Elektrotehniški vestnik 74(5): 241-247, 2007
Electrotechnical Review: Ljubljana, Slovenija

Using stakeholder-driven process performance measurement for monitoring the performance of a Scrum-based software development process

Viljan Mahnič[1], **Ivan Vrana**[2]

[1] *University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, 1000 Ljubljana*
[2] *Czech University of Life Sciences in Prague, Department of Information Engineering, Kamýcká 129, 165 21 Prague 6 - Suchdol*
E-mail: viljan.mahnic@fri.uni-lj.si

Abstract. We describe a metrics plan for monitoring and improving the performance of the software development process based on the Scrum agile method. After a short introduction to Scrum concepts a detailed description of the proposed metrics is provided. The metrics are defined using the principles of stakeholder-driven process performance measurement that requires a balanced approach considering viewpoints of different stakeholders. The goals of each stakeholder are defined first followed by the choice of appropriate performance indicators. The evaluation of each indicator is based on metric values collected during process execution. The metrics plan enables a stepwise introduction of metrics which can be incorporated into the Scrum method seamlessly without affecting the agility of the development process.

Keywords: Scrum, software process improvement, software metrics

Merjenje učinkovitosti razvoja programske opreme po metodi Scrum upoštevajoč cilje različnih interesnih skupin

Povzetek. V članku je predstavljen načrt meritev za spremljanje in izboljšanje učinkovitosti procesa razvoja programske opreme po metodi Scrum. Kratki uvodni predstavitvi metode Scrum sledi podroben opis predlaganih metrik. Metrike so definirane tako, da zagotavljajo uravnotežen pristop z upoštevanjem pogledov različnih interesnih skupin, ki sodelujejo v razvojnem procesu. Za vsako interesno skupino so najprej določeni njeni cilji, nato pa izbrani ustrezni kazalniki, s katerimi prikazujemo doseganje posameznih ciljev. Izračun vrednosti vsakega kazalnika poteka s pomočjo metrik, katerih vrednosti zbiramo med izvajanjem procesa. Načrt meritev omogoča postopno uvajanje posameznih metrik, ki jih lahko na preprost način vključimo v metodo Scrum, ne da bi s tem okrnili agilnost razvojnega procesa.

Ključne besede: Scrum, izboljšanje procesa za razvoj programske opreme, metrike v programski opremi

Received 28 May 2007
Accepted 13 November 2007

1 Introduction

In spite of all endeavours to improve the software process by introducing rigor and discipline as advocated by software quality models (e.g., CMMI [1]) we are still faced with a lot of failed projects. Surveys of more than 8,000 projects show that most project failures involve stakeholder problems causing that projects fail because of people and project management issues rather than technical issues [2]. For this reason, numerous agile methods have appeared in the last decade [3] that – in contrast to disciplined approach advocated by the quality models – value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [4]. Following the principle of "maximizing the amount of work not needed to be done" these methods to a great extent abandon many practices prescribed by software quality models including the need for comprehensive metrics plans.

Scrum [5, 6] is one of the most widely used agile methods that concentrates mainly on managing software projects. In the last few years several successful implementations of Scrum have been reported in the literature [7, 8, 9, 10]. Experience has shown that

adopting agile methods improves management of the development process and customer relationships [2], and decreases the amount of overtime and increases customer satisfaction [9].

Within Scrum only one software development metric is used: the estimate of the amount of work remaining that needs to be done in order to complete a Product Backlog item or a task in a Sprint Backlog. Using this metric, burndown charts can be developed showing work remaining over time. The Scrum books define a Sprint Burndown chart as a place to see daily progress, and a Product Burndown chart as where to show monthly (per Sprint) progress.

However, in the last few years researchers and practitioners have recognized that Scrum needs more elaborate metrics that would provide better insight into the software development process. Schatz and Abdel-shafi [7] cite the stakeholders' concern with the lack of metrics regarding the project's projected completion date. Yap [11] stresses the need for agile methods to provide a better way to measure the total value delivered in relation to cost. Hartmann and Dymond [12] discuss the criteria for defining appropriate agile metrics pointing out that improper metrics simply adopted from plan-driven approach not only waste resources but also skew team behaviour in counter-productive ways and undermine culture change inherent in Agile work. Sulaiman et al. [13] describe an adaptation of the Earned Value Management method [14] for Scrum projects.

The aim of this paper is to contribute to the aforementioned efforts by specifying a metrics plan that makes it possible to monitor and improve the software development process considering the views of different stakeholders. In the next section we briefly introduce Scrum in order to acquaint the reader with the basic Scrum concepts and terminology. Section 3 describes the metrics we propose to be introduced without harming the agility of the Scrum method. In Section 4 the points on the process timescale are described where the proposed metric values are collected. Section 5 outlines the most important conclusions and gives some directions for further work.

2 Overview of Scrum

Scrum starts with the premise that software development is too complex and unpredictable to be planned exactly in advance. Instead, empirical process control must be applied to ensure visibility, inspection, and adaptation. This is achieved through an iterative and incremental development process shown in Fig. 1.

2.1 Scrum roles

Scrum implements this process through three roles: the Product Owner, the Team, and the ScrumMaster.

The Product Owner is responsible for representing the interests of everyone with a stake in the project and its resulting system. He maintains the Product Backlog, a prioritized list of project requirements with estimated times to turn them into completed product functionality.

The Team is responsible for developing functionality. Teams are self-managing, self-organizing, and cross-functional, and they are responsible for figuring out how to turn Product Backlog into an increment of functionality within an iteration and managing their own work to do so. Team members are collectively responsible for the success of each iteration and of the project as a whole.

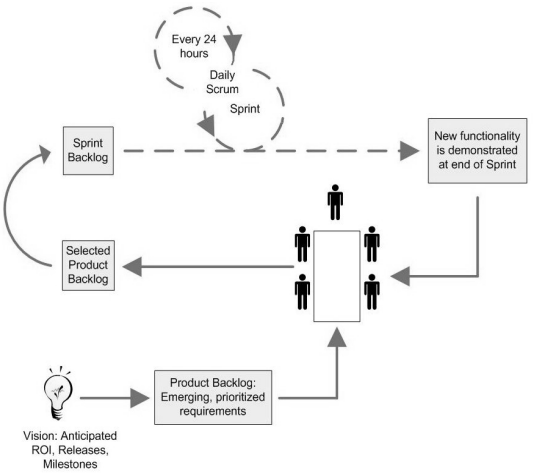

Figure 1: Detailed Scrum flow.

The ScrumMaster fills the position normally occupied by the project manager, but his/her role is slightly different. He/She is responsible for managing the Scrum process so that it fits within an organization's culture and still delivers the expected benefits, and for ensuring that everyone follows Scrum rules and practices.

2.2 Process description

As shown in Fig. 1, a Scrum project starts with a vision of the system to be developed. Then a Product Backlog list is created containing all the requirements that are currently known. The Product Backlog is prioritized and divided into proposed releases.

All the work is done in Sprints. Each Sprint is an iteration of 30 consecutive calendar days. It is initiated with a Sprint planning meeting, where the Product Owner and Team get together to agree upon Product Backlog items to be implemented over the next Sprint.

After deciding what has to be done in the next Sprint, the Team develops the Sprint Backlog, i.e., a list of tasks that must be performed to deliver a completed increment of potentially shippable product functionality

by the end of the Sprint. The tasks in the list emerge as the Sprint evolves and should be divided so that each takes roughly 4 to 16 hours to finish.

Every day the Team gets together for a 15-minute meeting called a Daily Scrum. At the Daily Scrum, each Team member answers three questions: What have you done on this project since the last Daily Scrum meeting? What will you do before the next meeting? Do you have any obstacles? The purpose of the meeting is to synchronize the work of all Team members and to schedule any meetings that the Team needs to forward its progress.

At the end of the Sprint, a Sprint review meeting is held at which the Team presents what was developed during the Sprint to the Product Owner and any other stakeholders who want to attend. After the Sprint review and prior to the next Sprint planning meeting, the ScrumMaster also holds a Sprint retrospective meeting in order to encourage the Team to revise, within the Scrum process framework, its development process to make it more effective and enjoyable for the next Sprint.

3 Definition of metrics

In addition to the estimated work remaining which is calculated daily and graphed resulting in a Sprint Burndown chart we propose a set of useful metrics that provide a comprehensive insight in project performance.

The definition of metrics is based on the concept of a process performance measurement system [15] which advocates a balanced approach considering views of different stakeholders that take part in the process. The metrics plan we propose considers the views of three stakeholders: IT management, Team members, and customers. Metrics are defined using a top-down approach similar to the Goal/Question/Metric Method [16]. The goals of each stakeholder are defined first followed by the choice of appropriate performance indicators. Finally, the metrics that enable evaluation of each indicator are defined. Considering the recommendations from [15], the proposed indicators describe the process quantitatively and qualitatively, thus providing a holistic view of the process performance. Goals of different stakeholders are defined on the basis of authors' experience in using agile methods in development of university information systems [10, 17]; however we believe that the proposed goals, indicators and metrics are general enough to be used in any software development organization using the Scrum method.

3.1 Stakeholder 1: IT management

IT management is mainly concerned with traditional aspects of software development performance considering time, cost, and quality pursuing the following goals:

- Goal 1: Timely information on project performance with emphasis on projects that tend to be late or over budget.
- Goal 2: Quality improvement.

3.1.1 Goal 1

Table 1 shows indicators and metrics for measuring the achievement of the first goal as well as notation of each metric used in the following formulae.

The ratio between the work spent and the decrement of work remaining for the period from day *d1* till day *d2* of the Sprint is computed using formula (1) where *n* is the number of tasks in the Sprint Backlog:

$$\frac{\sum_{i=d1}^{d2-1}\sum_{j=1}^{n} WS_{i,j}}{\sum_{j=1}^{n} WR_{d1,j} - \sum_{j=1}^{n} WR_{d2,j}} \tag{1}$$

The target value of this indicator is 1 or less which means that the amount of work remaining diminishes proportionally to the amount of work spent.

The Schedule Performance Index (SPI) is the ratio between the earned value (i.e., the value of all tasks completed) and the planned value (i.e., the initial estimate of effort for all tasks to be completed till a certain point within the project). Since Scrum does not prescribe the project schedule model, we assume that the amount of tasks that must be accomplished at a certain point in the Sprint is proportional to the time elapsed from the beginning of the Sprint. The work remaining and work spent metrics allow a precise definition of the earning rule (ER) for each task *j* in the Sprint Backlog on the day *d* of a Sprint. It can be expressed as a ratio between the amount of work already spent and all the work required (spent and remaining) to accomplish the task:

$$ER_{d,j} = \frac{\sum_{i=1}^{d-1} WS_{i,j}}{\sum_{i=1}^{d-1} WS_{i,j} + WR_{d,j}} \tag{2}$$

Using the earning rule from formula (2), the SPI on day *d* is computed as

$$SPI_d = \frac{\sum_{j=1}^{n} ER_{d,j} WR_{init,j}}{\sum_{j=1}^{n} WR_{init,j}} \cdot \frac{SL}{DE} \tag{3}$$

where $WR_{init,j}$ denotes the initial estimate of the work remaining for task *j*. SPI greater than 1 means that the project is ahead of schedule and vice versa. Therefore, the target value for SPI is 1 or more.

The Cost Performance Index (CPI) is the ratio between the earned value and actual costs as shown in formula (4). While the computation of SPI allows the earned value to be measured in any of the units (we use the initial estimates of hours of the work remaining for each task in the Sprint Backlog) the computation of CPI requires the earned value and actual costs to be

expressed in units of currency. Using the work spent metrics, we can compute the actual labor costs exactly by multiplying hours spent and the cost of an engineering hour CEH_j for all tasks in the Sprint Backlog. Similarly, the earned value is computed by multiplying the earned hours and CEH_j.

$$CPI_d = \frac{\sum_{j=1}^{n} ER_{d,j} \cdot WR_{init,j} \cdot CEH_j}{\sum_{i=1}^{DE} \sum_{j=1}^{n} WS_{i,j} \cdot CEH_j} \quad (4)$$

The target value for CPI is 1 or more, indicating that the cost of completing the work is right on plan or less than planned.

3.1.2 Goal 2

Indicators and metrics for Goal 2 of IT management are shown in Table 2.

Error density is a standard indicator used in the software industry. It requires measuring the size of code and the number of errors. Since Scrum advocates self-organization and self-management of Teams, we do not interfere in the process of testing and error discovery that takes place within the Sprint, but only measure the number of errors reported at the end of the Sprint (at the Sprint review meeting) and in a fixed period after release. The size of code and the number of errors are measured for each Product Backlog item (PBI) separately, thus giving a detailed figure of the quality of each PBI. The error density of a Sprint or release is derived by computing sums over all PBIs in the Sprint or release, respectively.

Measuring the costs of rework requires the tasks in the Sprint Backlog to be classified according to the type of work performed, e.g., development, testing, rework due to the change in requirements, rework due to error reported by the customer, etc. This is achieved by simply adding the corresponding attribute to each task in the Sprint Backlog. The amount of rework can be measured in hours spent or in currency units by rolling up either the values of work spent $WS_{i,j}$ or products $WS_{i,j} \cdot CEH_j$ for all tasks that refer to rework. In the same way the amount and costs of work spent can be obtained for all other types of work (e.g., development, testing, etc.).

Table 1. Indicators and metrics for Goal 1 of IT management

Indicator	Direct Metrics
Ratio between the work spent and the decrement of work remaining	Work spent on day *i* for each task *j* in the Sprint Backlog - WS_{ij} Work remaining on day *i* for each task *j* in the Sprint Backlog - WR_{ij}
Schedule Performance Index	Work remaining on day *i* for each task *j* in the Sprint Backlog - WR_{ij} Work spent on day *i* for each task *j* in the Sprint Backlog - WS_{ij} The length of the Sprint (number of working days in the Sprint) - *SL* The number of days elapsed from the beginning of the Sprint - *DE*
Cost Performance Index of labor costs	Work remaining on day *i* for each task *j* in the Sprint Backlog - WR_{ij} Work spent on day *i* for each task *j* in the Sprint Backlog - WS_{ij} Cost of Team member's engineering hour (for each task *j* in the Sprint Backlog) - CEH_j The length of the Sprint (number of working days in the Sprint) - *SL* The number of days elapsed from the beginning of the Sprint – *DE*

Table 2. Indicators and metrics for Goal 2 of IT management

Indicator	Direct Metrics
Error Density (number of errors per KLOC)	The number of errors found during the Sprint review meeting (for each PBI separately) The number of errors reported by the user in a fixed period after release (for each PBI separately) The size of the code (for each PBI separately)
Costs of rework	Work spent on day *i* for each task *j* in the Sprint Backlog referring to rework (classification of tasks in the Sprint Backlog is required) - WS_{ij} Cost of Team member's engineering hour (for each task *j* in the Sprint Backlog) - CEH_j
Fulfillment of Scope (have all PBIs or Sprint Backlog tasks been implemented)	Total number of PBIs in the release/Sprint The number of PBIs completed in the release/Sprint Total number of tasks in the Sprint The number of tasks completed during the Sprint Work remaining on day *i* for each task *j* in the Sprint Backlog - WR_{ij}

Table 3. Indicators and metrics for Goal "Job satisfaction"

Indicator	Direct Metrics
The average amount of overtime at Sprint/release/project level	Work spent on day *i* for each task *j* in the Sprint Backlog - WS_{ij} The length of the Sprint (number of working days in the Sprint) - *SL* Percentage of Team member's engagement in the project - $PTME_m$ The number of administrative days - *AD* The number of Team members (the size of Team *t*) - NTM_t
The average number of projects the employees work in parallel	The number of Team members (the size of Team *t*) - NTM_t The total number of developers – *ND*
Qualitative evaluation of working conditions like communication and teamwork, physical discomfort, psychological well-being, workload, supervision, opportunities for growth, etc.	Results of the survey conducted at the Sprint retrospective meeting. Each question is marked between 1 and 5, where 1 is the worst and 5 is the best mark.

Table 4. Indicators and metrics for Goal "Satisfied customers"

Indicator	Direct Metrics
Qualitative evaluation of customer satisfaction using criteria like the quality of product, price adequacy, reliability in terms of time and costs, completeness of product delivered at the end of each Sprint or release, flexible handling of changes in requirements, good collaboration with the development team, adequate training and documentation, etc.	Results of the survey conducted at the end of each Sprint/release. Each question is marked between 1 and 5, where 1 is the worst and 5 is the best mark.

Fulfillment of Scope is a simple indicator that shows how the project team fulfills the commitments agreed at the beginning of each Sprint. It can be computed in several ways offering different levels of detail: as the ratio between the number of PBIs actually implemented and the number of PBIs committed; as the ratio between the number of tasks completed and the number of all tasks in the Sprint Backlog; or as the ratio between the initial estimates of work remaining for all completed tasks and the initial estimates of work remaining for all tasks in the Sprint Backlog. The first way can also be used to compute the fulfillment of scope of each release. The target value of this indicator is 1, meaning that all commitments agreed at the beginning of each Sprint/release were fulfilled.

3.2 Stakeholder 2: Team members

The main goal of Team members is "Job satisfaction". Team members are most productive if they have good working conditions enabling a sustainable pace of progress without excessive workload and working overtime. In order to measure the achievement of this goal, we propose a combination of quantitative and qualitative indicators as shown in Table 3.

The average amount of overtime is measured quantitatively as the ratio between Actual Hours and Expected Hours. The Actual Hours are computed by rolling up the values of work spent $WS_{i,j}$ on all tasks of the Sprint, while the computation of the Expected Hours must take into account that some Team members are engaged on the project only part-time and that Team members may not be at work all the days of the Sprint due to administrative days (e.g., sickdays, vacation, coursedays, compassionate leave). Assuming that the working day has 7.5 hours, the Expected Hours of each Team Member EH_m are calculated as follows:

$$EH_m = 7.5 \cdot (SL - AD) \cdot PTME_m / 100 \quad (5)$$

The amount of overtime OT_t for Team *t* is then computed considering the Expected Hours (5) of all Team members as shown in (6):

$$OT_t = \frac{\sum_{i=1}^{SL}\sum_{j=1}^{n} WS_{i,j}}{\sum_{m=1}^{NTM_t} EH_m} \quad (6)$$

The average number of projects (ANP) the employees work on in parallel may also be one of disturbing factors affecting the job satisfaction. It can be computed using formula (7) by dividing the sum of the sizes of all Teams by the total number of developers:

$$ANP = \frac{\sum_t NTM_t}{ND} \quad (7)$$

The remaining indicators in Table 3 are qualitative and can be obtained by surveying Team members at the end of each Sprint during the Sprint retrospective meeting.

The value of each indicator can be between 1 and 5, where 1 is the worst and 5 is the best mark. These indicators represent a subjective evaluation of job conditions by software developers and contribute to the overall picture of the development process. The target value of each indicator should not be less than 3; however, each organization can define its own target values considering its plans and goals.

3.3 Stakeholder 3: Customers

The main goal is "Satisfied customers" that can be measured through different indicators, e.g., the quality of product, price adequacy, reliability in terms of time and costs, completeness of product delivered at the end of each Sprint or release, flexible handling of changes in requirements, good collaboration with the development team, adequate training and documentation. Some of these indicators can be measured quantitatively (e.g., the quality of product and the completeness of product delivered at the end of each Sprint or release) and have already been considered through IT management indicators (e.g., error density, fulfillment of scope). However, the most of them are best covered by a questionnaire allowing the customers to express their subjective opinions. The survey can take place during the Sprint review meeting at the end of each Sprint or release and must contain questions that serve as metrics for each indicator.

4 Collection of metrics

In order to preserve agility, all the metrics proposed in the previous section (except the number of errors reported by the user after release) have been chosen in such a way that they can be collected during meetings already prescribed by Scrum, thus not requiring a substantial additional effort of the Team.

At the Sprint planning meeting the values of the basic parameters must be established: the Sprint length, composition of the Team (the number of the Team members, percentage of each Team member's engagement in the project), and costs of each Team member's engineering hour.

At Daily Scrum meetings the Sprint Backlog is maintained. For each task Team members report the amount of work spent and estimate the amount of work remaining. The amount of work spent is obtained simply when each Team member answers the question what he/she has done on the project since the last Daily Scrum. If a new task is added, the type of work performed and the cost of the engineering hour must be defined. For Team members not present the administrative days are recorded.

During the Sprint review meeting the number of errors reported by the user is recorded and a survey of customer satisfaction can be done.

During the Sprint retrospective meeting the code size of each PBI is measured and the numbers of PBIs/Tasks committed, but not completed are determined. However, these numbers can be computed on spot by an appropriate project management tool. At this meeting the survey of job satisfaction can also be done.

The computation of indicators is best done by an appropriate project management tool. Since tasks in the Sprint Backlog emerge as the Sprint evolves (e.g., a task that was only roughly defined at the beginning is split into several smaller ones) the tool should maintain a list of active tasks and keep history of all changes in order to compute the indicators properly.

5 Conclusions

We presented a metrics plan that enables monitoring and continuous improvement of the performance of the software development process. The plan consists of a set of indicators and corresponding metrics that measure the performance from the viewpoint of different stakeholders. The indicators and metrics are chosen considering the stakeholders' goals and the characteristics of Scrum.

The metrics plan can be implemented stepwise giving each software development organization freedom to adapt it to its specific needs. Nevertheless, we suggest the amount of work spent metric to be introduced first since it fits perfectly to the concept of Daily Scrum meetings and is analogue to the estimate of the work remaining metric already proposed by Scrum.

Indicators described in Section 3 were carefully chosen in order to be presented in a form of performance dashboards [18] providing timely information on software process performance. The development of such dashboards and the underlying business intelligence infrastructure is the aim of our further research.

6 Acknowledgement

This work was conducted within the research project MSM 6046070904 "Information and knowledge support of strategic management" funded by the Czech Ministry of Education.

7 References

[1] *CMMI® for Development (CMMI-DEV), Version 1.2. CMU/SEI-2006-TR-008*, Software Engineering Institute, Carnegie Mellon University, 2006.

[2] M. Ceschi et al., Project Management in Plan-Based and Agile Companies, *IEEE Software*, May/June 2005, pp. 21-27.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, *Agile software development methods*, VTT Electronic, Espoo, 2002.

[4] Manifesto for Agile Software Development, http://www.agilemanifesto.org/

[5] K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2004.

[6] L. Rising, N. S. Janoff, The Scrum Software Development Process for Small Teams, *IEEE Software*, July/August 2000, pp. 26-32.

[7] B. Schatz, I. Abdelshafi, Primavera Gets Agile: A Successful Transition to Agile Development, *IEEE Software*, May/June 2005, pp. 36-42.

[8] B. Upender, Staying Agile in Government Software Projects, *Proceedings of the Agile Development Conference (ADC'05)*, pp. 153-159.

[9] C. Mann, F. Maurer, A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction, *Proceedings of the Agile Development Conference (ADC'05)*, pp. 70-79.

[10] V. Mahnič, S. Drnovšček, Agile Software Project Management with Scrum, *EUNIS 2005 Conference – Session papers and tutorial abstracts*, University of Manchester, June 2005.

[11] M. Yap, Value based Extreme Programming, *Proceedings of AGILE 2006 Conference (AGILE'06)*, pp. 175-184.

[12] D. Hartmann, R. Dymond, Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value, *Proceedings of AGILE 2006 Conference (AGILE'06)*, pp. 126-134.

[13] T. Sulaiman, B. Barton, T. Blackburn, AgileEVM - Earned Value Management in Scrum Projects, *Proceedings of AGILE 2006 Conference (AGILE'06)*, pp. 7-16.

[14] F. Quentin, J. Koppelman, *Earned Value Project Management*, Third Edition, Project Management Institute, 2005.

[15] P. Kueng, Process performance measurement system: a tool to support process-based organizations, *Total Quality Management*, Vol. 11, No. 1, 2000, pp. 67-85.

[16] R. van Solingen, E. Berghout, *The Goal/ Question/Metric Method: a practical guide for quality improvement of software development*, McGraw-Hill, 1999.

[17] V. Mahnič, S. Drnovšček, Introducing agile methods in the development of university information systems, V. Lillemaa (ed.), *Proceedings of the 12th International Conference of European University Information Systems EUNIS 2006*, Tartu, June 2006, pp. 61-68.

[18] W. Eckerson, *Performance Dashboards*, John Wiley & Sons, Inc., 2006.

Viljan Mahnič is an Associate Professor and the Head of the Software Engineering Laboratory at the Faculty of Computer and Information Science of the University of Ljubljana.

Ivan Vrana is a Professor of Computer Science and the Head of the Department of Information Engineering at the Czech University of Life Sciences in Prague.