Measurement repository for Scrum-based software development process

VILJAN MAHNIC, NATASA ZABKAR
Faculty of Computer and Information Science
University of Ljubljana
Trzaska 25, SI-1000 Ljubljana
SLOVENIA

Abstract: - A measurement repository design for Scrum-based software development process is presented. A meta-model of Scrum is shown first that serves as a basis for the development of the repository data model. Then the mapping between CMMI practices and Scrum is presented for the Measurement and Analysis (MA) Process Area, including the specification of base and derived measures. This is followed by the description of the repository design for collecting and storing project data and measurement results.

Key-Words: - Scrum, CMMI, software measures, performance indicators, measurement repository

1 Introduction

In the last decade we are facing increased usage of agile methods for software development. According to the research from March 2007 [1], 69% of 781 respondents work in companies that adopted one or more agile techniques. The Standish Group 2006 research report states that 41% of agile projects succeeded as opposed to 16% of waterfall projects [17]. The most commonly used agile methods are XP and Scrum [12]. More than 14.500 ScrumMaster certificates were issued since 2003 [13] and the list of companies using Scrum includes IBM, Microsoft, SAP, Google and Yahoo [14].

At first glance, agile concepts seem to be in conflict with disciplined approach advocated by CMMI (Capability Maturity Model Integration [3]), but several authors suggest that it is possible to build organizational software process through a balance of agility and discipline [2, 16]. Synergy between CMMI and Scrum has been explored by providing mappings between CMMI and Scrum practices [10].

The aim of this paper is to demonstrate how CMMI requirements for Measurement and Analysis (MA) Process Area can be used in designing a measurement repository for Scrum-based software development process. We focus on the following specific practices:

- SP 1.1 Establish Measurement Objectives
- SP 1.2 Specify Measures
- SP 1.3 Specify Data Collection and Storage Procedures
- SP 1.4. Specify Analysis Procedures.

After a short description of Scrum concepts and meta-model we present mapping between CMMI and Scrum for the aforementioned practices. Based on our previous research in the Scrum usage and performance measurement ([7], [8], [9]), we then propose a set of measures that can be used to monitor satisfaction of different stakeholders involved in the software development process. Points on the process timescale are described where the proposed measures are collected without affecting the agility of the Scrum method. Finally, a generic data model of measurement repository for collecting and storing measurement results is presented.

2 Scrum Overview

2.1 Process Description

Scrum [11] is an iterative and incremental software development method driven by the Product Backlog list, which contains all active product requirements. The Product Backlog is managed by Product Owner, who is the only person authorized to change priorities of the requirements.

All work is done in Sprints. Each Sprint is an iteration of 30 consecutive calendar days and is initiated with a Sprint planning meeting, where the Sprint Backlog is agreed. This is a subset of Product Backlog requirements that defines functionality to be developed in the current Sprint. Every requirement is further broken into tasks that each takes roughly 4 to 16 hours to finish.

Functionality is developed by the Team, i.e. a group of developers that are collectively responsible for the success of each iteration and of the project as a whole. 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.

The ScrumMaster 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. Every day ScrumMaster leads a 15-minute Daily Scrum meeting where every 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? ScrumMaster is also responsible for resolving impediments encountered during the Sprint in order to assure smooth running of the development process.

At the end of the Sprint, a Sprint review meeting is held at which the Team presents Sprint results to the Product Owner. After the Sprint review and prior to the next Sprint planning meeting, the ScrumMaster also holds a Sprint retrospective meeting in order to ensure continuous improvement.

2.3 Meta-model of Scrum

For the purpose of introducing appropriate measures we present a meta-model of Scrum using the entity-relationship notation. The meta-model is further expanded in Section 4 in order to describe the design of the measurement repository that serves for storing project data and measurement results obtained during the development process.

In Fig. 1, we see that for each Project a Product Backlog must exist that contains several Product Backlog items (PBI). For each PBI one or more estimates of work remaining are provided. The Project is implemented through several Sprints.

For clarity reasons, the events associated with each Sprint (viz. the Sprint planning meeting, the Sprint review meeting, the Sprint retrospective meeting, and the Daily Scrum meetings) are shown as separate entities. For each Sprint a Sprint Backlog must be maintained that corresponds to PBIs the Team committed to implement during the Sprint. Each PBI is split into several tasks and for each task the estimates of work remaining are provided on daily basis. The implementation of a Sprint is appointed to a Team that consists of several members each of them being responsible for several tasks. For each project a Product Owner must be assigned (either being an employee or a customer representative) and for each Team a ScrumMaster must exist. Several Sprints can be combined into a release; however, the experience of the developers of some tools that support Scrum process [e.g., 15] has shown that a mandatory relationship between Sprints and a release represents a problem when a Team works on multiple releases within the same Sprint. Therefore, the relationship between Sprint and release entities is only provided through PBIs.

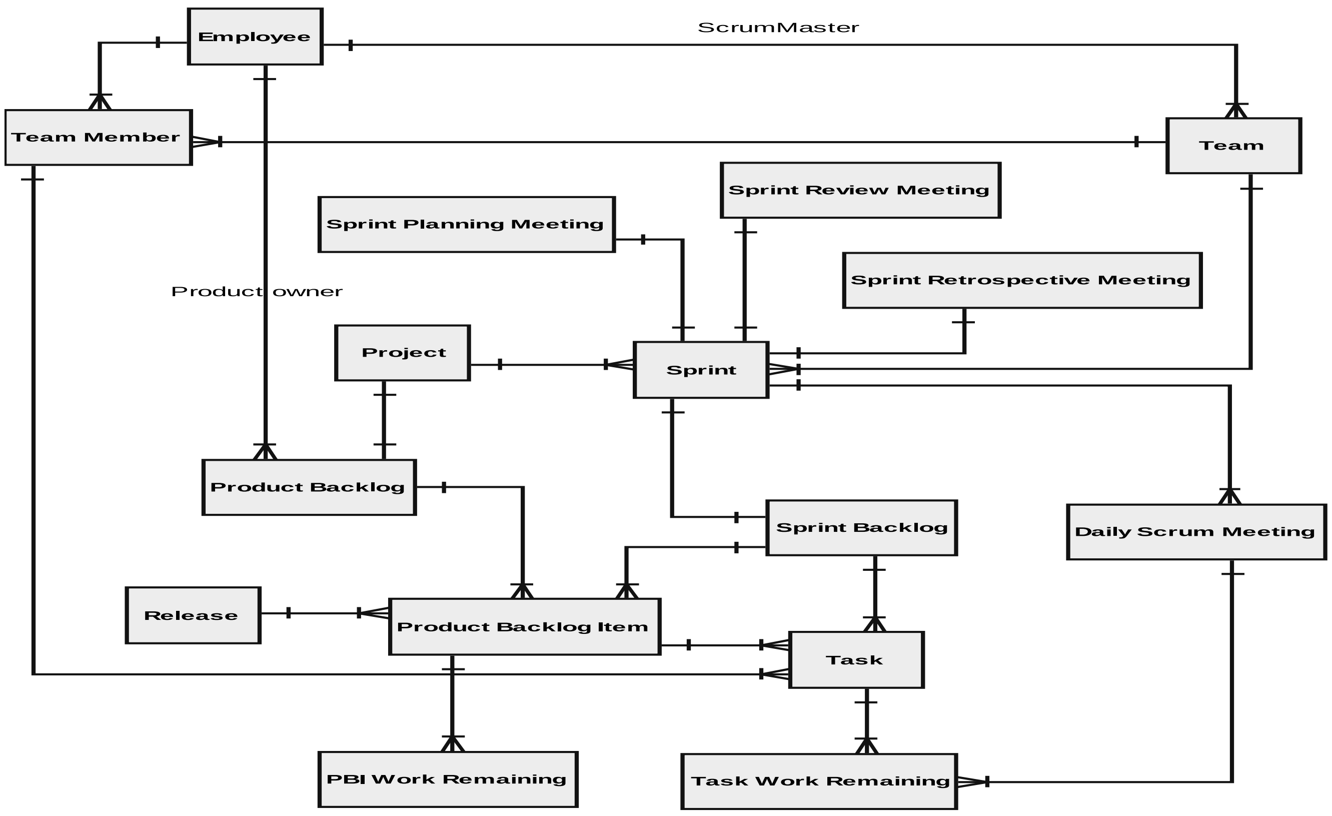

Fig. 1. Meta-model of Scrum

3 CMMI Practices Implementation

3.1 Establish Measurement Objectives

The main measurement objective is to monitor software process performance. Kueng [6] defines process performance as “the degree of stakeholder satisfaction”; therefore, the best performance is achieved when the goals of all stakeholders are satisfied. The achievement of goals should be measured quantitatively and qualitatively, thus giving a comprehensive view of the process performance. Based on our previous research [9], we have identified four stakeholders for Scrum process: IT management, Team members, ScrumMaster and Customers. Their goals are shown in Table 1.

Table 1. Stakeholders’ Goals

Stakeholder	**Goal**
IT management	Timely Information on Project Performance
	Quality Improvement
Team members	Job Satisfaction
ScrumMaster	Efficient Impediments Resolution
Customers	Customer Satisfaction

3.2 Specify Measures

According to CMMI, measures may be either “base” or “derived”. While data for base measures are obtained by direct measurement, data for derived measures come from other data, typically by combining two or more base measures. Derived measures serve as performance indicators showing the achievement of particular goals. In this subsection only base measures are described. Derived measures will be presented in subsection 3.4.

Originally, Scrum had only one base measure: 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 the Sprint Backlog (SB). In order to measure the achievement of goals identified in 3.1, we have defined some additional measures as shown in Table 2.

Table 2. Base Measures

Goal: Timely Information on Project Performance
Work remaining on day *d* for each task in the SB
Work spent on day *d* for each task in the SB
Goal: Quality Improvement
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)
Total number of PBIs committed 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
Goal: Job Satisfaction
Administrative days
Results of the survey (Job Satisfaction) 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.
Goal: Efficient Impediments Resolution
The number of impediments that refer to the given Task/Sprint/Team
The date the impediment was encountered
The date the impediment was resolved
Goal: Customer Satisfaction
Results of the survey (Customer Satisfaction) 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.

3.3 Specify Data Collection and Storage Procedures

All base measures proposed in previous section (including some basic parameters that must be established at the beginning of each Sprint) can be easily collected during meetings already prescribed by Scrum. The only exception is the number of errors reported by the user after release. Base measures collected at each type of meeting are shown in Table 3.

Table 3. Data Collection Points

Sprint planning meeting
Sprint length (number of working days in the Sprint)
The number of Team members (the size of Team)
Percentage of Team member’s engagement in the project
Cost of each Team member’s engineering hour
Daily Scrum meeting
Work remaining on day *d* for each task in the SB
Work spent on day *d* for each task in the SB
Administrative days
Impediment data
Sprint Review meeting
The number of errors found during the Sprint review meeting (for each PBI separately)
Results of the survey (Customer Satisfaction) conducted at the end of each Sprint/release
Sprint Retrospective meeting
The size of the code (for each PBI separately)
Total number of PBIs committed 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
Results of the survey (Job Satisfaction)

3.4 Specify Analysis Procedures

Base measurement data are grouped into derived measures or indicators that serve for analyzing software process performance in comparison to target values set by software development organization.

Achievement of the goal "Timely Information on Project Performance" is analyzed using the following indicators:

- Work effectiveness, which refers to the ratio between the work spent and the decrement of work remaining,
- Schedule Performance Index (SPI), which refers to the ratio between the earned value (i.e., the value of all tasks completed) and the planned value (i.e., the initial estimate of value of all tasks to be completed till a certain point within the project), and
- Cost Performance Index of labor costs (CPI) , which refers to the ratio between the earned value (measured in units of currency) and actual costs.

Indicators for goal "Quality Improvement" are:

- Error density, which refers to number of errors per KLOC (kilo-lines of code).
- Costs of rework, which refers to the product of hours spent on rework and cost of an engineering hour.
- Fulfillment of scope, which shows if all Product Backlog Items (PBIs) and Sprint Backlog Tasks have been implemented, and refers to the ratio between the number of tasks completed in the Sprint and total number of tasks in the Sprint Backlog or between the number of PBIs completed in the release and total number of PBIs committed.

Achievement of the goal "Job Satisfaction" is measured quantitatively and qualitatively through the following indicators:

- The average amount of overtime at Sprint/release/project level considering the expected hours, the amount of work spent and administrative days,
- The average number of projects the employees work in parallel,
- Qualitative evaluation of working conditions like communication and teamwork, physical discomfort, psychological well-being, workload, supervision, opportunities for growth, etc.

ScrumMaster's goal "Efficient Impediments Resolution" is measured by computing the average number of impediments per Task/Sprint/Team and the mean time for resolving an impediment (at Task/Sprint/Team level).

Indicators for goal "Customer Satisfaction" are measured quantitatively and qualitatively. The quality of product and the completeness of product delivered at the end of each Sprint or release can be expressed in terms of quality improvement indicators "error density" and "fulfillment of scope". Values of qualitative indicators are gathered from the survey allowing the customers expressing their subjective opinion regarding:

- price adequacy,
- reliability in terms of time and costs,
- flexible handling of changes in requirements,
- good collaboration with the development team,
- adequate training and documentation, etc.

Detailed descriptions and formulas for evaluation of aforementioned indicators can be found in [9].

4 Repository Design

In this section we present data model of the measurement repository for storing data that arise during the software development process (see Fig. 2). The model is derived from the meta-model in Fig. 1 by adding entity types that describe appropriate measures and enable the accommodation of measurement results. Beside, some new entity types are introduced in order to enable impediments tracking, describing the classification of tasks (regarding the type of work performed and current status), and keeping records of administrative days when a Team member is not at work.

The model is generic and does not prescribe in advance the kind and number of measures, thus enabling a stepwise introduction of the measurement program. New measures can be simply added and the measures that are no more needed or proved to be useless can be simply removed. The only prerequisite is that all measurement results are of the same type (viz. numeric). Each measure is represented as an instance of the `Measure` entity type (i.e., as a row of the corresponding relation containing measure key, name, description and other attributes), while the measurement results are stored in different relations, depending on the level and point in the process they are collected. E.g., measurement results that are obtained at the PBI level are stored in the `PBI_M`(easurement)`R`(esult) relation; values that are measured at the Sprint level are stored in the `Sprint_MR` relation, etc. Each row of these relations contains a compound primary key (a part of which is the measure key) and the measurement result.

Impediments tracking is introduced in order to provide data required for the computation of the average number of impediments per Task/Sprint/Team and the mean time for resolving an impediment (at Task/Sprint/Team level). Each impediment is described within the `Impediment` relation containing the impediment key, description, the dates when the impediment occurred and when it was resolved, and foreign keys providing relationship with the `Team` that encountered the impediment, the `Sprint` in which the impediment was encountered, and the `Employee` who was responsible for the resolution.

The classification of the type of work performed is necessary if we want to track the amount of different kinds of work during each Sprint, e.g., development, testing, rework due to error reported by the customer, rework due to the change in requirements, etc. Each row of the `Task Type` relation defines one of the aforementioned types of work, thus allowing each organization to specify the classification that best suits its needs. For the proper functioning of the measurement system it is necessary that each task in the Sprint Backlog is assigned the corresponding task type.

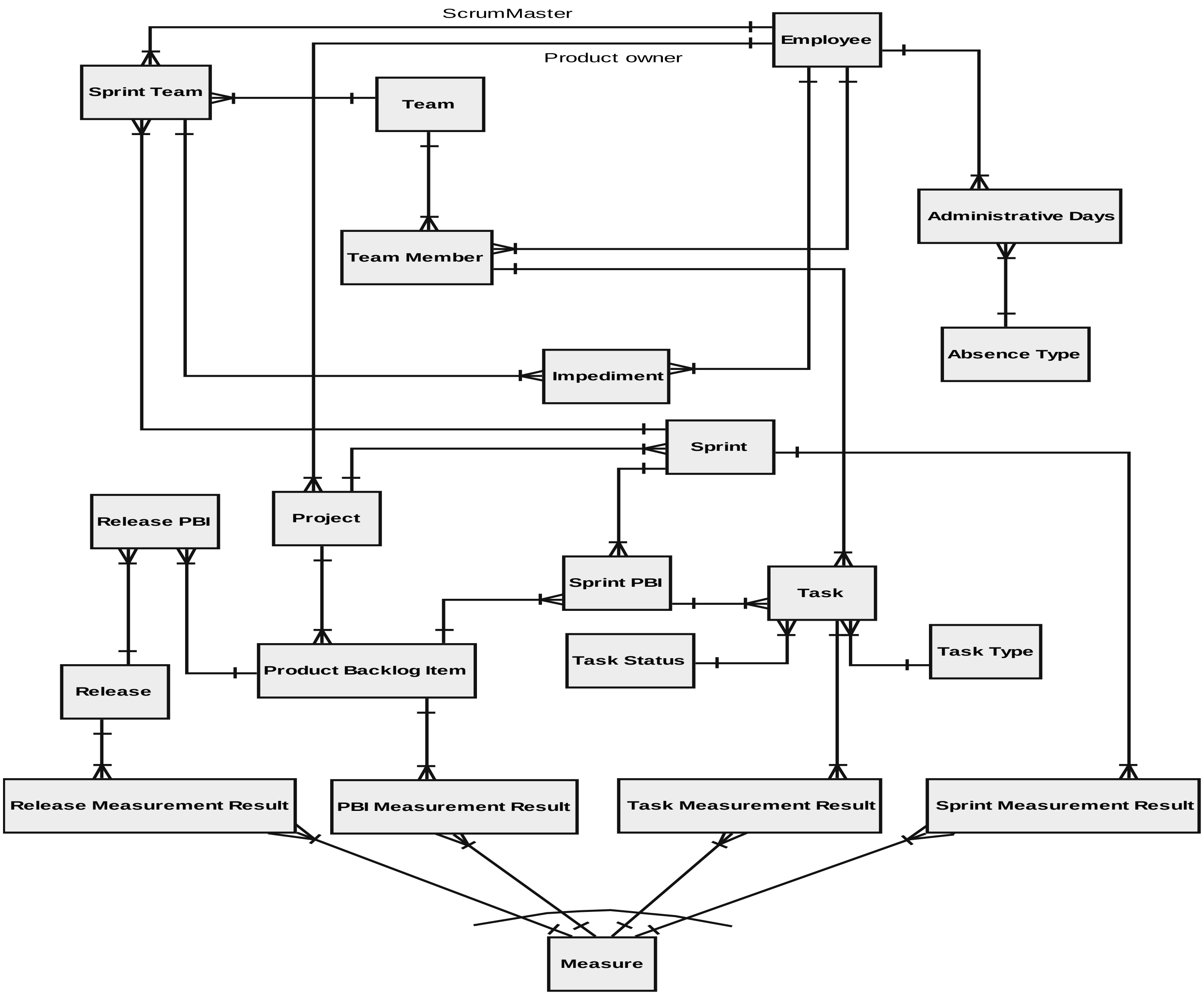

Fig. 2. Measurement repository design

In a similar way the classification of tasks according to their current status is introduced. Rows of the `Task Status` relation describe all possible statuses of a Sprint Backlog task, e.g., not started, in progress, completed, omitted, moved into next Sprint, etc. Again, each organization is allowed to specify different possible statuses according to its needs. The ScrumMaster maintains the status of each task during the Daily Scrum Meeting, at the same time when the amount of the work spent and the estimate of the work remaining for the task is entered.

Keeping records of administrative days (viz. sick days, vacation, course days, compassionate leave etc.) when a Team member is not at work enables exact calculation of overtime. When a Team member is absent (and consequently does not attend the Daily Scrum meeting), the ScrumMaster simply records his absence as a new row in the `Administrative Days` relation.

The `Absence Type` relation is necessary if we want to track different types of absence more in detail.

Due to limited space we do not describe explicitly the attributes of all relations; however, in order to calculate the labor costs precisely, the hourly rate of each Team member must be provided. In order to avoid problems with tracking history if the hourly rate changes frequently, it seems to be the best solution to record this attribute within the Task relation at the time when the task is created and a Team Member assigned to it.

A careful reader has also noticed that (in comparison to Fig. 1) all 1:1 relationships were removed and corresponding entity types merged into a single entity. Additionally, the 1:N relationships connecting `Product Backlog Item` to `Release` and `Sprint` were changed to M:N relationships in order to accommodate peculiar situations when the implementation of a Product Backlog Item requires more than one Release or Sprint. The M:N relationships were removed by the introduction of the `Release_PBI` and `Sprint_PBI` relations. Similarly, the `Sprint_Team` relation was introduced in order to support the concept of Scrum of Scrums that allows several Teams to work on the project within the same Sprint.

5 Conclusion

In this paper we presented the design of measurement repository that enables monitoring and continuous improvement of the performance of a Scrum-based software development process. Base and derived measures were defined considering characteristics of Scrum and practices prescribed by the Measurement and Analysis Process Area of CMMI. In order to preserve agility, all measures have been chosen in such a way that can be collected during meetings already prescribed by Scrum, thus not requiring a substantial additional effort of the Team. Derived measures serve as indicators for measuring achievement of goals of different stakeholders that are involved in software development process.

The results of this paper present the basis for the research of the use of modern concepts of business intelligence [5] in the area of agile software development. Data described by the proposed generic data model can be used as the main input to a data warehouse containing all data pertaining to the software development process organized in a way that enables detailed analyses through drilling-down and reporting techniques of business intelligence. Using indicators described in Section 3 performance dashboards [4] can be developed providing real-time information of the software process performance, thus enabling an immediate reaction in the case of deviation from target values.

References:

[1] S. Ambler, March 2007 Agile Adoption Survey posted, www.agilemodeling.com/surveys/ (4.12.2007).

[2] B. Boehm, R. Turner, *Balancing Agility and Discipline – A Guide for the Perplexed*, Pearson Education, 2004.

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

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

[5] M. Golfarelli et al., Beyond Data Warehousing: What's Next in Business Intelligence?, *DOLAP'04*, November 12–13, 2004, Washington, DC, 2004.

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

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

[8] V. Mahnic, S. Drnovscek, Introducing agile methods in the development of university information systems, *Proceedings of the 12th International Conference of European University Information Systems EUNIS 2006*, Tartu, June 2006.

[9] V. Mahnic, I. Vrana, Using stakeholder driven process performance measurement for monitoring the performance of a Scrum based software development process, *Electrotechnical Review*, Ljubljana (to appear).

[10] A. S. Marcal et al., Mapping CMMI Project Management Process Areas to SCRUM Practices, *SEW 2007*, *31st Annual Software Engineering Workshop*, Loyola College, Baltimore, MD, 2007.

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

[12] C. Schwaber et al., The Truth About Agile Processes, www.forrester.com (3.12.2007).

[13] Scrum Alliance, Scrum Alliance Membership Survey Shows Growing Scrum Adoption and Project Success, www.scrumalliance.org (28.9.2007).

[14] http://scrumalliance.pbwiki.com/Firms-Using-Scrum (4.12.2007).

[15] Danube, *ScrumWorks Basic User Guide*, http://www.danube.com/docs/scrumworks/1.8.2/userguide.html (6.4.2007).

[16] J. Sutherland et al., Scrum and CMMI Level 5: The Magic Potion for Code Warriors, *AGILE 2007*.

[17] T. Wailgum, From Here to Agility, 2007 http://www.cio.com (3.12.2007).