MMT – a Tool for Observing Metrics in Software Projects

Pekka Mäkiaho
Katriina Löytty
Timo Poranen
Faculty of Natural Sciences, University of Tampere, Finland

ABSTRACT

This paper presents the Metrics Monitoring Tool (MMT) that was developed in university graduate and undergraduate courses on software project work in 2014-2016. The tool aims to support project members, project managers and upper management in reporting and monitoring software and project metrics for their easier and more effective utilization. The paper covers the development process of the tool, evaluation assessment, its current composition and features. The paradigm applied in this study is Design Science Research and the methods for evaluation include prototype, expert evaluation, case study and technical experiment. Data was collected from the tool users by two questionnaires. As a result, MMT was evaluated to ease the metrics handling, while several aspects related to the richness of functionalities and usability still require further development.

Keywords: metrics, project management, software development, evaluation, monitoring, student courses, teaching, design science research.

INTRODUCTION

This article extends the paper MMT – a Project Tool for Observing Metrics in Software Projects (Mäkiaho et al., 2016).

Project work and basic project management skills are essential to all computer science students (Computer Science Curricula 2013, 2013). In many universities, group work skills are learned as a part of different course assignments starting from the first year of studies. During the third or fourth year, when students have studied enough core courses, there can be a larger capstone project as a stand-alone course.

In addition to group work and communication skills required in the capstone project, students have a possibility to combine their knowledge from different courses, like programming, databases, software and user interface design, into practice when they implement a software product. The student teams need to find a suitable combination of software tools (Portillo-Rodríguez et al., 2012) to utilize in the project for programming, requirements management, communication, user interface design and other main software development activities.

When course staff organizes different capstone projects to a larger group of students, there are challenges in following and supervising many projects at the same time. The projects also generate many standard metrics: whether a specific deadline was met or not, whether a specific deliverable was returned on time, how many hours of work has been done so far by different students, etc. Then, depending on the field of study, there can be many other process and product related metrics. In software development projects, there can be metrics like the number of written code lines, number of planned features to be implemented, number of features under development at a given moment, number of implemented features so far, number of passed test cases, etc.

The motivation for developing the Metrics Monitoring Tool (MMT) originates from the University of Tampere Project Work (PW) and Software Project Management (SPM) courses. During the courses, undergraduate PW students act as software development team members, and graduate SPM students act as project managers for those teams. The overall objective of the teams is to design and implement a functioning piece of software for a real client during one semester.

Prior to implementing MMT there were several challenges related to collecting and monitoring student projects' metric data. The metric data was gathered by using weekly reports submitted by project managers via a text based email template. The main challenges in this conduct included the high amount of manual work of project managers in aggregating the metric data, inconsistencies in and varying formats of the submitted data, lack of visibility to projects' overall progression versus their goals, as well as limited visualization capabilities.

The authors wanted to create a tool to help students to report their progress and to see the state of their projects by using project metrics. The aim was also to help the course supervisors to more easily see the progression of multiple projects. Thus far, the authors have not found any existing tool to fill with the requirements mentioned above, which would not force the use of some particular project management tool, and which can be used web-based.

This paper presents the tool that was created as a solution to these issues; MMT is a software for observing and visualizing project metrics in software development projects. The tool helps project managers in reporting, team members to be more aware of the state of the project and the course staff to compare and follow how all projects are progressing. The requirements did not include the features of a typical software project management tool, like project planning and scheduling, resource allocation or change management. Thus the MMT-tool cannot be called a project management tool and that is the reason, why the authors do not compare it to the common project management tools, like JIRA or Redmine. Based on the research needs, for example, the tool can be extended to collect new data in future. The authors present here the evolution process of the tool from the proof of concept version to the current version that is in production.

The rest of the paper is organized as follows. In the next section, a brief introduction to software project metrics is given. The Methods-section tells how the Design Science Research (DSR) -paradigm (Hevner et al., 2004) was applied to this research. Then, the development process and the implementation of MMT is described. After that the evaluation phases are introduced. The final section concludes the work.

SOFTWARE PROJECT METRICS

Software project management consists of five phases: *initiating*, *planning*, *executing*, *monitoring and controllin*g, and *closing* (PMBOK, 2013). One of the most important reasons that a project fails is poor reporting of the project's status (Charette, 2005). Because of the poor reporting, the management does not know the state of the project and thus does not execute the right actions. So projects often fail in the monitoring and controlling phases, as the management does not know the state of the project.

Software metrics can be used for measuring the characteristics of 1) a piece software, 2) a software development project, or 3) a software development process. According to Goodman (1993), utilizing software metrics entails “the continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products”.

Moreover, software metrics focus either on *controlling* or *predicting.* The first relates to software development *processes* and the latter to software *products* (Sommerville, 2010). A controlling metric can for instance measure the average time used for fixing a bug whereas a predicting metric could focus on the total lines of code. Further, controlling metrics can be targeted at measuring the process or measuring the project (Fenton, 1991). Process metrics would support strategic decision making while project metrics lean towards supporting more tactical decisions (Fenton, 1991) on project progression, for example.

METHODS

The paradigm chosen for this research is Design Science Research. According to the design science principles (Hevner et al., 2004; Peffers et al., 2007) the researchers build an artefact to solve a problem and then evaluate how it succeeded. DSR usage in information systems development is defended by many researchers (Gregor & Jones, 2007; Hevner & Chatterjee, 2010; Hevner et al., 2004; March & Smith, 1995; Nunamaker et al., 1991; Peffers et al., 2007). In this research the researchers design, build and evaluate a software tool (MMT) that can be used for observing software project metrics. The research data can be divided into three classes: the data from observing the activities of the students in the Project Work courses, the documentation from the developing phases of the tool, and the data from the evaluation rounds of the software.

The activities of the students in the Project Work courses have been observed since the academic year 2011-2012. The researchers have collected metrics from the weekly reports, and on how deadlines have been kept. They have also set up Moodle questionnaires for finding out which software tools have been used in the projects, which metrics the students have observed, and what kind of processes they have followed. From the fixed-form weekly reports the researchers have read metrics on the statuses of requirements, the number of test cases, and the working hours done in the projects. In addition, Mäkiaho has acted as a supervisor in most of the projects and thus got data from the projects’ progression in the formal project reviews.

The MMT software has been developed in the Project Work courses since the fall semester 2014. During each development iteration, a piece of working software, development documentation, testing documentation and the final report have been produced. Löytty participated in the Software Project Management course as a student during the fall semester 2015 and was a project manager of the team that implemented the first production pilot version of MMT (1.0). Mäkiaho and Poranen have acted as the client in these MMT projects and participated to the planning and testing phases.

In principle, the development process of MMT followed an iterative cycle where evaluation of early versions of the design fed into further developments of the solution (Hevner et al., 2004; Simon, 1996 cited in Hevner et al., 2004). According to the guidelines of DSR, design is a "search process" for finding a satisfactory solution to a problem, which "requires utilizing available means to reach desired ends while satisfying laws in the problem environment" (Hevner et al., 2004, p.83). The available means include "actions and resources" as well as knowledge of the application and solution domains. The desired ends mean the "goals and constraints" of the design. The laws consist of "uncontrollable forces" of the business environment. (Hevner et al., 2004, p.88)

THE DEVELOPMENT PROCESS AND FUNCTIONALITIES OF MMT

MMT has been developed iteratively in various phases by several teams as part of the PW, SPM, and PP (Programming Project) courses, where a preceding team generally hands the work over to the next team. The following paragraphs discuss the development, or Develop/Build–Justify/Evaluate (Hevner et al., 2004) cycle during which the tool has evolved from the initial proof-of-concept version 0.10 to the current version 3.0. Figure 1 provides a high level illustration of this process incorporating the DSR aspects of means, ends and laws as well as the recurring steps of assessment and refinement (Hevner et al., 2004).

The first construct of the tool was initiated during the fall 2014 through the summer 2015. This early construct, a proof-of-concept version (versions 0.10-0.19), provided knowledge for upcoming work on desired features, deployable methods and anticipated pitfalls. Development of these versions faced some significant issues in project communication, requirements management, programming work and code quality. Also some of the design decisions made during the proof-of-concept phase were deemed unsuitable after the client testing and evaluation.

Based on the evaluation results where the initial versions were judged to be of too low maturity for further development, the design was started afresh with revisited requirements, revised database design and a decision to go forward with a feasible model-view-controller (MVC) framework. This stage provided the basis (versions 0.2-0.9) for the first version (version 1.0) of the solution to go live with piloting use in the spring 2016. This version included the very basics of the required functions. Also some usability issues and bugs remained unsolved in this version of the solution. However, the overall maturity level was perceived as high enough for initial production deployment and piloting.

Several bug fixes and usability improvements were implemented in the subsequent versions (versions 1.1-1.3) during the same spring, such as allowing users more flexibility in editing their working hours.

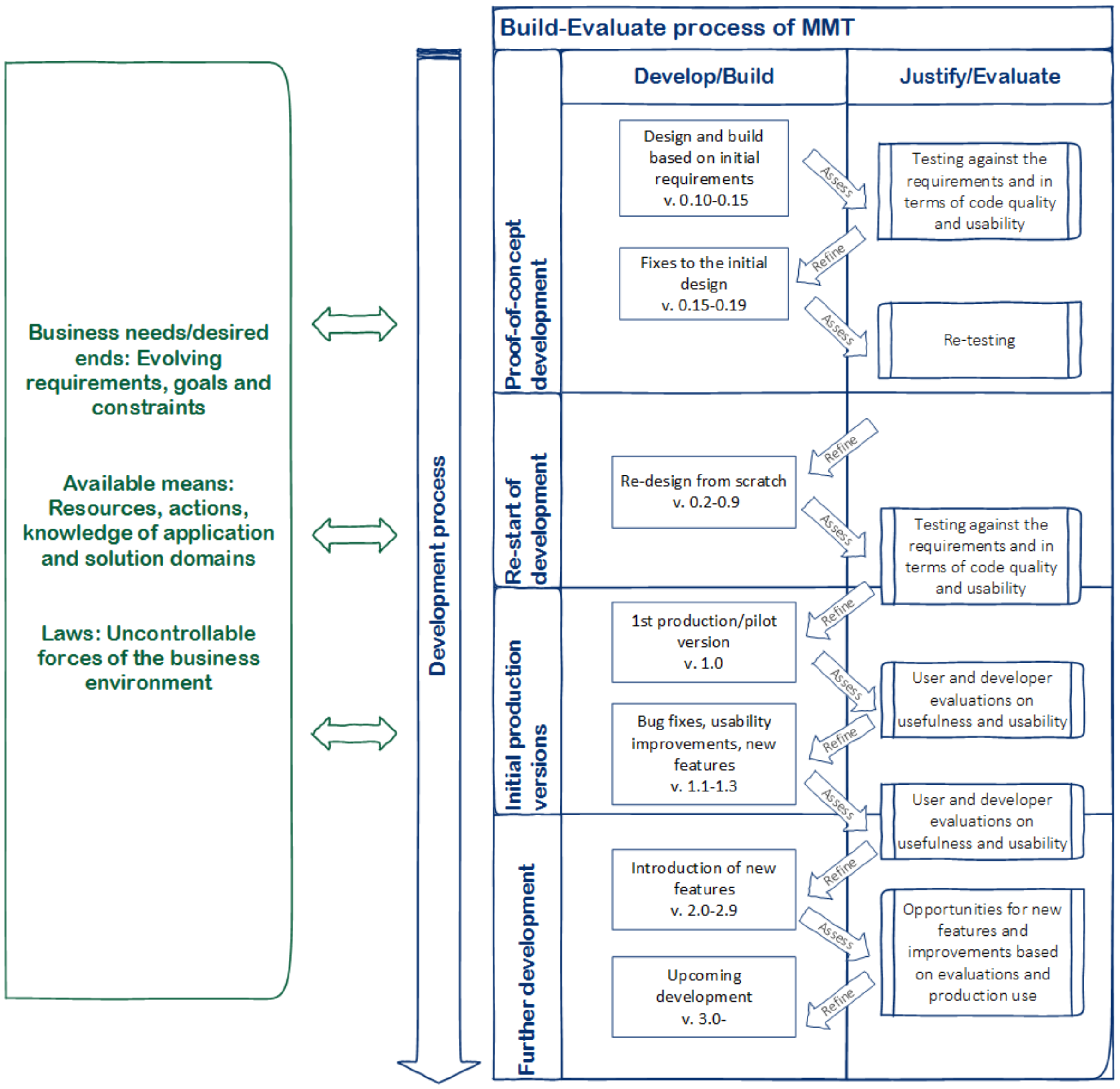

Build-Evaluate process of MMT
Develop/Build
Justify/Evaluate
Business needs/desired ends: Evolving requirements, goals and constraints
Available means: Resources, actions, knowledge of application and solution domains
Laws: Uncontrollable forces of the business environment
Development process
Proof-of-concept development
Re-start of development
Initial production versions
Further development
Design and build based on initial requirements v. 0.10-0.15
Fixes to the initial design v. 0.15-0.19
Re-design from scratch v. 0.2-0.9
1st production/pilot version v. 1.0
Bug fixes, usability improvements, new features v. 1.1-1.3
Introduction of new features v. 2.0-2.9
Upcoming development v. 3.0-
Testing against the requirements and in terms of code quality and usability
Re-testing
Testing against the requirements and in terms of code quality and usability
User and developer evaluations on usefulness and usability
User and developer evaluations on usefulness and usability
Opportunities for new features and improvements based on evaluations and production use
Assess
Refine

Figure 1. Build-Evaluate process of MMT (adapted from Hevner et al., 2004; Simon 1996 cited in Hevner et al., 2004)

The version 2 has been implemented during the summer semester 2016 and the developing continues during the fall semester. Additional functional features have been introduced at this stage, like the possibility for giving feedback to project teams on their weekly reports. Further similar enhancements have continuously been implemented by students based on feedback collected from users. The current deployment version is 2.9, which is expected to be upgraded to version 3.0 by the end of the year 2016.

Currently MMT runs on a virtual server provided by the University of Tampere School of Information Sciences (School of Information Sciences, University of Tampere, 2016). It relies on an SQL database and is implemented in PHP. MMT utilizes CakePHP, an open source MVC framework for PHP, and Highcharts, a JavaScript library for presenting various kind of plots and charts.

Figure 2 provides a high level description of MMT's database and its relations: MMT contains users and projects. The users have inherent user roles, and additional project roles according to the members table. The projects have weekly reports and their related metrics stored in the respective tables. Similarly, the members have weekly and daily working hours. The tool presents projects' metrics, progression and status in graphs derived from data in the weeklyreports, metrics, workinghours and weeklyhours tables. Worktypes and metrictypes tables define the types of working hours and metrics that the tool uses. The notifications, newreports and comments tables are designed to allow for notifying of new reports and commenting functionalities.

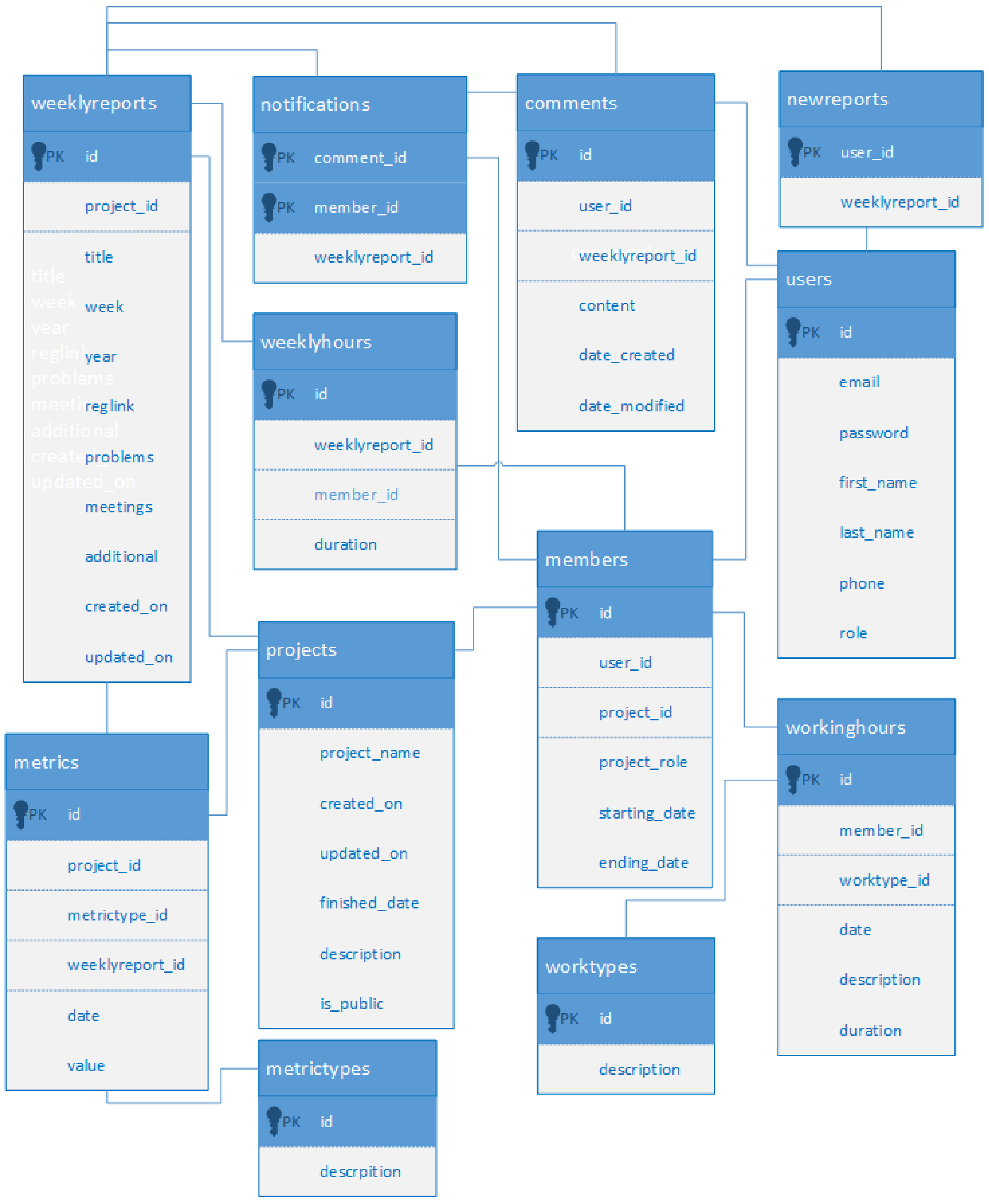


Figure 2. Database diagram of the solution

The current tool (version 2.9) includes the following features based on user and project roles: A public projects listing, public statistics and frequently asked questions (FAQs) are offered for all visitors of the MMT web site. Project developers can view reports and charts of their own and public projects as well as log and update their own daily working hours. Project managers can additionally compose weekly project reports, log hours on behalf of team members and assign and remove members of their projects. Project supervisors have similar rights as managers bar the hour logging functionality. Any project member and supervisors can comment on the weekly reports and give feedback about the tool to its developers. Additionally, the supervisor sees new, unread weekly reports on his or her page. In the near future, a notification function of unread comments on one's projects will also be introduced. Finally, a new user role, project client, is upon implementation. User and project basic data in the tool is maintained by admin users. The general look and feel of the tool is shown in Figure 3.

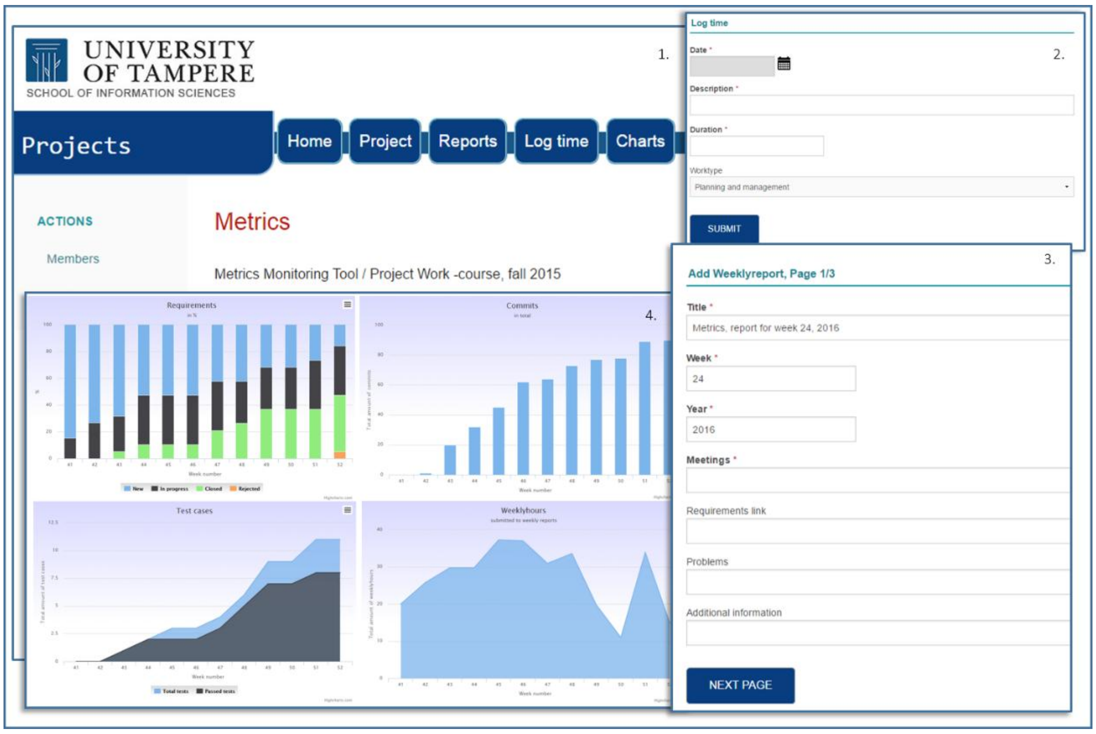
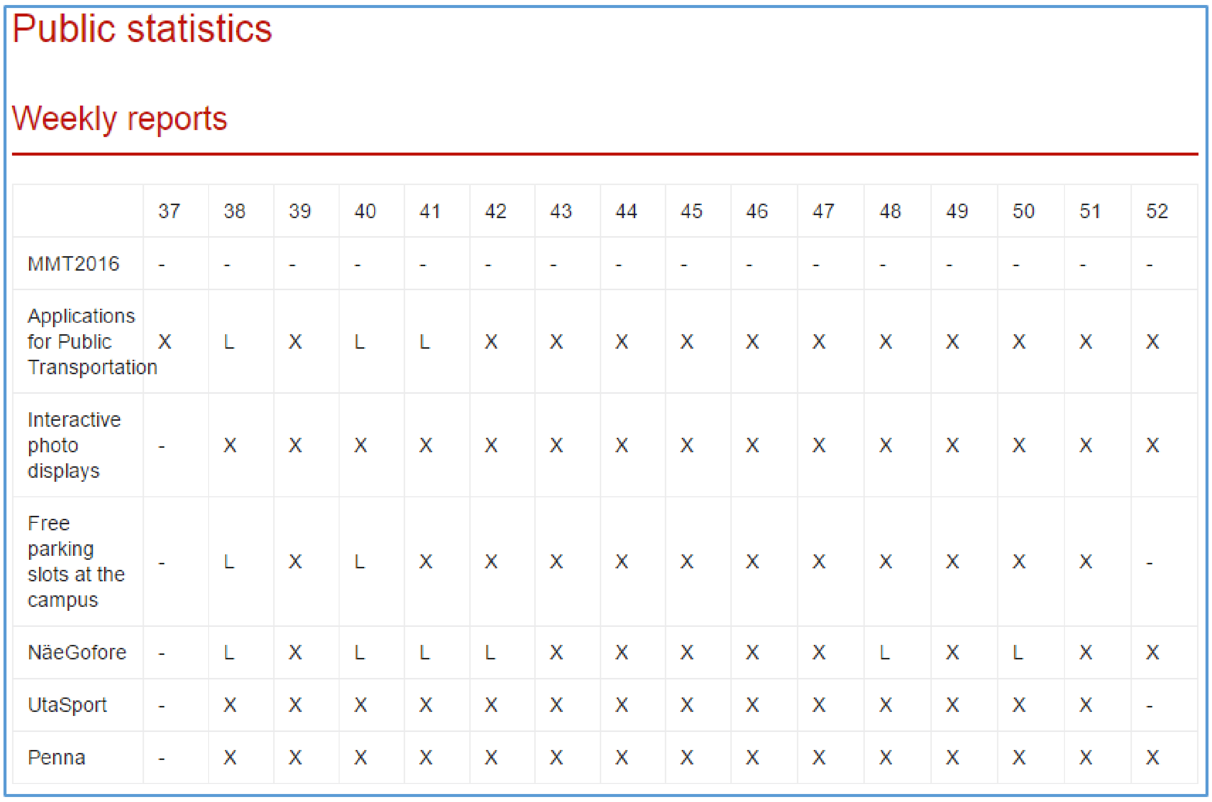


Figure 3. Layout and feel of MMT

Figure 4 shows the supervisor's view of weekly report statistics, where "X" denotes a weekly report submitted on time, "L" a late delivery of a report, and "unread" a report that the supervisor has not read yet.


Public statistics

Weekly reports

	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
MMT2016	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Applications for Public Transportation	X	L	X	L	L	X	X	X	X	X	X	X	X	X	X	X
Interactive photo displays	-	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Free parking slots at the campus	-	L	X	L	X	X	X	X	X	X	X	X	X	X	X	-
NäeGofore	-	L	X	L	L	L	X	X	X	X	X	L	X	L	X	X
UtaSport	-	X	X	X	X	X	X	X	X	X	X	X	X	X	X	-
Penna	-	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 4. Supervisor's view of weekly report statistics

MMT generates the metric data from the weekly reports submitted by project managers. The current metrics include:

- the number of current sprints vs. planned sprints,
- the number of requirements in the statuses of New (i.e. unprocessed), In progress (i.e. currently being worked on), Closed (i.e. implemented) and Rejected (i.e. cancelled altogether),
- the cumulative number of code commits,
- the number of total test cases vs. passed test cases, and
- the number of working hours per person, team and work type

Additionally, the timeliness of submitted weekly reports (see Figure 4) can be used as a metric in monitoring the teams' performance from a teaching perspective. In future versions, derived metrics, such as traffic lights on project success, are designed, which utilize the data from the weekly reports in a more versatile way. MMT visualizes the metrics as graphs, as was portrayed in Figure 3. The user can also view the data in textual form in the weekly reports submitted by the project managers.

EVALUATION

Peffers et al. (2012) classified six artefact types that can be built in DSR: *algorithm*, *construct*, *framework*, *instantiation*, *method*, and *model*. MMT is an instantiation as it is a piece of software that will be put to real use in a real organization. They also classified the types of evaluation methods: logical argument, expert evaluation, technical experiment, subject-based experiment, action research, case study and illustrative scenario. Table 1 shows which evaluation methods were used for each version of MMT.

Table 1. Evaluation methods for each MMT-versions

Main version/Evaluation Method	**0**	**1**	**2**
Prototype	Prototype versions 0.1 and 0.2.	-	
Expert evaluation	By supervisors/client.	By supervisors/ client and the developing team.	By supervisors/client and the developer.
Case study	-	Piloting in real environment.	Production use in real environment during fall semester 2016.
Technical experiment	Functional testing by the team. Experiments by the supervisors/client.	Functional testing by the team. Experiments by the client/supervisor.	Functional testing by the team. Acceptance testing by the supervisors/client.

Evaluation of the version 0 – Proof of Concept

The evaluation of the 0-versions (0.1-0.19) is based on the functional testing made by the developing team with the version 0.15. The client read the test report and according to him, there were too many defects to put the software to use. The team also proposed to correct bugs and some usability-issues before this could be put to (pilot) use.

The decision made in the spring 2015 was to continue the development. The versions 0.16-0.19 were implemented during the summer 2015. The client used the versions of software during the development and after getting the final version (0.19) of this phase. According to the experiments, some of the features worked ok but there were major usability issues and a lot of bugs. The test and final reports written by the developers further confirmed that this version could not be put to pilot use yet.

A new project team was formed among the Project Work course students in the fall 2015. One of the team's very first tasks was to evaluate the maturity of the current version including the source code and the documentation. After the evaluation, a decision was made, together with the client/supervisors, to start the building of the next version from scratch.

Evaluation of the Version 1 – Piloting

Piloting the tool in production environment was started with the version 1.0 after the supervisor/client had accepted it based on the testing report and the recommendation by the project manager. After the piloting period, the solution was evaluated, which is covered in the following paragraphs.

The data was collected from the Software Project Management and Project Work courses at the School of Information Sciences, University of Tampere during the spring semester 2016. The projects started in January and ended in May. There were 4 project teams, 8 students (project managers) on the Software Project Management course and 15 students (developers) on the Project Work course, who used the pilot version of the tool.

Using MMT was mandatory to all teams for, at least, logging hours and weekly reporting. During the course, the teams had 5 reviews with the supervisor and the client and they also formed a final report at the end of the course. The supervisors observed the project teams by using MMT, participating in reviews and communicating in face-to-face -meetings and by email.

There were also two mandatory Moodle questionnaires for the students with questions on MMT and its usage. The first questionnaire was executed in the beginning of March, after the projects had started. The second questionnaire took place at the end of May, when all the projects had finished. The versions of MMT are related to the questionnaires so that the version 1.0 was in use before the first questionnaire. MMT was updated to the versions 1.1 and 1.2 before the second questionnaire. The version 1.3 was deployed just after the second questionnaire.

The four main features of the tool were logging hours, composing and viewing weekly reports and viewing the visual charts of the project metrics. Table 2 shows the average of the grades given by the students to each feature and MMT as a whole regarding usability, functionality and usefulness as well as the total grade. The scale was (1) poor, (2) fair, (3) good, (4) very good and (5) excellent.

Table 2. The average grade (1-5) given to each feature and to the feature's attributes

Feature/ Attribute	Logging hours	Viewing reports	Viewing charts	Composing reports	MMT-tool as a whole
Usability	4.5	4.3	4.75	4.3	3.2
Functionality	4.5	4.0	4.5	4.0	3.6
Usefulness	5.0	4.0	4.5	4.3	4.0
Overall	4.2	3.9	3.7	3.3	3.8

Logging Hours

Students were asked to use MMT for logging their working hours: whenever they worked for the project, they logged the date, the hours, the type of work, and the description in MMT. As this feature was mandatory to use for all students, it is not surprising that most of the comments related to this. When a weekly report was composed, the hours were copied to the weekly report's hours. Because of this, there were restrictions in editing the logged hours afterwards: In the version 1.0, a developer was not allowed to change the logged hours afterwards but had to ask his/her manager or supervisor to do it. In the questionnaire 1 this was complained about by two managers and two developers. This restriction was changed to version 1.1 so that also developers were allowed to edit the hours until the weekly report was sent. However, also this restriction was seen as a problem by one developer.

Overall, the logging hours -feature was seen as very useful: The managers said that it saved their time in comparison to a situation where they would have had to collect the hours manually or extract them from some other tool for the report. The developers reported that logging hours helped them to observe their own time usage.

Viewing Reports

The only reports one could get from MMT are the weekly reports composed by the managers. Viewing (weekly) reports was not seen very useful by the developers. However, there were comments on extending this feature: project managers from all teams, and also five developers, requested additional features for getting different kinds of reports from MMT, especially on working hours, by using various filters.

It was extremely useful for the supervisors to have all the reports from different weeks and different projects in one tool. Getting this information from one place reduced the time required for preparing to project meetings from 60-90 minutes to 15 minutes.

Viewing Charts

The charts show the state and the proceeding of the project in one view. One can see the current state and the history of the working hours, requirements, test cases and commits. For the supervisor, this feature is very important, as from the graphs one can not only see the history and the current state but also predict the project's progression and give some suggestions to the team.

The project members did not see the charts very useful now, but they would like to have additional charts on individual level, especially for seeing how the amount of working hours has built up. The managers from all teams reported this feature to be very useful for seeing the status and proceeding. However, one team commented that viewing charts was not so useful from a project management point of view but it was interesting to compare their own project to other projects.

Composing Reports

Composing weekly reports is done by the project managers by the following Monday of each week. The managers reported the amount and states of test cases and requirements as well as the amounts of weekly hours per person, commits to the code repository, meetings and possible issues.

When creating a weekly report, each project member's working hours were automatically copied to the report as a template, and then the managers could modify these if needed. So, in the system there were reported weekly hours and working hours. This was seen very confusing. There were also some other usability issues. However, the feature was seen useful and it was reported to make reporting easier compared to writing and sending emails. In the previous semesters, the reports were sent by email and the average time for composing the report was 42 minutes. Now the average time was 16 minutes, varying from 2 to 30 minutes.

Evaluation of the Version 2 – Production Use

Versions from 2.0 to 2.9 have been developed iteratively. At the end of the sprint the developers run the test cases in development environment. If the tests pass, the software is put to acceptance testing in the test server, which is identical with the production server. The acceptance test is done by the client and if the maturity of the software is good enough, the software is deployed to the production server.

The use of MMT is observed during the fall semester 2016, two Moodle questionnaires are set up for the students and the clients are interviewed. In the beginning of the year 2017, a similar analysis will be done as was done for the version 1.

To Be Developed

During the evaluation of the version 1, the researches asked in both questionnaires, which properties or features of the tool should be further developed. In several of the general comments, there were requested on using Bootstrap (Bootstrap) or a similar technology for increasing usability and support for mobile interface.

Restricting the editing of the logged hours afterwards was seen as a problem by both managers and developers. In the next versions, the restrictions could be removed.
The reports seen from the tool are limited only to the weekly reports composed by the managers. Also the amount of different types of charts is quite limited. More flexible reports and charts could be available in the next versions.

Developing the tool into a real project management tool, like Redmine (Redmine), for not needing to use many different tools in the same project, was suggested by four managers. One manager asked for a feature that could predict the success of the project and to give tips to do the right actions. This feature has been in the product backlog from the very beginning and will certainly be implemented in the future.

CONCLUSIONS AND FURTHER WORK

Metrics Monitoring Tool benefits different user groups. Logging hours feature helps the project members to observe their own time usage. MMT reduces the time for creating weekly reports. Visualizing the metrics helps to monitor the projects. Especially from the supervisor’s point of view: it is easy to catch, for example, if the amount of commits or working hours is not increasing fast enough, or if the requirements stay too long in certain states. When using MMT in teaching of software project work or in software development, there could be some material (lecture, video, online help) to introduce the tool for the users to help and to motivate them to use useful features.

Using the tool will be continued during the next semesters and new features will be developed on the base of this evaluation. It could also be worth researching further how the new features, like interactive commenting of weekly reports (implemented in version 1.3) or predicting the project’s state (proposed to be developed) would help in teaching and software development.

REFERENCES

Bootstrap. *A framework for developing responsive, mobile first projects on the web.* Retrieved October 30, 2016, from http://getbootstrap.com/.

Charette, R.N. (2005). Why Software Fails. *IEEE Spectrum.* Retrieved October 28, 2016, from http://spectrum.ieee.org/computing/software/why-software-fails.

Computer Science Curricula 2013. (2013). *The joint task force on computing curricula*. Association for Computing Machinery (ACM) and IEEE Computer Society.

Fenton, N. (1991). *Software Metrics - A Rigorous Approach*. London, UK: Chapman & Hall.

Goodman, P. (1993). *Practical Implementation of Software Metrics*. London, UK: McGraw Hill.

Gregor, S., & Jones, D. (2007). The anatomy of a design theory. *Journal of the association for information systems*, *8*(2), 312-355.

Hevner, A., & Chatterjee, S. (2010). *Design research in information systems. Theory and practice*. New York: Springer Publishing.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly, 28*(1), 75-105.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, *15*(4), 251-266.

Mäkiaho, P., Löytty, K., & Poranen, T. (2016). MMT – a Project Tool for Observing Metrics in Software Projects. In *International Conference on e-Learning – Proceedings from 2016 (e-Learning'16)* (pp. 80-85). Bratislava, Slovakia: ETN FETCH.

Nunamaker, J. R., Chen, M., & Purdin, T. (1991). Systems development in IS research. *MIS Quarterly, 7*(3), 89-106.

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design Science Research Evaluation. In K. Peffers, M. Rothenberger, &, B. Kuechler (Eds.), *DESRIST 2012 (LNCS)* (Vol. 7286, pp. 398-410).

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information System*, *24*(3), 45-77.

PMBOK. (2013). *A Guide to the Project Management Body of Knowledge. 5th edition.* Project Management Institute.

Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., & Beecham, S. (2012). Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology, 54*(7), 663-685.

Redmine. *Web-based project management and issue tracking tool.* Retrieved October 30, 2016, from http://www.redmine.org.

School of Information Sciences, University of Tampere. (2016). *Metrics Monitoring Tool*. [Software] Available at http://metricsmonitoring.sis.uta.fi/. Accessed October 28, 2016.

Simon, H. A. (1996). *The Sciences of the Artificial. 3rd edition*. Cambridge, MA: MIT Press.

Sommerville, I. (2010). *Software Engineering. 9th edition.* Addison-Wesley.