Systematic Quality Engineering – Lessons Learned

Markus Großmann, Frank Salger

CSD Research
Capgemini
Carl-Wery-Strasse 42
81739 München
{markus.grossmann;frank.salger}@capgemini.com

Abstract: Three main areas determine customer satisfaction in software development projects: Time, Budget and Quality. In contrast to 'time' and 'budget' where achievement can be determined quite effectively, judging 'quality' often remains difficult. In this work, we outline Quasar Analytics®, the quality assurance framework used at Capgemini Germany. We describe best practices and lessons learned of applying Quasar Analytics®.

1 Situation

Time, Budget and Quality are the main aspects that determine customer satisfaction in software development projects. But whereas being in time and budget can be determined quite effectively, determining quality often remains elusive. A systematic engineering-like approach to quality assurance of critical applications is missing:

- Fundamental conceptual decisions miss the original business objectives. The wrong system is built.

- Core artefacts like requirements specifications and software architectures are not rigorously assessed. 'Downstream' artefacts like code, test cases and documentation are affected by very expensive changes.

- Quality Assurance is just a synonym for 'testing' and not comprehensive enough, leaving quality attributes like 'maintainability' unconsidered. A cost effective maintenance is difficult to achieve.

- Quality assurance is missing continuity and only done when the project 'has the time to do it'. Technical debt slows down development and maintenance.

2 Goal

From these observations, requirements for a truly systematic quality assurance technique can be derived. A comprehensive ‘quality engineering’ approach would: a) Ensure that business objectives are systematically traced down into system development. b) Start early to check maturity of (conceptual) core work products like software requirements specifications and software architectures. c) Take comprehensively all aspects of quality into account, in particular hard ones like ‘maintainability’. d) Ensure continuity of quality assurance over the whole development lifecycle and automate quality assurance to a high extend to achieve cost efficient system development.

3 Solution

From these requirements, we derived a systematic quality assurance approach, called ‘Quasar Analytics®’ [QA]. It is based on three pillars: ‘Quality Gates’, ‘Software Measurement’ and ‘Testing’. At Software Engineering 2010, we gave an overview Quasar Analytics®. In our SE 2011 talk, we will share some details of the method. More importantly however, we will reports our lessons learned and insights from applying the Quality Gates and Software Measurement in various large-scale and business critical projects.

<u>Quality Gates:</u> Quality gates are systematic evaluations applied at specific points in time assessing the maturity and sustainability of produced artefacts as well as the processes followed to produce them. We pursue the following high-level objectives: Make the maturity of development artefacts and processes transparent, derive effective countermeasures for major problems encountered and ‘standardize’ audits to a reasonable extent. The main characteristics of applying quality gates are: They are applied early in the respective project phase. They are applied according to a defined quality gate process and end with a decision. They are no formal process checks but evaluate the content of artefacts. They are conducted by project-external experts.

Lessons learned (Quality Gates): In the talk, we will share lessons learned from applying the Quality Gates. For Example: a) Non-functional requirements pose more problems than functional ones. b) Using non-formal notations (e.g., pictures or examples) help to complement more formal software requirements specifications.

Though quality gates play an important role for the validation of conceptual results of a software project, they must be complemented as soon as the scene changes to implementation results: Here, ‘Testing’ and ‘Measurement’ become the main actors of systematic quality engineering. Testing focuses on assessment of the external quality attributes of a system (functionality, usability, etc.) whereas measurement has its focal point on the internal quality attributes (maintainability, testability, etc.).

Measurement: The idea of measurement in Quasar Analytics is to use a so called "Software Blood Count", a set of quality metrics that are continuously collected and managed by a Software Cockpit. The Software Blood Count is supplemented with a set of common quality indicators, which describe certain metric value patterns that allow the detection of typical quality problems. This approach is the result from our practical experience that on the one hand it is easier to verify the presence of quality deficits but not their absence, and, on the other hand, the impact estimations of quality indicators can't be automated as they need context-specific knowledge and software engineering experience. Hence, human experts perform the quality rating. Differences between Measurement and Quality Gates are: Quality gates just provide a snapshot assessment of the system whereas measurement provides a continuous quality feedback that is needed to recognize the insidious effects of software erosion. Quality gates just assess samples of the system whereas measurement provides automation features that allow the assessment of a whole code base.

Lessons learned (Measurement): The talk will share some lessons learned from the application of Measurement: a) The TOP 3 important questions and most used metrics with respect to internal software quality, showing the most relevant maintainability problems in the past. b) How to use metric thresholds in a right way and keep a relationship of trust in order to prevent measurement dysfunction: the team concentrates on optimizing metric values but the real goals are lost of sight. c) Weaknesses of measurement tools in large-scale project contexts

Quality gates and Measurement are less well suited to verify that the final system complies with the original requirements specification. This is where the third pillar, namely our ISTQB-based test method comes into play. Together, quality gates, measurement and systematic testing ensure comprehensive quality assurance, aligning business objectives and system development and cover the whole software development lifecycle. The overall benefits of Quasar Analytics® from a company point-of-view are a more standardized quality assurance by reusable tools and methods. Further, Quasar Analytics® can be seen as consolidated software engineering knowledge and improve quality assurance skills of employees ("I will do it differently the next time"). Last but not least, we can provide better services to the Customer as quality gets more visible! Our next steps comprise the development of a measurable quality standard for custom software development within the BMBF-funded research project QUAMOCO (Quality Modelling and Integrated Controlling) [QU]. The collection of best practices and processes for software measurement within a guideline and setting up a quantitative controlling process to supervise the quality of the quality gates is another future goal.

References

[QA] Homepage von Quasar Analytics®, Capgemini (accessed 10.01.2011) : http://www.de.capgemini.com/capgemini/forschung/research/analytics/

[QU] Homepage von QUAMOCO (accessed 10.01.2011) : http://www.quamoco.de