Index-based Process and Software Quality Control in Agile Development Projects

Nicole Rauch and Eberhard Kuhn and Holger Friedrich

andrena objects ag
[Nicole.Rauch,Eberhard.Kuhn,Holger.Friedrich]@andrena.de

Abstract. In software development, it is important to assure a high level of process and software quality. In the agile context, suitable approaches to measure and analyze these aspects are hard to find. CMMI and SPICE are too heavyweighted, while EN ISO 9001 is too lightweight. To fill this gap, andrena objects ag developed ISIS, a navigation system for process and software quality management that combines several metrics. It is based on more than 200 person years of software development experience. ISIS proved its practicability in several customer projects and was certified according to EN ISO 9001 in late 2007.

1 Introduction

For software companies, a high quality level of their products is important, allowing for efficient maintenance and extensibility. Market success relies on an optimal cost-benefit-ratio. Over the past years, andrena objects ag has established a continuous quality improvement process. To achieve a high software quality, the developers' qualification, motivation, and creativity are crucial. For a satisfactory process quality, andrena applies XP [Bec03] and Scrum [Sch04]. In addition, quantitative quality measurements and analyses are highly desirable.

The agile development process followed at andrena is characterized by few but guiding rules. Existing approaches like CMMI [CKS06] and SPICE [Loo07] are far too restrictive and inefficient to be applied to an agile process and thus are aloof the cost-benefit-optimum. EN ISO 9001 [Cas06], on the other hand, defines too few rules to be useful as guidance in software development since the metric "customer satisfaction" is the only index controlling the process. In real projects, the vast amount of available metrics [Kan02] is widely being ignored due to time restrictions. Life sciences, for example, follow a different path. To evaluate the water quality, they also feature innumerable metrics, but the analysis only takes place based on a highly restricted set of *indicators*. This led us to designing a method allowing systematic quantitative control and optimization of the process and software quality based on a small number of metrics. Our goal is to develop software of adequate quality while at the same time being highly productive.

The contribution of this paper is the presentation of the ISIS navigation system as well as the selection of metrics and tools it is based upon. Section 2 introduces ISIS, its metrics and underlying tools, presents the integration into the Scrum process and EN ISO 9001 and gives an experience report.

Metric	In PQI	In SQI
customer satisfaction	17 %	
Number of bugs	15 %	
Deviation from approximated time usage	11 %	
Test coverage	Δ 13 %	23 %
Packages in cycles	Δ 11 %	19 %
Average Component Dependency (Class)	Δ 9 %	16 %
Classes > 20 methods	Δ 6 %	10.5 %
Methods > 15 LOC	Δ 6 %	10.5 %
Cyclomatic Complexity (num. methods > 5)	Δ 10 %	17.5 %
Compiler warnings	Δ 2 %	3.5 %

Fig. 1. Metrics incorporated into process and software quality index (PQI and SQI).

2 ISIS

System Overview. To supplement the agile development methods Scrum and Test-Driven Development, andrena developed ISIS, a navigation system for quality management. ISIS' main component is the project logbook. Characteristics regarding process and software quality are recorded, condensed, and documented in time series. It offers a continuous comparison to predetermined quality goals. Erroneous trends can instantaneously be counteracted. ISIS supports developer teams in keeping their projects on track. ISIS also offers a high degree of transparency to the project management and to the IT management. They have immediate access to objectified indicators for process and software quality.

Included Metrics. ISIS is based on a number of metrics, i.e. indices that describe certain aspects of source code. The metrics included in ISIS should easily be collected and interpreted, and it should be hard to manipulate them. Furthermore, the selected indicators should cover the overall quality as well as possible. To keep the analysis manageable, only a small set of indicator metrics was selected for the evaluation of the software quality. We regard software quality as being holistic, that is, we assume to be able to draw conclusions regarding the whole by only looking at parts. The same holds for the process quality. Although this assumption still lacks scientific validation, we noticed in many code reviews a strong correlation between the quality of architecture, design, and coding of software. Fig. 1 lists the included indicator metrics. The condensation into the two central control indices *process quality index* and *software quality index* applies a heuristic based on andrena's long standing experience in object-oriented software development.

Customer satisfaction is a highly integrated metric. It is decisive for customer oriented services such as software development. The external software quality is indicated by the *number of bugs*. A bug is defined as behavior that deviates from a given specification and that occurs at the customer site. Unfortunately, developers massively repress bugs instead of benefiting from the potential of learning

and avoidance that can be utilized by collecting and analyzing programming errors. The *deviation from the approximated time usage* was added for two reasons. It is important for a customer to have some indication for the cost of a task, and an adequate task approximation is essential in planning. Another indicator for the external software quality as well as for the maintainability and extensibility of the code is *test coverage*. *Packages in cycles* and *average component dependency* both indicate the architecture quality. The former represents the quality at a medium scale while the latter indicates the modularization at the level of subsystems. The *class size* represents an application's design quality as well as its maintainability and extensibility. *Method length* and *Cyclomatic complexity* indicate the readability and maintainability of the code and therefore the coding quality, while *compiler warnings* indicate the workmanship.

The first three indicators are pure process metrics, while the others are software metrics. The process quality index (PQI) is based on the values of the three process metrics in a given time interval as well as on the changes of the software metrics in the same interval (indicated by Δ in Fig. 1). This way, the process quality is determined by current values as well as by recent improvements. The software metrics underlie the software quality index (SQI) at a given moment. For example, 20 % of PQI and 35 % of SQI are based on the architecture quality.

The indicators and the indicated properties can also be regarded as aesthetic criteria, e.g. regarding the proportion of the whole and its parts (modularization, design, class and attributes), symmetry (architecture) or wellformedness (class size, method length).

Tool Zoo. We apply several tools to measure the indices and to evaluate the results as well as for the historiography and visualization. The software metrics are measured by two tools: *Sotograph* [Sot] performs automated static analysis, historiography, delta functions, manual identification, and removal of weaknesses, while *EclEmma* (freeware) is used to measure the average test coverage and to identify local deficits at the level of classes and methods. To capture and historicize programming errors, the *BugCollector* (developed by andrena) is used. The central instrument for the integration, condensation, visualization and historiography is the *project logbook* (developed by andrena). For each datapoint, the Sotograph results are automatically integrated.

Integration into Scrum. Scrum [Sch04] is an agile method to manage work in a socially complex environment. Two meetings are defined which serve the continuous improvement of the process. For both meetings, ISIS provides substantiated input that quantifies qualitative changes. Concrete measurements for improvement are determined and their implementation is being supervised.

ISIS and EN ISO 9001. EN ISO 9001, besides measuring the customer satisfaction, requires at least one management report per year. ISIS exceeds these requirements by far. In a monthly report addressed to management and customer,

each team presents the quality indices, their interpretation and measurements for improvement, if necessary. A *transparent production* is the result.

User Experience. The introduction of ISIS initiates an intensive discussion of software quality, metrics and production processes and establishes quality awareness. Measurements can be taken immediately. Transparent production fosters confidence by management and customer and leads to steadier production. In all andrena projects the process and software quality was significantly increased by ISIS. An indispensable prerequisite is a learning-oriented no blame culture. Due to the limited number of indicators and extensive automation, it only takes us about one hour per month to determine the quality indices.

3 Conclusions

We presented ISIS, a quality management tool developed and used by andrena. This tool is based on a select number of metrics that are condensed into two main indices representing the process and the software quality. This tool is being applied to all projects at andrena. To our experience it captures the quality of a piece of software at a given moment accurately and follows its development process sensitively. Therefore, software companies that are interested in producing and maintaining high-quality software are likely to benefit from applying ISIS.

We have not yet been able to find adequate indicators for some aspects. For example, the *working productivity*, which is often being measured by ΔLOC per time or function points per time, has not yet been integrated. We generally regard ΔLOC to be inappropriate: When refurbishing existing systems, one generally observes a reduction of LOC due to the removal and cleanup of duplicated code, duplicated logic and unspeakable constructions. In the development of new applications, this metric fosters a tendency to using copy and paste. Another important software metric is *duplicated code*. It is not easily being measured and interpreted. Future activities will concentrate on integrating this metric.

References

[Bec03] Kent Beck. *Extreme Programming.* Addison Wesley, 2003.

[Cas06] Michael Cassel. *ISO 9001 Qualitätsmanagement prozessorientiert umsetzen.* Hanser Fachbuchverlag, 2006.

[CKS06] Mary Beth Chrissis, Mike Konrad, and Sandy Shrum. *CMMI: Guidelines for Process Integration and Product Improvement.* Addison Wesley, 2006.

[Kan02] Stephen H. Kan. *Metrics and Models in Software Quality Engineering.* Addison-Wesley, 2002.

[Loo07] Han Van Loon. *Process Assessment and ISO/IEC 15504: A Reference Book.* Springer, 2007.

[Sch04] Ken Schwaber. *Agile Project Management with Scrum.* Microsoft Press, Redmont, Washington, 2004.

[Sot] Sotograph. http://www.software-tomography.de/html/sotograph.html.