

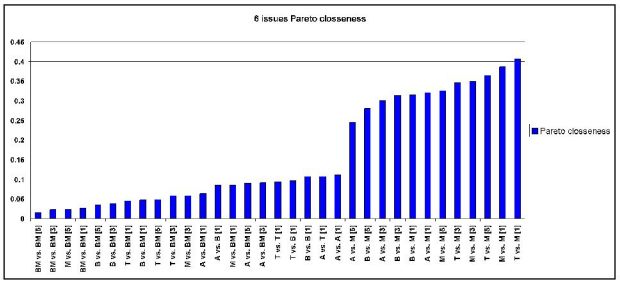


Figure 3:6-issue Pareto Frontier Closeness Outcomes

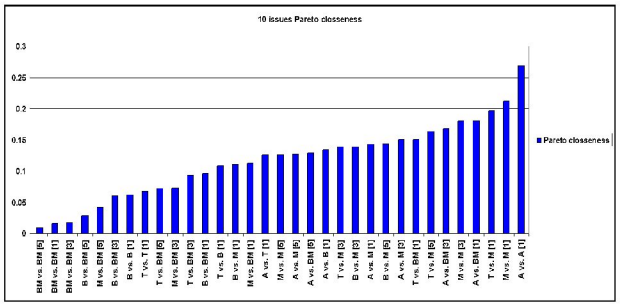


Figure 4:10-issue Pareto Frontier Closeness Outcomes

It is also noticed that when agent B follows the Bayesian–mining strategy, it generally achieves the shortest distance between the agreement and the Pareto Frontier when its opponent is more rational. However, when the agent uses the Mining strategy, there is no general rule to judge which opponent strategy would be better. When agent B follows the Trade-off strategy, it reaches better agreement when the opponent uses the Trade-off, then the Bayesian, and lastly ABMP. When agent B follows the Bayesian strategy (Figures 1, 2 and 3), it reaches better agreements when the opponent is the ABMP, then the Trade-off, and lastly the Bayesian itself. However, in 10-issue experiments (Figure 4), the order of its opponent strategies are the Bayesian, then the Trade-off and lastly ABMP. When agent B follows the ABMP, it reaches better agreements when the opponent uses the Trade-off, then the Bayesian and lastly ABMP.

7.2.2 Negotiation Cost and Utility

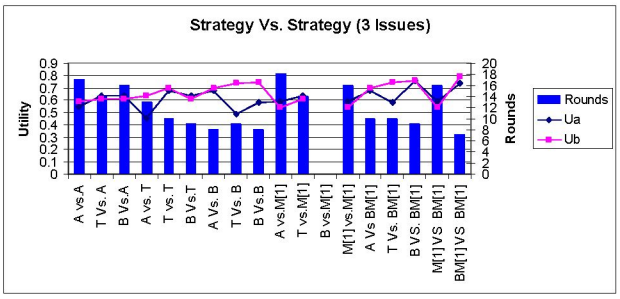


Figure 5:3-issue Single-Session experiments

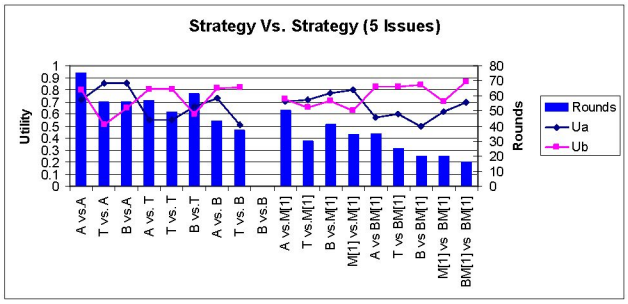


Figure 6. 5-issue Single-Session Experiments

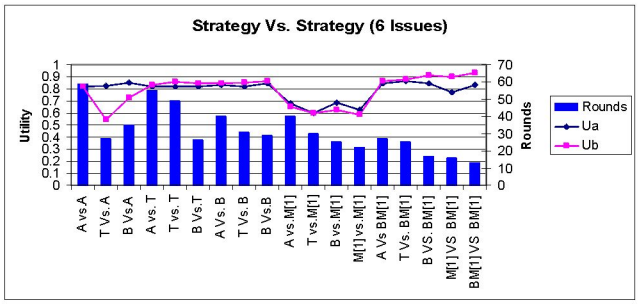


Figure 7: 6-issue Single-Session Experiments

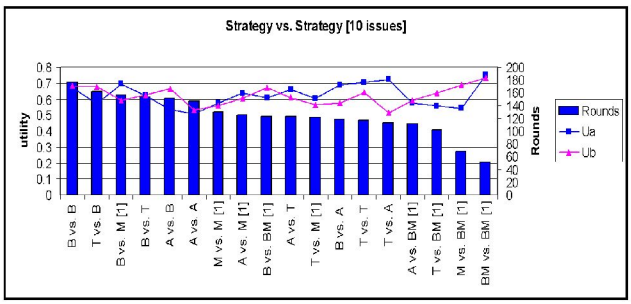
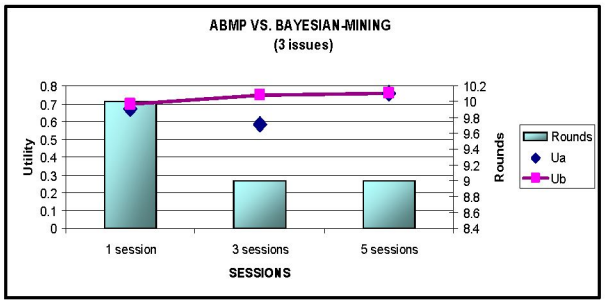
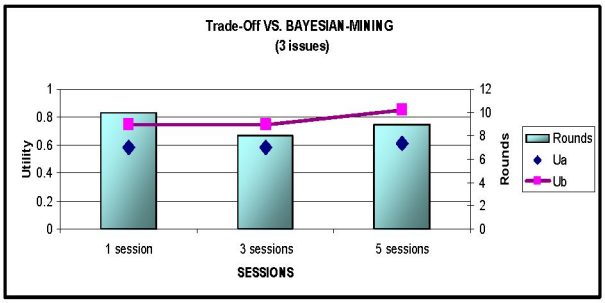
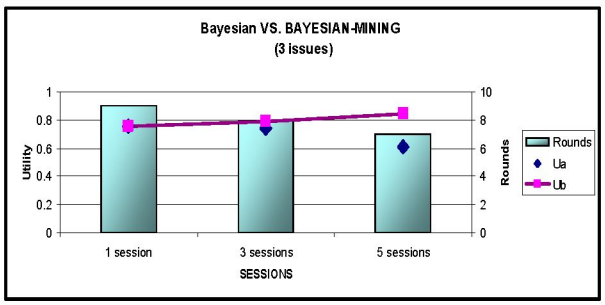
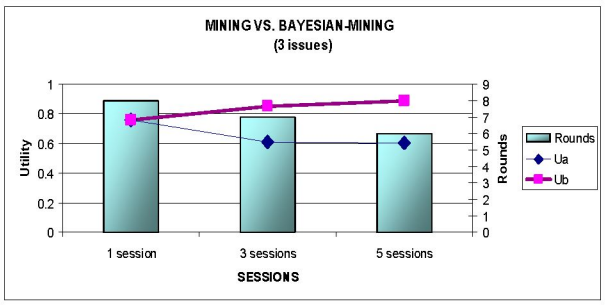


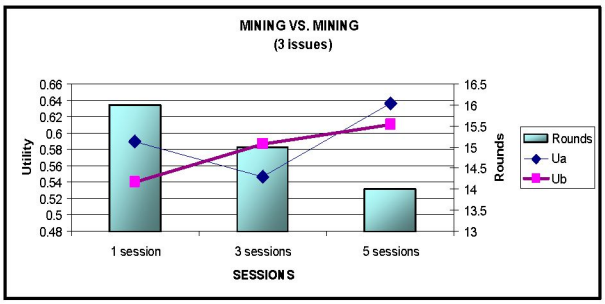
Figure 8: 10-issue Single-Session Experiments











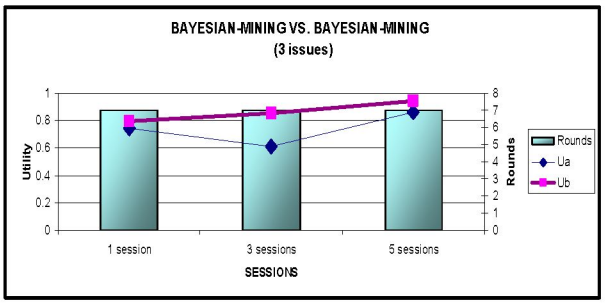
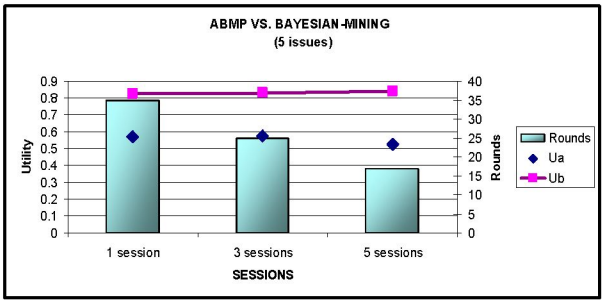
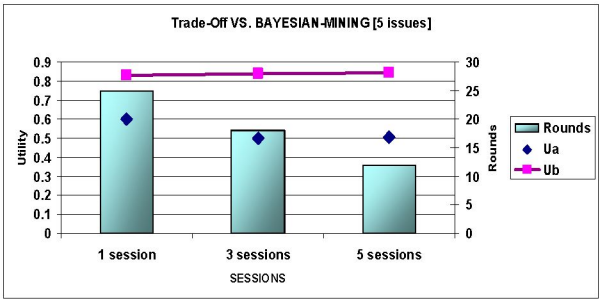
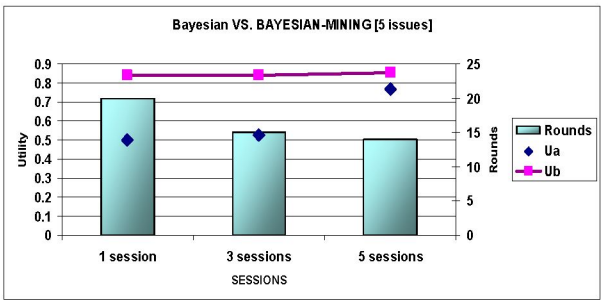
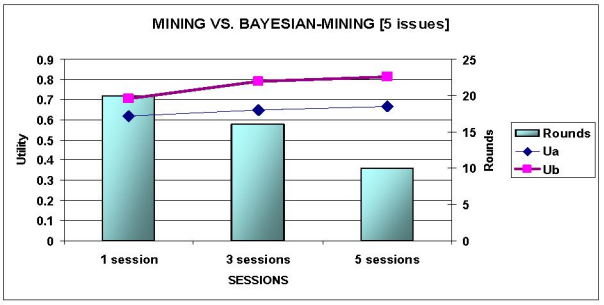


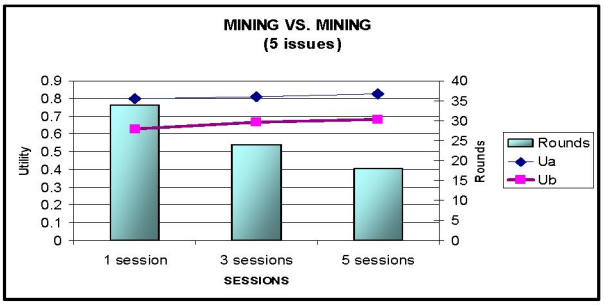
Figure 9: 3-issue Opponent vs. Bayesian-Mining











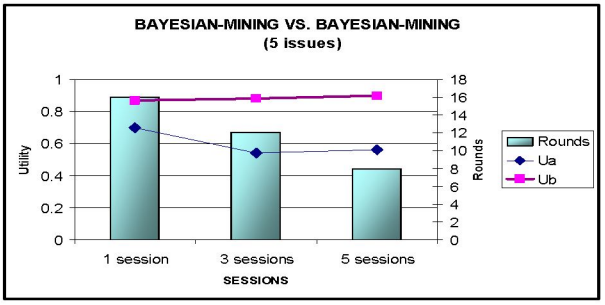
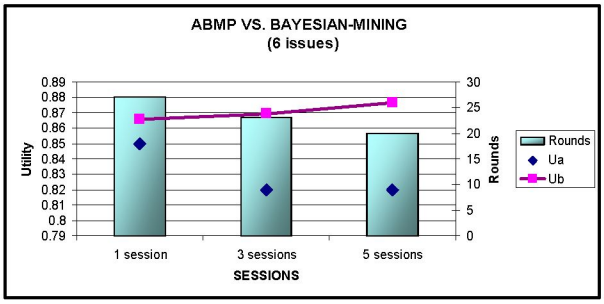
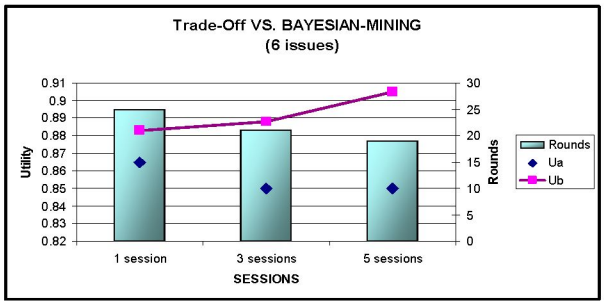
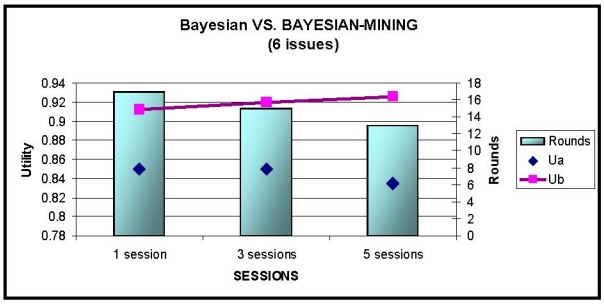
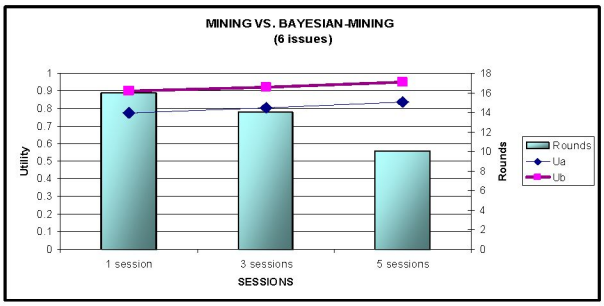


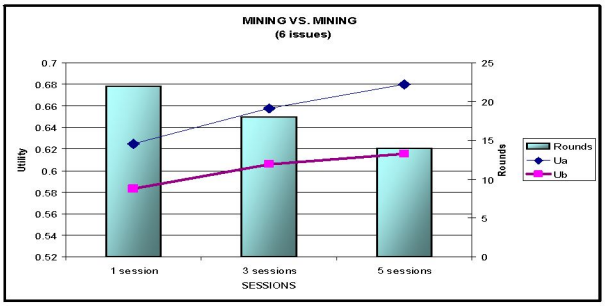
Figure 10: 5-issue Opponent vs. Bayesian-Mining











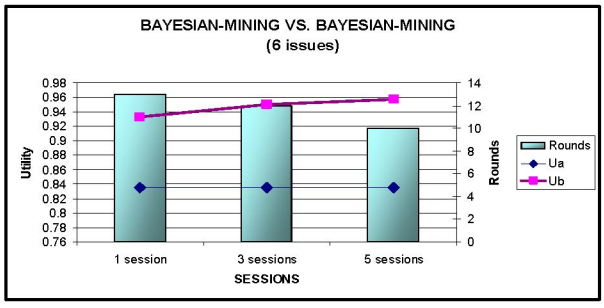
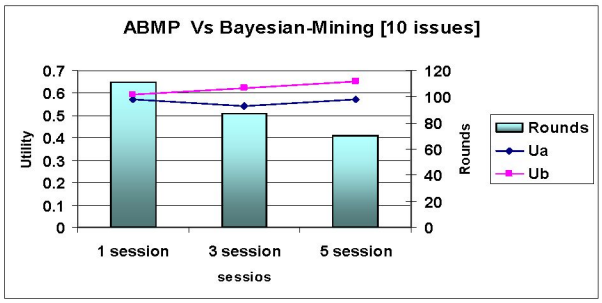
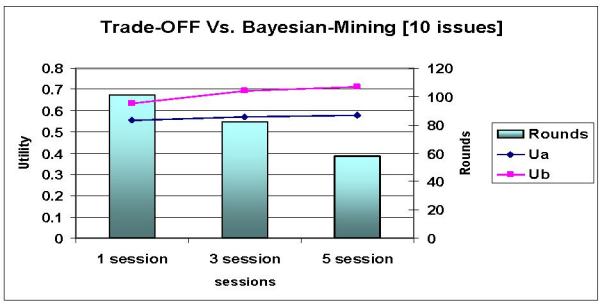
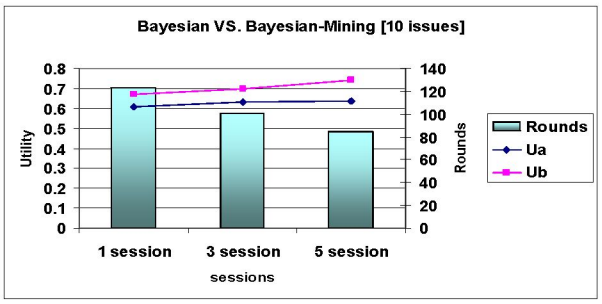
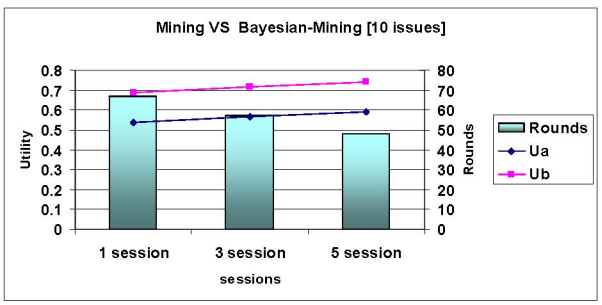


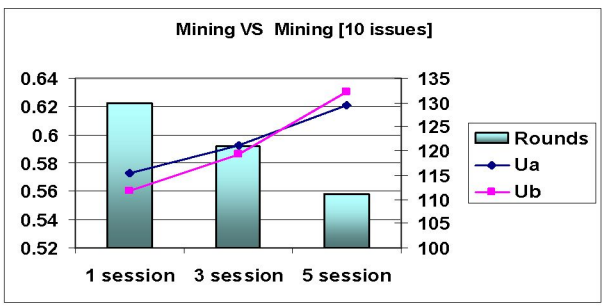
Figure 11: 6-issue Opponent vs. Bayesian-Mining











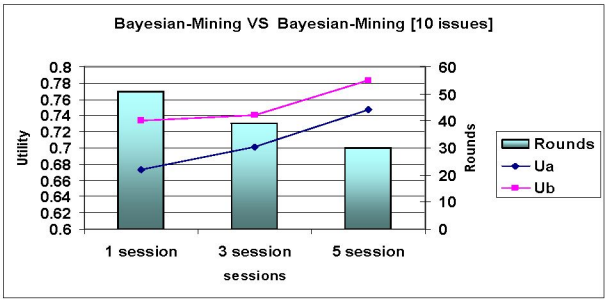


Figure 12: 10-issue Opponent vs. Bayesian-Mining

	3-issue	5-issue	6-issue	10-issue
ABMP	1.085	1.269	0.843	1.58
Trade-off	1.250	1.315	1.260	2.68
Bayesian	1.618	1.571	1.703	3.03
Mining	1.629	1.568	1.581	2.213
Bayesian-Mining	2.500	3.443	3.750	8.622

Table 7: Average Sensitivities for all strategies

The negotiation cost is presented by the number of rounds and increase in utility. It should be noticed that single session experiments are the baseline to compare all agent strategies (Figures 5, 6, 7and 8). Additionally, in all multi-session experiments (Figures 9, 10, 11 and 12), the curves of all existing agent strategies, except for our proposed strategy, are presented only by points as they never have previous session knowledge to shorten the negotiation rounds thus the related experiments are independent. To sum up the results, the following points may be stated.

- In all Bayesian–mining experiments, the agent wins its opponent from the first session. It gains more experience through negotiation steps and sessions such that gradually its utility is increased and the session rounds are decreased. For example, when Agent B follows Bayesian–mining and plays against an opponent Agent A which follows ABMP on 5-issue data set (Figure 10), Agent B utilities outcomes are (0.825, 0.83, 0.842) and the game session rounds are (35, 25, 17) in 1, 3 and 5 sessions respectively. Moreover, when the opponent agent A is Bayesian on 6-issue data set (Figure 11), Agent B utilities are (0.9125, 0.92, 0.92566) and the game session rounds are (17, 15, 13) in 1, 3 and 5 sessions respectively. One may notice that the first session between these parties has 17 rounds only, which are less than 29 rounds of Bayesian (Agent A) vs. Bayesian (Agent B) single session experiment (Figure 7).
- All other opponents (agent A) gain from playing with a Bayesian–mining agent (agent B). For example, agent A which follows ABMP (Figure 6) on 5-issue data set gets session rounds (75, 57, 43, 35) vs. Agent B which follows ABMP, Trade-Off, Bayesian, Bayesian–mining respectively. Also, the Trade-Off fastest single-session agreements (Figures 5, 6 and 7) occurred with the Bayesian–mining agent; 9 session rounds (3-issue data set) and 25 session rounds (5-issue and 6-issue data set).
- The Bayesian–mining agent outcomes (agent B) in all its experiments on 3-issue, 5-issue, 6-issue and 10-issue data sets effectively prove its principles.
- It is illustrated that the Bayesian–mining approach has less offers to reach rapidly the final agreement and the final utility. In most cases, especially in 10-issue experiments, it raises the opponent utilities. The Bayesian–mining approach works with larger number of sessions having several issues as the accumulated knowledge becomes valuable.
- It is noticed that in terms of the overall negotiation quality and number of proposals exchanged to reach an agreement, the Bayesian–mining approach outperformed the other strategies. This confirmed the intuition that building mining and learning capability into agents help the agents to work more accurate with its opponent with better performance and less expensive process.

7.2.3 Sensitivity analysis

The sensitivity analysis is interested only in the negotiation intra-transaction. Table 7 summarizes the average sensitivity for all negotiation strategies used in this research. Figures 13, 14, 15 and 16 illustrate the values of this study for all the negotiation experiments.

The average sensitivity for the Bayesian-mining strategy is greater than all other strategies, which is also influenced by the preferences' alternatives of each kind of issue (Table 7). In 3-issue experiments, the sensitivity increment ratio between Bayesian-mining (2.50) and the second highest sensitivity, i.e. Mining approach, (1.629) is 53%. In 5-issue experiments, the sensitivity increment ratio between Bayesian-mining (3.443) and second highest sensitivity, i.e. Bayesian approach, (1.571) is 119%. In 6-issue experiments, the sensitivity increment ratio between Bayesian-mining (3.750) and the second highest sensitivity, i.e. Bayesian approach, (1.703) is 120%. In all 10-issue experiments, the sensitivity increment ratio between Bayesian-mining (8.622) and the second highest sensitivity, i.e. Bayesian approach (3.03) is 184%. Figures 13, 14, 15 and 16 show that the sensitivity has increased after the first session due to the nature of the Bayesian-mining strategy which ranks the proposed offers during the session at the end of each session; the higher sensitivity value means that the agent who owns the related strategy has more information about the behaviour and the weights of preferences which the opponent gives to the issues.

In the sensitivity deep analysis, it can be found that for 3-issue, the Bayesian–mining approach sensitivity is between 2 and 4, but for other agent strategies, the sensitivity is between 0.667 and 2. For 5-issue, the Bayesian–mining approach is between 1.4 and 6, but for other agent strategies, the sensitivity is between 1.0 and 2.33. For 6-issue, the Bayesian–mining approach sensitivity is between 2.143 and 7, but for other agent strategies, the sensitivity is between 0.5 and 2.50. For 10-issue, the Bayesian–mining approach strategy is between 5.824 and 13, but for other agent strategies, the sensitivity is between 0.915 and 4.058.

To sum up the sensitivity results, the following points should be clarified:

- Increasing the issues number leads to increasing the sensitivity values of all strategies, because the

ability to select among great number of offers having many alternatives is increased.

- The sensitivity is proportional to the agent rationality. Table 7 shows evidence that the descending rationality order would be Bayesian-Mining, Bayesian, Mining, Trade-off, and ABMP.
- The sensitivity for the agent uses the Bayesian–mining strategy to the opponent in negotiation is more than any other strategy, which means that the agent outcomes are nearly closed to Pareto frontier.
- If the agent has several history proposals, it has enough knowledge to be more sensitive and more close to the Pareto Frontier. Consequently, the proposed agent records most sensitive steps to its opponent.
- The sensitivity feature is asymmetric; one agent may be sensitive to the other's preferences, but not vice-versa.

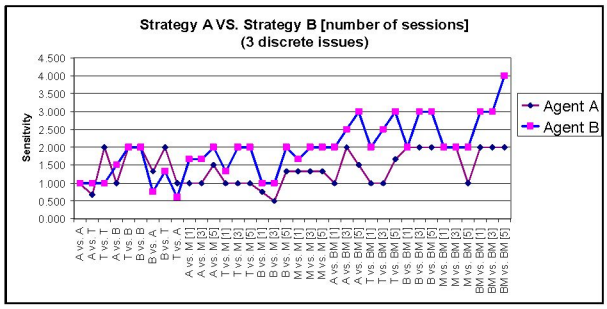


Figure 13: 3-issue Sensitivity Outcomes

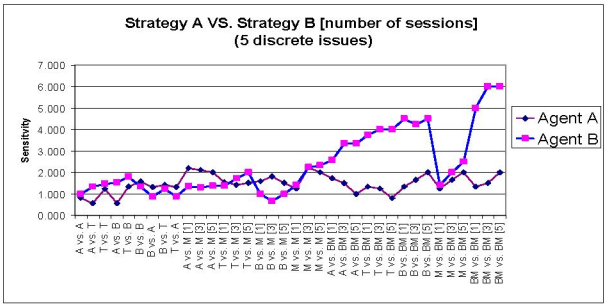


Figure 14: 5-issue Sensitivity Outcomes

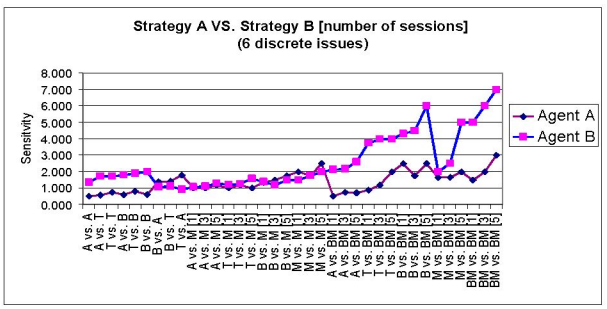


Figure 15: 6-issue Sensitivity Outcomes

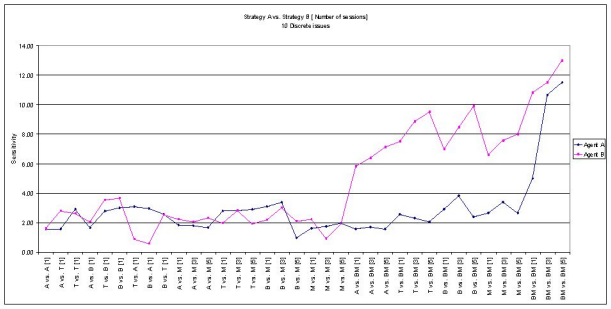


Figure 16: 10-issue Sensitivity Outcomes

7.2.4 Class studies

In the class studies, Bayesian–mining sensitivity calculations, i.e. Fortunate Step, Selfish Step, Concession Step, Unfortunate Step, Nice Step, and Silent Step, are dependent on all historical bids. Hence, the selection of suitable bids which satisfy the conditions of these steps becomes almost granted. This may lead to more accurate/rational calculations than the other models' criteria. In this section, the study focuses on the effect of the presence of the Bayesian mining strategy in any agent (A or B) in the negotiation process.

The average unfortunate step over a negotiation trace decreases by 0.217 to 0.453 when the Bayesian–mining approach is included (average of all cases of 3-issue, 5-issue, 6-issue and 10-issue experiments), However; the average unfortunate steps is between 0.431 and 0.531 when the Bayesian–mining approach is not included.

The average concession step is between 0.477 and 0.952 when the Bayesian–mining approach is included; however, the average concession step is between 0.475 and 0.78 when the Bayesian–mining approach is not included. Also the average fortunate step is between 0.056 and 0.424 when the Bayesian–mining approach is included while the average fortunate step is between 0.019 and 0.392 when the Bayesian–mining approach is not included.

In 5-issue experiments (Table 8), the average unfortunate steps over a negotiation trace decreases with using Bayesian-Mining. When Bayesian–mining approach is included, the average unfortunate step is 0.369, while the average unfortunate step is 0.444 when Bayesian–mining approach is not included. Also the average fortunate step is 0.262 when the Bayesian–mining approach is included while the average fortunate step is 0.182 when the Bayesian–mining approach is not included.

In 6-issue experiments (Table 8), the average unfortunate steps over a negotiation trace decreases with using Bayesian-Mining. When Bayesian–mining approach is included the average unfortunate steps is 0.365, while the average unfortunate steps is 0.528 when Bayesian–mining approach is not included. Also the average fortunate step is 0.224 when the Bayesian–mining approach is included while the average fortunate step is 0.106 when the Bayesian–mining approach is not included. Also the average concession step is 0.635

when the Bayesian–mining approach is included while the average concession step is 0.545 when the Bayesian–mining approach is not included.

	Bayesian-Mining					
	Not included			Included		
Issues	Fortunate	Concession	Unfortunate	Fortunate	Concession	Unfortunate
10	0.375	0.673	0.473	0.409	0.828	0.294
6	0.106	0.545	0.528	0.224	0.635	0.365
5	0.182	0.541	0.444	0.262	0.580	0.396
3	0.041	0.522	0.448	0.231	0.552	0.363

Table 8: Averages of Some Class Studies

For 10-issue as illustrated in Table 8, the average unfortunate steps over a negotiation trace decreases with using Bayesian-Mining. When Bayesian–mining approach is included the average unfortunate steps is 0.294, while the average unfortunate steps is 0.473 when Bayesian–mining approach is not included. Also the average fortunate step is 0.409 when the Bayesian–mining approach is included while the average fortunate step is 0.375 when the Bayesian–mining approach is not included.

From the previous analysis, it is illustrated that the size of the unfortunate steps for an agent that uses the Bayesian–mining approach is lower than other agent strategies. The frequency size of the concession and fortunate steps for the Bayesian–mining agent is higher than other types of agent. Consequently, the Bayesian–mining agent, in most cases, performs the steps that increase its opponent's utility being sensitive to its opponent needs.

8 Conclusion and future work

This paper proposes an agent approach for automated multi-session multi-issue win-win competitive bilateral negotiation. The proposed approach exploits the historical and accumulated knowledge while negotiation advances between two competitive agents having the same number of issues, to select the suitable bid for both negotiating sides. In this research, the order statistics Bayesian–mining agent is used to estimate the opponent thinking, and then the enhanced trade-off is used to propose the counter-offer, reducing the processing cost in terms of number of rounds and increasing the chances of reaching an agreement with higher utilities for both competitive agents.

Extensive experiments are carried out to prove that the assumptions, hypothesis, and opponent modelling are effective. Furthermore, many proposed agent imperative features are verified. The proposed Bayesian–mining agent is sensitive and rational. When the agent meets its opponent, both parties win since they reach the negotiation agreement fast. Moreover, the Bayesian–mining agent utility is increased while negotiation sessions advance. However, its opponent utility is dependent on the related opponent preferences. Further experiments are conducted on large-scale data sets having 10-issue data set. It is proved that the Bayesian–mining strategy is valid for scalable number of issues.

Handling continuous issues should be investigated in the future. Studying how to minimize the large number of offers before starting the negotiation process is another topic that needs further investigations. It is also needed to get benefit from the domain knowledge to spur the negotiation skilfully. Our approach is based on negotiation strategies among rational agents; it could be worth investigating how to handle cases when our rational agent negotiates with irrational or emotional agent.

Acknowledgement

The authors are deeply indebted to Prof. Atef AbdelMeneim who have thoroughly revised the proposed statistical approach; his valuable input has enriched our work.

References

[1] T. Bosse, C.M. Jonker (2005). Human vs. Computer Behaviour in Multi-Issue Negotiation, *Proceedings of the 1st International Workshop on Rational, Robust, and Secure Negotiations in Multi-Agent Systems,* IEEE Computer Society Press, pp. 11-24.

[2] T. Bosse, C.M. Jonker, L.V der Meij, V. Robu, J. Treur (2005). A System for Analysis of Multi-Issue Negotiation, *Software Agent-Based Applications, Platforms and Development* Kits, R. Unland, M. Klusch and M. Calisti, eds, Birkhaeuser Publishing Company. pp. 253-280.

[3] L. Breierova, M. Choudhari (2001), *An Introduction to Sensitivity Analysis*, Massachusetts Institute of Technology MIT.

[4] G. Casella, S. Fienberg, I. Olkin (1999). *Mathematical Statistics A Unified Introduction*, Springer-Verlag New York, Inc.

[5] S. Chakrabarti, E. Cox, E. Frank, R.H. Güting, J.H., X. Jiang, M. Kamber, S.S. Lightstone, T.P. Nadeau, R.E.N.D Pyle, M. Refaat, M. Schneider, T.J. Teorey, I.H. Witten, (2009). *Data mining know it all*, Elsevier Inc.

[6] Y.M. Chen, P. Huang (2009). "Agent-based bilateral multi-issue negotiation scheme for e-market transactions", Elsevier Science Publishers B. V. Amsterdam, , The Netherlands, Volume 9, pp. 1057-1067.

[7] R.M. Coehoorn, N.R. Jennings (2004). Learning an Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs, *Proc. of 6th International Conference on E-Commerce*, pp. 59-68.

[8] H. Faiffa (2006). "*THE ART & SCIENCE OF NEGOTIATION*", Belknap Press of Harvard University Press.

[9] P. Faratin, C. Sierra, N. Jennings (2003). Using Similarity Criteria to Make Negotiation Trade-Offs in automated Negotiations, *Journal of Artificial Intelligence,* 142 (2), pp. 205-237.

[10] P. Giudici (2005). *Applied Data mining*, John Willy & Sons, Ltd.

[11] K. Hindriks, D. Tykhonov (2008), Towards a Quality Assessment Method for Learning Preference Profiles in Negotiation, *The Tenth International Workshop on Agent Mediated Electronic Commerce* (AMEC 2008).

[12] K. Hindriks, D. Tykhonov (2008). Opponent Modeling in Automated Multi-Issue Negotiation Using Bayesian Learning, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems* (AAMAS08), pp. 331-338.

[13] K. Hindriks, C.M. Jonker, D. Tykhonov (2007). Negotiation Dynamics: Analysis, concession tactics, and outcomes, *Intelligent Agent Technology, IAT.'07*. IEEE/WIC/ACM International Conference, 2-5 Nov. pp. 427-433.

[14] R.V. Hogg, A. Craig, J.W. McKean (2005). *Introduction to Mathematical Statistics*, Prentice Hall.

[15] C.M. Jonker, J. Treur (2001). An Agent Architecture for Multi-Attribute Negotiation, *Proc. of the 17th Int*. Joint Conference on AI (IJCAI'01), pp. 1195-1201.

[16] C.M. Jonker, V. Robu, J. Treur (2007). An Agent Architecture for Multi-Attribute Negotiation Using Incomplete Preference Information, *Autonomous Agents and Multi-Agent Systems Journa*l, Springer Netherlands Publisher, Volume 15, pp. 221-252.

[17] S. Kraus (2001). *Automated Negotiation and Decision Making in Multiagent Environments*, Springer-Verlag Berlin Heidelberg.

[18] R.Y.K. Lau, O. Wong (2007). Mining Negotiation Knowledge for Adaptive Negotiation Agents in e-Marketplaces, *Proceedings of the 40th Hawaii* International Conference on System Sciences.

[19] R.J. Lewicki, A. Hiam (2006). *Mastering business negotiation: a working guide to making deals and resolving conflict*, Published by Jossey-Bass A Wiley Imprint.

[20] R. Lin, S. Kraus, J. Wilkenfeld, J. Barry (2006). An Automated Agent for Bilateral Negotiation with Bounded Rational Agents with Incomplete Information. ECAI 2006, pp. 270-274.

[21] V. Narayanan, N.R. Jennings (2006). Learning to Negotiate Optimally in Non-stationary Environments, *CIA 2006*, LNAI 4149, pp. 288-300.

[22] J. Oesch, A. Galinsky (2003). First Offers in Negotiations: Determinants and Effects. *16th Annual IACM* Conference Melbourne, Australia.

[23] J.R. Oliver (1997). "A Machine-Learning Approach to Automated Negotiation and Prospects for Electronic Commerce", *Journal of Management Information Systems*. Vol. 13 No. 3, Winter pp. 83 – 112

[24] S. Parsons, C. Sierra, N.R. Jennings (1998). Agents that reason and negotiate by arguing, *Journal of Logic and computation*, 8(3), pp. 261-292.

[25] S.T. Rachev, J.S.J. Hsu, B.S. Bagasheva, F.J. Fabozzi, (2008). *Bayesian Methods in Finance*, John Wiley & Sons, Inc. Hoboken, New Jersey.

[26] P. Tan, M. Steinbach, V. kumar (2006). *Introduction to Data mining*, Pearson Education, Inc.

[27] D.A. Van Veldhuizen, G.B. Lamont (1998). Evolutionary Computation and Convergence to a Pareto Front, *Morgan Kaufmann*, pp. 221—228.

[28] T. Velden, K.W. Carsten (2007). *Person Perception in Negotiation: When Perceiving is (Not) for Doing* (February 2, 2007). IACM 2007 Meetings Paper, Working Paper Series.

[29] N. Vulkan N.R. Jennings (2000). Efficient Mechanisms for the Supply of Services in Multi-Agent Environments, *Int. Journal of Decision Support Systems*, 28 (1-2). pp. 5-19.

[30] L. Wang, X. Fu (2005). *Data Mining with Computational Intelligence,* Springer-Verlag Berlin Heidelberg.

[31] M. Wooldridge (2005). *An introduction to Multiagent Systems,* John Wiley & Sons, ltd.

[32] I. Zelic, I. Kononenko, N. Lavrac, V. Vuga (1997). Induction of decision trees and Bayesian classification, *Journal of Medical Systems, Volume 21, Number 6*, Springer Netherlands, pp. 429-444(16).

[33] D. Zeng, K. Sycara, (1997). *Benefits of Learning in Negotiation,* in Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), Providence, RI, pp. 36-41.

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 at: 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 seventeen 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:

Ajith Abraham, Siby Abraham, Renato Accornero, Raheel Ahmad, Cutting Alfredo, Hameed Al-Qaheri, Gonzalo Alvarez, Wolfram Amme, Nicolas Anciaux, Rajan Arora, Costin Badica, Zoltán Balogh, Andrea Baruzzo, Borut Batagelj, Norman Beaulieu, Paolo Bellavista, Steven Bishop, Marko Bohanec, Zbigniew Bonikowski, Borko Bosković, Marco Botta, Pavel Brazdil, Johan Brichau, Andrej Brodnik, Ivan Bruha, Maurice Bruynooghe, Wray Buntine, Dumitru Dan Burdescu, Yunlong Cai, Juan Carlos Cano, Tianyu Cao, Norman Carver, Marc Cavazza, Jianwen Chen, L.M. Cheng, Chou Cheng-Fu, Girija Chetty, G. Chiola, Yu-Chiun Chiou, Ivan Chorbev, Shauvik Roy Choudhary, Sherman S.M. Chow, Lawrence Chung, Mojca Ciglarič, Jean-Noël Colin, Vittorio Cortellessa, Jinsong Cui, Alfredo Cuzzocrea, Darko Čerepnalkoski, Gunetti Daniele, Grégoire Danoy, Manoranjan Dash, Paul Debevec, Fathi Debili, Carl James Debono, Joze Dedic, Abdelkader Dekdouk, Bart Demoen, Sareewan Dendamrongvit, Tingquan Deng, Anna Derezinska, Gaël Dias, Ivica Dimitrovski, Jana Dittmann, Simon Dobrišek, Quansheng Dou, Jeroen Doumen, Erik Dovgan, Branko Dragovich, Dejan Drajic, Jozo Dujmovic, Umut Riza ErtÃijrk, CHEN Fei, Ling Feng, YiXiong Feng, Bogdan Filipič, Iztok Fister, Andres Flores, Vladimir Fomichov, Stefano Forli, Massimo Franceschet, Alberto Freitas, Jessica Fridrich, Scott Friedman, Chong Fu, Gabriel Fung, David Galindo, Andrea Gambarara, Matjaž Gams, Maria Ganzha, Juan Garbajosa, Rosella Gennari, David S. Goodsell, Jaydeep Gore, Miha Grčar, Daniel Grosse, Zhi-Hong Guan, Donatella Gubiani, Bidyut Gupta, Marjan Gusev, Zhu Haiping, Kathryn Hempstalk, Gareth Howells, Juha Hyvärinen, Dino Ienco, Natarajan Jaisankar, Domagoj Jakobovic, Imad Jawhar, Yue Jia, Ivan Jureta, Dani Juričić, Zdravko Kačič, Slobodan Kalajdziski, Yannis Kalantidis, Boštjan Kaluža, Dimitris Kanellopoulos, Rishi Kapoor, Andreas Kassler, Daniel S. Katz, Samee U. Khan, Mustafa Khattak, Elham Sahebkar Khorasani, Ivan Kitanovski, Tomaž Klobučar, Ján Kollár, Peter Korošec, Valery Korzhik, Agnes Koschmider, Jure Kovač, Andrej Krajnc, Miroslav Kubat, Matjaz Kukar, Anthony Kulis, Chi-Sung Laih, Niels Landwehr, Andreas Lang, Mohamed Layouni, Gregor Leban, Alex Lee, Yung-Chuan Lee, John Leggett, Aleš Leonardis, Guohui Li, Guo-Zheng Li, Jen Li, Xiang Li, Xue Li, Yinsheng Li, Yuanping Li, Shiguo Lian, Lejian Liao, Ja-Chen Lin, Huan Liu, Jun Liu, Xin Liu, Suzana Loskovska, Zhiguo Lu, Hongen Lu, Mitja Luštrek, Inga V. Lyustig, Luiza de Macedo, Matt Mahoney, Domen Marinčič, Dirk Marwede, Maja Matijasevic, Andrew C. McPherson, Andrew McPherson, Zuqiang Meng, France Mihelič, Nasro Min-Allah, Vojislav Misic, Vojislav Mišić, Mihai L. Mocanu, Angelo Montanari, Jesper Mosegaard, Martin Možina, Marta Mrak, Yi Mu, Josef Mula, Phivos Mylonas, Marco Di Natale, Pavol Navrat, Nadia Nedjah, R. Nejabati, Wilfred Ng, Zhicheng Ni, Fred Niederman, Omar Nouali, Franc Novak, Petteri Nurmi, Denis Obrul, Barbara Oliboni, Matjaž Pančur, Wei Pang, Gregor Papa, Marcin Paprzycki, Marek Paralič, Byung-Kwon Park, Torben Bach Pedersen, Gert Schmeltz Pedersen, Zhiyong Peng, Ruggero G. Pensa, Dana Petcu, Marko Petkovšek, Rok Piltaver, Vid Podpečan, Macario Polo, Victor Pomponiu, Elvira Popescu, Božidar Potočnik, S. R. M. Prasanna, Kresimir Pripuzic, Gabriele Puppis, HaiFeng Qian, Lin Qiao, Jean-Jacques Quisquater, Vladislav Rajkovič, Dejan Rakovic, Jean Ramaekers, Jan Ramon, Robert Ravnik, Wilfried Reimche, Blagoj Ristevski, Juan Antonio Rodriguez-Aguilar, Pankaj Rohatgi, Wilhelm Rossak, Eng. Sattar Sadkhan, Sattar B. Sadkhan, Khalid Saeed, Motoshi Saeki, Evangelos Sakkopoulos, M. H. Samadzadeh, MariaLuisa Sapino, Piervito Scaglioso, Walter Schempp, Barabara Koroušić Seljak, Mehrdad Senobari, Subramaniam Shamala, Zhongzhi Shi, LIAN Shiguo, Heung-Yeung Shum, Tian Song, Andrea Soppera, Alessandro Sorniotti, Liana Stanescu, Martin Steinebach, Damjan Strnad, Xinghua Sun, Marko Robnik Šikonja, Jurij Šilc, Igor Škrjanc, Hotaka Takizawa, Carolyn Talcott, Camillo J. Taylor, Drago Torkar, Christos Tranoris, Denis Trček, Katarina Trojacanec, Mike Tschierschke, Filip De Turck, Aleš Ude, Wim Vanhoof, Alessia Visconti, Vuk Vojisavljevic, Petar Vračar, Valentino Vranić, Chih-Hung Wang, Huaqing Wang, Hao Wang, Hui Wang, YunHong Wang, Anita Wasilewska, Sigrid Wenzel, Woldemar Wolynski, Jennifer Wong, Allan Wong, Stefan Wrobel, Konrad Wrona, Bin Wu, Xindong Wu, Li Xiang, Yan Xiang, Di Xiao, Fei Xie, Yuandong Yang, Chen Yong-Sheng, Jane Jia You, Ge Yu, Borut Zalik, Aleš Zamuda, Mansour Zand, Zheng Zhao, Dong Zheng, Jinhua Zheng, Albrecht Zimmermann, Blaž Zupan, Meng Zuqiang

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, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2011 (Volume 35) 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: Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

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 (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

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

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 35 Number 1 March 2011 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Informatica **35** (2011) Number 1, pp. 1–139