12-15-2012

Towards an integrated model for managing product and process quality in agile software projects

Sebastian Gruschwitz

Frank Schlosser

Towards an integrated model for managing product and process quality in agile software projects

Research-in-progress paper

Sebastian Gruschwitz
University of Bamberg
sebastian-alexander.gruschwitz@uni-bamberg.de

Frank Schlosser
University of Bamberg
frank.schlosser@uni-bamberg.de

ABSTRACT

Many software projects still experience delays, exceed budget or fail to deliver the expected quality due to poor project management, often caused by a lack of information about the real status of the project. This is particularly problematic in agile projects with their dynamic team configurations, high number of iterations and short development cycle times. A key challenge to effectively and efficiently manage agile projects is to select and implement both the right product and process quality metrics. We develop a catalogue of 40 metrics covering different product and process quality criteria. The catalogue is then used to select and evaluate a specific set of metrics that are implemented in an agile software development project. Our preliminary findings show that while the combination of product and process quality metrics is important, more research into their interdependencies and selection criteria is needed.

Keywords

Agile software development, software product quality, software development process quality.

MOTIVATION

Despite many years of academic studies and practical experience many IT projects are characterized by poor quality, schedule overruns and high costs, and thus fail or at least do not deliver all expected benefits (Southekal and Levin, 2011). The reasons are manifold, e.g. unrealistic planning, poor project management or change requests leading to scope creep (Nelson 2007). With agile software development becoming more and more popular, there is an increasing need to look at both advantages and challenges that come along with these new approaches. In general, agile software development projects are expected to better account for change requests during implementation, thus being less rigid compared to traditional approaches (e.g., waterfall). However, this flexibility needs to be controlled in a way that does not significantly increase project management efforts. Many metrics that are used in traditional software development are not applicable to agile projects because of the large number of small releases. Even within an agile project the effectiveness of metrics may change over the course of several iterations as the software matures (Olague, Etzkorn, Gholston, and Quattlebaum 2007). Companies and project teams often face various problems in agile IT projects, e.g., poor and/or unclear performance indicators, lack of holistic information available for making suitable corrective actions, lack of responsibilities, and high complexity (Southekal and Levin, 2011). Agile software development is regularly critiqued for leaving too little place for external oversight of projects (Talby and Dubinsky, 2009). Often, metrics are not chosen and implemented appropriately, and also are directed too much towards measuring product quality while giving too little attention on process quality. In this study, based on existing literature and expert interviews we investigate what metrics help to control agile IT projects in terms of both product quality and process quality. We conduct a case study to learn how the right set of metrics can be chosen and implemented, thus improving project reporting effectiveness. While there will be lessons learned regarding the usefulness of specific metrics within our project, these metrics have to be chosen for each IT project based on a number of factors, e.g., scope and character of the considered project, scale of product to be developed, applied process model, and relevant quality characteristics. Nevertheless, our metrics catalogue can serve as a helpful foundation for these decisions. Also, depending on the characteristics of a project and the available resources, the number of metrics to be implemented needs to be thoroughly considered and balanced to ensure both product and process quality can be managed during the project.

APPROACH AND METHODOLOGY

This study consists of several steps which are presented in Figure 3. Having started with a literature review on product and process characteristics and related metrics in agile software development projects (phase 1), we conducted a number of expert interviews to validate and further extend the initial metrics catalogue (phase 2). All interviewees in phase 2 work in a large international consulting company and are experienced in agile IT project management. Next, we developed a consolidated metrics catalogue (phase 3) intended to serve as a foundation for the selection of project-specific metrics. We adopted a single-case study approach within the same consulting company to test our catalogue. The chosen IT project consists of around 150 employees and is part of a large, multi-year transformation program in a German government institution. Eight project team members holding different positions (manager, team leaders, scrum master, …) have been selected to support us during phases 4 to 6 by giving interviews and feedback, managing the processes of metric selection (phase 4) and implementation (phase 5), and supporting data collection as well as metric evaluation (phase 6). As can be seen in Figure 3, phases 1 to 5 have been successfully completed and the selected metrics are now being used in the project. Data collection is intended to take place until end of Q1/2013 in order to have a sufficient set of data, plus feedback from the project team which should allow us to conduct sound analyses and come up with a set of valuable insights into the selection and implementation of product and process quality metrics, and lessons learned on what are key success factors for effectively managing product and process quality conjointly.

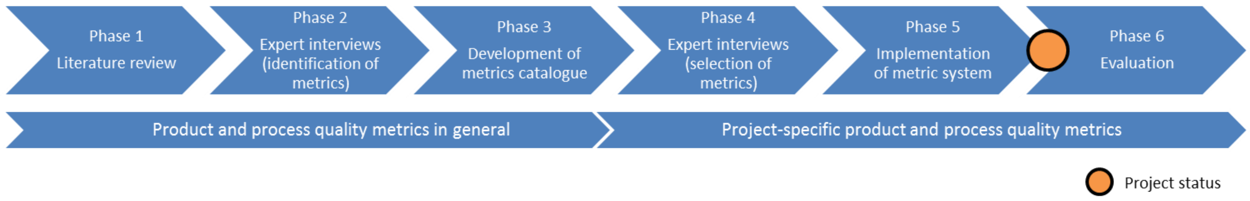
Phase 1
Literature review
Phase 2
Expert interviews (identification of metrics)
Phase 3
Development of metrics catalogue
Phase 4
Expert interviews (selection of metrics)
Phase 5
Implementation of metric system
Phase 6
Evaluation
Product and process quality metrics in general
Project-specific product and process quality metrics
Project status

Figure 3. Study overview.

PRODUCT AND PROCESS QUALITY MODEL FOR AGILE IT PROJECT DEVELOPMENT

Based on an extensive literature review and expert interviews in an IT consulting company, suitable metrics for agile software development projects have been identified. Figure 4 gives an overview of key characteristics for both product and process quality in agile IT projects. For each characteristic, one or more specific metrics exist as presented in Table 3 and Table 4, also showing brief descriptions of all metrics. A more detailed description of the characteristics and related metrics is given below each table.

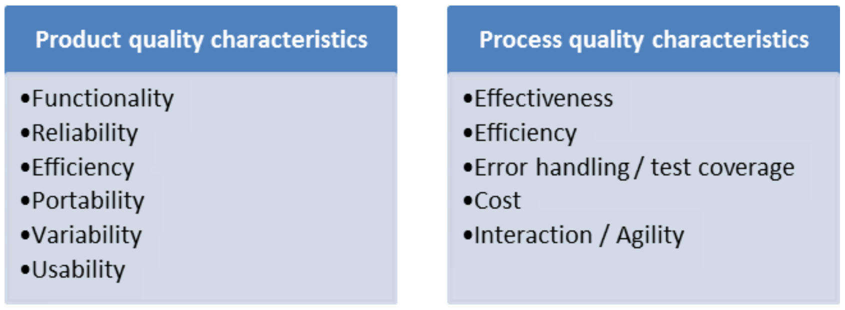

Figure 4. Overview of product and process quality characteristics.

Product Quality Metrics			
Quality characteristic	Metric	Description	Source
Functionality	Accuracy	Number of errors as reported by customer	Bhatti (2005)
	Suitability	Occurrence of unintended system behavior	Bhatti (2005)
Reliability	Test case coverage (code)	Number of test cases relative to lines of code	Nacchiapan (2004)
	Test case coverage (requirements)	Number of test cases relative to requirements	Nacchiapan (2004)
	Test lines ratio	Number of lines of test code relative to number of lines of code	Nacchiapan (2004)
	Mean time to failure	Average time between errors	Kan (2002)
	Downtime ratio	Total downtime relative to total runtime	Expert interviews
Efficiency	Application server capacity	CPU capacity for performing tasks	Expert interviews
	Database capacity	CPU capacity for database queries	Expert interviews
Portability	Interfaces	Number of interfaces	Mooney (1997)
Variability	Classes	Number of classes	Concas et al. (2008)
	Depth of inheritance tree (DIT)	Distance of class to its root (indicator for complexity of inheritance tree)	Ambu et al. (2006)
	Number of children (NOC)	Number of sub-classes	Ambu et al. (2006)
Usability	Error message density	Number of error messages per functional element	Bertoa and Vallecillo (2004)
	Operations density	Number of operations per interface	Bertoa and Vallecillo (2004)
	Time behavior of system	Time for execution of system functions	Seffah et al. (2006)

Table 3. Overview of product quality metrics.

Product Quality Metrics

There are six quality characteristics for which we have identified 16 metrics in total. Bhatti describes different kinds of metrics in order to measure product quality via *functionality* (Bhatti 2005, p. 3). One question he focuses on refers to the product not meeting certain customer demands, identified by counting how often bugs or defects are reported by the customer. This aspect is covered by the metric “accuracy”. The more defects are found and reported by the customer, the less satisfying the usage and the lower the quality is. Another aspect he mentions is “suitability”: according to Bhatti (2005) the metric is defined as the number of requirements that, although covered by the product, do not meet the customer´s expectations.

The second product quality characteristic is *reliability*. In this area especially the metric „mean time to failure“ should be highlighted. It measures the mean time difference between the occurrence of product defects or failures. When the mean time to failure score is low, the product quality is needs to be improved. Kan states that this metric is an important one for systems that are critical in terms of safety or security, for example in aviation (Kan 2002, p. 86). As it is crucial for such systems to work in a highly reliable way, the chosen metrics have to be significant. This kind of measurement could also be effective and helpful for core systems in any business in order to generate customer satisfaction. In addition, Nacchiapan describes a metrics catalogue called “Software testing and reliability early warning” (STREW) (Nacchiapan 2004, p. 1). The covered metrics are used to evaluate and forecast the reliability of a system depending on the number of test cases compared to the size of the system (that is the number of lines of code or the number of requirements).

We could not find suitable metrics to evaluate the *efficiency* of a software product. Nevertheless, two metrics which have already been used in the IT project at hand were mentioned in the expert interviews. Overall, these metrics address the system’s capacity utilization compared to the number of operations performed. In fact the capacity utilization of the application servers and the data base servers are evaluated (expert interviews).

Mooney describes how *portability* can be measured in software development processes. He underlines that the control of interfaces offers the possibility to realize portability, as they are used to communicate with other modules or the environment (Mooney 1997, p. 3). Therefore, we conclude that the number of interfaces can be used as indicator of software portability. However, as increasing numbers of interfaces drive complexity, this metric might have to be more critically investigated in regard to interface efficiency.

For *variability* there are some classic object-oriented metrics suites which focus on the measurement of complexity or maintainability of software (Succi et al. 2005, p. 83ff.). It can be deduced that the more complex the code is, the more difficult it will be to change the software. Rech and Weber (2005) as well as Ambu et al. (2006) describe the metrics “depth of inheritance tree” (DIT) and “number of children” (NOC). DIT evaluates the number of sub-classes of a single class. The more sub-classes a class has and the more complex the inheritance is, the higher the reusability will be. However, at the same time complexity and error-proneness will also increase. According to Rech and Weber (2005), NOC can generally be seen as an indicator of how significant the influence of a single class on the software design is. Another metric is the “number of classes” (Concas et al. 2008, p. 88). The more classes an object-oriented system has, the higher its complexity. As the number of classes normally does not rise in a linear way during the development process, the respective score can help to indicate the complexity of the system at a given point of time.

Both Seffah et al. (2006) and Bertoa and Vallecillo (2004) describe metrics designed to evaluate the *usability* of an application system. First, the “number of error messages per interface density” can help to assess if the user does get enough feedback in regard to occurring errors or wrong usage. Second, an appropriate “number of operations offered by a user interface” indicates high clarity and good usability. Third, “time behaviour” measures if the system conducts operations in reasonable time. This metric has already been used in several ways in our case study project.

Process Quality Metrics			
Quality characteristic	Metric	Description	Source
Efficiency	Velocity	Number of completed features/user stories per iteration	Birk and Heller (2010); Hartmann and Dymond (2006)
	Burndown chart	Amount of work left relative to available time	Birk and Heller (2010)
	Story cycle time	Number of iterations for completion of specific user story	Birk and Heller (2010)
	Builds per iteration	Number of completed builds per iteration	Hartmann and Dymond (2006)
	Obstacles cleared per iteration	Number of obstacles cleared per iteration	Hartmann and Dymond (2006)
	Defect removal effectiveness	Number of removed defects relative to total number of defects	Kan (2002)
	Ratio of re-opened defects	Number of re-opened defects relative to total	Expert interviews

		number of defects	
Effectiveness	Milestone success	Ratio of milestones reached as planned	Expert interviews
	Release date based on velocity	Projected release date based on current velocity	Sulaiman et al. (2006)
	Sprints left based on velocity	Number of sprints to go based on current velocity	Sulaiman et al. (2006)
	Obstacles carried over	Number of obstacles not solved and carried over to next iteration	Hartmann and Dymond (2006)
	Defects carried over	Number of defects not solved and carried over to next interation	Hartmann and Dymond (2006)
Error handling / Test coverage	Tested features	Number of successfully tested features per user story	Birk and Heller (2010); Zenker (2008)
	Test development status	Number of test cases and respective status (open, in progress, completed)	Birk and Heller (2010)
	Test coverage	Number of test cases per user story and respective status	Birk and Heller (2010)
	Defect backlog	Number of defects per iteration	Birk and Heller (2010)
	Defect validation backlog	Number of solved defects	Birk and Heller (2010)
	Defect removal time	Number of iterations for removing a defect	Birk and Heller (2010)
	Defects per million opportunities (DPMO)	DPMO = $\frac{D}{N*O}$*1,000,000	John et al. (2008); Murugappan and Keeni (2000)
	Defects per unit (DPU)	DPU = $\frac{total\ number\ of\ defects}{total\ number\ of\ units}$	John et al. (2008); Murugappan and Keeni (2000)
Cost	Cost Performance Index (CPI)	Earned value relative to actual costs	Expert interviews; Chen (2008)
	Schedule performance index (SPI)	Earned value relative to planned value	Expert interviews; Chen (2008)
Interaction/ Agility	Work item status	Number of task board updates per iteration	Expert interviews
	Sprint meetings	Number of team meetings per iteration	Expert interviews

Table 4. Overview of process quality metrics.

Process Quality Metrics

There are five quality characteristics for which we have identified 24 metrics in total. A key metric for measuring the *efficiency* of the development process is “velocity”, which evaluates how much software a team produces in one iteration/sprint. Measurement is generally done by counting the number of completed user stories per iteration (Hartmann and Dymond 2006 , p. 5). According to Hartmann and Dymond (2006) the metric can be used on a project level to predict how much software a team will deliver in the next or current iteration by comparing it to the former ones. However, it is difficult to compare the velocity across different teams, as the same task could probably be finished by two different teams, but with different efforts. Hartmann and Dymond (2006) also describe additional metrics: the “number of builds per iteration” and the “number of obstacles cleared per iteration”. We should note that obstacles can also reflect issues that are not caused by the team itself but hinder the team’s progress. Birk and Heller (2010) discuss several other metrics like “burndown charts” which can be seen as a traditional agile tool. The chart shows the amount of work which is left until the end of the sprint/iteration. The evolution of this value over time can be used for approximations and predictions. In addition, “story cycle time” can be used to measure the number of iterations required to complete a certain user story. Kan describes the metric „defect removal effectiveness” (Kan 2002, p. 103) which can be used to evaluate the efficiency in removing defects. This is done by comparing the number of solved defects to the number of all defects in the respective iteration, representing the number of solved and missed ones (which are the ones detected in the following iteration).

The second process quality characteristic is *effectiveness*. Using the velocity metric, Sulaiman et al. (2006) describe a method to estimate the release date. It can be calculated by using the work left and the average amount of work that the team can produce per sprint. Hartmann and Dymond (2006, p. 5) state additional metrics which can be used to analyze if obstacles and defects could not be solved and have to be transferred to the following sprint.

The largest section regarding process quality contains metrics for *error handling and test coverage*. The “number of defects” allows a detailed view on the current functional state of the product. Moreover, the more defects exist over a longer period of time, the worse the defect removal process is. Defects that are not solved in appropriate time and that have to be re-opened can be a strong indicator for low process quality. Birk and Heller (2010) describe a number of metrics to measure the mentioned aspects: The “test development status” describes the development of test cases and shows how many of these are “in progress” or “done”, while “test coverage” focuses on the execution of the test cases: Was a test case successful or did it fail (p. 30ff.)? Another issue is the number of defects over time in the “defect backlog” (Birk and Heller, 2010). Defects can be distinguished by type and severity. If there are, e.g., many defects in the backlog at the end of a sprint, it can help to evaluate the quality of defect removal. In addition, the “defect validation backlog” only counts defects that still have to be tested in re-tests. Birk and Heller (2010) found that the defect validation backlog metric improves the communication between development and test teams. When there are too many defects in the backlog this can be a sign of poor coordination between development and test unit. Another aspect is the “defect removal time” which measures the time or the number of iterations needed to remove defects. The catalogue also includes two “Six Sigma” metrics: the number of “defects per million opportunities” and “defects per unit”. Six Sigma includes precise rules for process quality evaluation (John et al., 2008). That is why defects have to be calculated based on the amount of a million opportunities in order to make the results comparable.

CASE STUDY

Selection of Metrics

In the next step, a reduced set of key metrics had to be identified for implementation in our case study project. In order to get meaningful results interviews have been carried out to evaluate each metric based on three main criteria: (1) the usefulness for successfully managing the project; (2) the applicability within the project (e.g., security and confidentiality issues, definition of thresholds, etc.); and (3) the measurability in general, considering access to data, efforts to easily and regularly collect the data. Due to a lack of experience and historic data, the three criteria were weighted equally, each contributing one third to the total score of each metric. Four out of the eight project team members were asked to discuss all metrics and rate them on a scale of 1 (indicating the lowest score) to 5 (indicating the highest score). Based on the group interviews two rankings have been generated, one for product and one for process quality metrics (cf. Table 5 and Table 6). In addition, the interviews revealed that the project team already had a high workload and would not be able to spend much time on providing data on a regular basis. Thus, it became clear that the number of metrics should not be too high and manual efforts for collecting and consolidating the data had to be kept at a minimum. Ideally, at least some metrics could be measured automatically (e.g., by using data from error logs, existing project management tools, …). Having these aspects in mind, the final decision for a set of metrics was made (green cells in Table 5 and Table 6). The project team chose 8 metrics for product and process quality, respectively. However, while the top metrics from the ranking were selected for measuring product quality, some of the top process quality metrics had to be dropped. Since the project was already in its implementation stage, some compromises had to be made, particularly in regard to easy access to relevant data (little to no additional efforts) and high usefulness and applicability. Given that the customer of the project is a government organisation, a few confidentiality issues (e.g., no collection of individual performance data) had to be considered.

No metric has been selected to control for functionality of the product, because it would take the project team too much time to accurately collect and process the related data. Due to the agreed-upon importance of functionality within the team, efforts are undertaken to allow including at least one functionality metric by end of 2012. For the process quality metrics, although ranked overall second, release date based on velocity has not been implemented because the project team did not have any practical experience with this metric and finally decided to use milestone success and defects carried over as metrics for effectiveness. No cost metric was selected due to the very low applicability score for the project. Other issues that have been highlighted during the interviews, but will not be further discussed in this version of the paper, were possible dependencies between metrics, subjectivity of some data, the role of stakeholders, and the need to responsibly work with the collected data.

Characteristic	Metric	Usefulness	Applicability	Measureab.	Total Score	Rank
Functionality	Accuracy	4	4	4	4,0	9
	Suitability	4	5	3	4,0	9
Reliability	🟩 Test case coverage (code)	4	5	4	4,3	7
	Test case coverage (requirements)	3	5	3	3,7	12
	Test lines ratio	1	5	5	3,7	12
	🟩 Mean time to failure	5	5	5	5,0	1
	🟩 Downtime ratio	5	5	4	4,7	2
Efficiency	🟩 Application server capacity	3	5	5	4,3	7
	🟩 Database capacity	4	5	5	4,7	2
Portability	🟩 Interfaces	5	4	5	4,7	2
Variability	🟩 Classes	4	5	5	4,7	2
	Depth of inheritance tree (DIT)	3	3,5	3,5	3,3	16
	Number of children (NOC)	3	4	4	3,7	12
Usability	Error message density	3	4	4	3,7	12
	Operations density	4	5	3	4,0	9
	🟩 Time behavior of system	4	5	5	4,7	2

🟩 implemented in case study project

Table 5. Ranking of product quality metrics.

Characteristic	Metric	Usefulness	Applicability	Measureab.	Total Score	Rank
Efficiency	🟩 Velocity	4,5	3,5	5	4,3	3
	🟩 Burndown chart	5	3	4,5	4,2	7
	Story cycle time	2,5	4,5	4,5	3,8	11
	Builds per iteration	2	5	5	4,0	9
	🟩 Obstacles cleared per iteration	4,5	4,5	4	4,3	3
	🟩 Ratio of re-opened defects	3,5	4,5	3,5	3,8	11
Effectiveness	🟩 Milestone success	3	5	4	4,0	9
	Release date based on velocity	5	4,5	4,5	4,7	2
	🟩 Defects carried over	4	4,5	4,5	4,3	3
Error handling / Test coverage	Tested features	1	5	5	3,7	13
	Test development status	3,5	4	2,5	3,3	14
	🟩 Defect backlog	5	5	5	5,0	1
	Defect removal time	4	4,5	4	4,2	7
	Defects per million opportunities (DPMO)	4	4	2	3,3	14
	Defects per unit (DPU)	4	4	2	3,3	14
Cost	Cost Performance Index (CPI)	4,5	1,5	3	3,0	18
	Schedule performance index (SPI)	4,5	1,5	3	3,0	18
Interaction/ Agility	Work item status	4	4,5	1,5	3,3	14
	🟩 Sprint meetings	4	5	4	4,3	3

🟩 implemented in case study project

Table 6. Ranking of process quality metrics.

Implementation of Metrics

The next step in the case study was to implement the quality model and the previously selected metrics. We followed an approach similar to developing a Balanced Scorecard (but without connecting the metrics to each other) including the following steps for each metric: identification of data source and rules for analysis, thresholds and respective color in reporting cockpit, type of display in reporting cockpit (there exist two, one for the management team showing highly aggregated scores, and one for the project team showing detailed analyses and results). Due to the length restrictions and because this part is not at the core of our interest, we do not provide further details at this point.

DISCUSSION AND CONCLUSION

In this research-in-progress paper, we investigate key metrics that can be used to control both product and process quality in agile software development projects. Given the fact that many IT projects still fail due to poor project management and/or a lack of appropriate controlling, there is a need to more thoroughly identify key metrics that can be effectively and efficiently implemented and used in agile projects with their particularly challenging conditions. We have developed a metrics catalogue consisting of 40 metrics which cover different quality characteristics. This can serve as a good starting point to enrich our understanding of what the best agile project metrics are. Although we have only collected a small amount of data yet it already became clear that in order to effectively manage agile IT projects, product and process quality data have to go hand-in-hand. Using data from the case study and by conducting additional interviews, we expect to learn what metrics work best in the given scenario and what problems need to be addressed. Finally, we hope to make one step towards a more generic model for product and process quality metrics, and showing how project teams can select the most appropriate metrics for a specific project.

Nevertheless, a number of issues need to be critically reflected and leave more work to do. Of course, since we have only worked with one project team in one company, generalizability is a major issue at this stage. However, since most metrics are not per se project-specific rather than quite common particularly in agile projects, our results should be adoptable to other projects. We plan to validate our findings by conducting additional case studies in other companies and project settings to address the issue of generalizability. Furthermore, we are aware that there exist and might be proposed other categorizations of the quality characteristics. From a practitioner´s point of view, it might be easier to have categories as the number of metrics grows, but essentially the metrics are the key elements of quality measurement. Thus, in our opinion, the discussion should not focus on the categories too much. The selection criteria for the metrics (usefulness, applicability, measurability) were primarily given and might be a weak point. Other criteria could be added, and the weights could be derived via appropriate methods instead of just assuming equal weights. Also, since the evaluations have all been done by project team members, the whole selection process was highly subjective in nature and phase 6 in our project will have to show to what extent the reported values on availability of data, data collection efforts, and applicability of distinct metrics were correct.

Overall, we have successfully gone through phases 1 to 5 in our research process, with phase 6 (collection of longitudinal data and evaluation) being on its way. We expect valuable new insights on how to effectively manage product and process quality in agile software development projects. Future research could, e.g., investigate the relationship between product and process quality metrics. There might be a feedback loop in a way that the need to implement many product quality metrics decreases with the number of process quality metrics or with all/most process quality metrics being on ‘green’. More social metrics could be incorporated to also cover employee motivation and similar aspects.

The mentioned issues show that the character of quality models is diverse and depends on a variety of factors. Quality can hardly be measured in absolute numbers alone and has to be defined for every project and product individually. So far, this research has shown that metrics have to be specified in a very detailed way and also have to be adapted to the language and characteristics of the project. Nevertheless, the developed metrics catalogue can serve as a good starting point for the development of project-specific quality models.

REFERENCES

Ambu, W., Concas, G., Marchesi, M. and Pinna, S. (2006) Studying the Evolution of Quality Metrics in an Agile/Distributed Project, *Abrahamsson, P., Marchesi M. and Succi, G. (eds.): Extreme Programming and Agile Processes in Software Engineering, Springer*, Berlin, Heidelberg.

Bertoa, M. and Vallecilla, A. (2004) Usability Metrics for Software Components, *Universidad de Málaga*, Málaga.

Bhatti, S. N. (2005) Why Quality? ISO 9126 Software Quality Metrics (Functionality) Support by UML Suite, *ACM SIGSOFT Software Engineering Notes*, 30, 2, New York.

Birk, A. and Heller, G. (2010) Supporting Team Collaboration: Testing Metrics in Agile and Iterative Development, *Agile Record*, October 2010, 4.

Chen, M. T. (2008) The ABCs of Earned Value Application, *2008 AACE International Transactions*, EVM.03.1-EVM.03.8.

Concas, G., Francesco, D. M., Marchesi, M., Quaresima, R. and Pinna, S. (2008) An Agile Development Process and its Assessment Using Quantitative Object-Oriented Metrics, *Abrahamsson, P. et al. (eds.): Agile Processes in Software Engineering And Extreme Programming*, Springer, Berlin, Heidelberg.

Hartmann, D. and Dymond, R. (2006) Appropriate Agile Measurement, *IEEE – Agile 2006 Conference*, No. 0-7695-2562-8/06, IEEE Computer Society Press, Washington.

John, A., Meran, R., Roenpage, O. and Staudter, C. (2008) Six Sigma and Lean Toolset – Executing Improvement Projects Successfully, Springer, Heidelberg, Berlin.

Kan, S. H. (2002) Metrics and Models in Software Quality Engineering, Second Edition, Addison-Wesley Longman, Amsterdam.

Mooney, J. D. (1997) Bringing Portability to the Software Process, West Virginia University, Morgantown.

Murugappan, M. and Keeni, G. (2000) Quality Software – The Six Sigma Way, *IEEE Proceedings of the First Asia-Pacific Conference on Quality Software*, No. 0-7695-0825-1/00, IEEE Computer Society Press, Washington, DC, USA.

Nelson, R. R. (2007) IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices, *MISQ Executive*, 6, 2, 67-78.

Nacchiapan, N. (2004) Toward a Software Testing and Reliability Early Warning Metric Suite, *IEEE – 26th International Conference on Software Engineering*, No. 0270-5257/04, IEEE Computer Society Press, Washington.

Olague, H. M., Etzkorn, L. H., Gholston, S. and Quattlebaum, S. (2007) Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes, *IEEE Transactions on Software Engineering*, 33, 6, 402-419.

Rech, J. and Weber, S. (2005) Werkzeuge zur Ermittlung von Software-Produktmetriken und Qualitätsdefekten, *IESE-Report*, Nr. 108.05/D, Fraunhofer IESE, Kaiserslautern, Germany.

Seffah, A., Donyaee, M., Padda, H. K. and Kline, R. B. (2006) Usability Measurement and Metrics: A Consolidated Model, *Software Quality Journal*, 14, 2, Springer, Heidelberg, Berlin.

Succi, G., Pedrycz, W., Djokic, S., Zuliani, P. and Russo, B. (2005) An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite, *Mockus, A. (ed): Empirical Software Engineering*, Springer, Berlin, Heidelberg.

Southekal, P. H. and Levin, G. (2011) Formulation and Empirical Validation of a GQM Based Measurement Framework for a Software Project, *2011 International Symposium on Empirical Software Engineering and Measurement*, IEEE Computer Society Washington, DC, USA, 404-413.

Sulaiman, T., Barton, B. and Blackburn, T. (2006) Agile EVM – Earned Value Management in SCRUM-Projects, *IEEE – Agile 2006 Conference*, No. 0-7695-2562-8/06, IEEE Computer Society Press, Washington.

Talby, D. and Dubinsky, Y. (2009) Governance of an Agile Software Project, *31st International Conference on Software Engineering*, Vancouver, Canada, 40-45.