Cost-Benefit Analysis of the Quality Measurement System of EB tresos®

Jörg Eibisch, Bernd Hardung, Susanne Hartkopf, Alexander Much

Elektrobit Automotive GmbH

Am Wolfsmantel 46

91058 Erlangen, Germany

Abstract

Quality management staff is often faced with questions regarding the cost-benefit ratio of their activities. Positive cost-benefit results, for example, from software process improvement programs can be found in literature (e.g., [1]). Especially, in small or medium-sized companies, quality management staff needs powerful statements of monetary benefit to convince project staff and management of their activities. It is often the case that a cost-benefit analysis performed in the starting phase is critically questioned because monetary benefits are mostly based on assumptions.

In this paper we report about the success of an automated quality measurement system. It is applied in the context of the quality management process of the AUTOSAR implementation EB tresos® at Elektrobit Automotive GmbH. The success mainly depends upon two facts: First, the benefits that different software product development roles experience (e.g., for the project manager: earlier identification of critical product parts in order to plan respective measures). Second, cost savings due to the quality measurement system after the cost of the system has been deducted. Conservatively calculated, the quality measurement system pays off in less than two years.

Keywords:

Cost-benefit analysis, quality measurement system, SPICE

1 Introduction

It is a fact that software development has to cope with short development cycles, small budgets, volatile requirements, unclear quality demands, distributed developed locations, and many other challenges. One important factor to deal with these challenges is to establish adequate processes. However, processes can only be adequately established, if the objectives for the software development and its processes are set, communicated, and controlled. The EB tresos® product development staff realized that. Supported by the market demand for Automotive SPICE© [2], [3] conformant processes, but also confronted with tight cost constraints, product development staff has built up an automated Quality Measurement System (QMeaS). The QMeaS

- demands the setting of measurable goals,
- automates software builds and tests,
- collects data regarding specifications, bug reports, or test results,
- processes the data, and
- reports the attainment or deviations of goals to every person involved in the product development.

We see the QMeaS as a successful quality management initiative to improve quality and efficiency in the software product development of EB tresos®. In order to show the value of the QMeaS, we performed a cost-benefit analysis with impressing qualitative and quantitative results. To make quantitative statements, the actual costs of the QMeaS are compared with the savings that were made because of the system. We found out that the system in a conservative calculation pays off in less than two years.

Before presenting the results of the cost-benefit analysis in Chapter 4, the company Elektrobit Automotive GmbH, the software product EB tresos®, and the QMeaS are described in Chapter 2. Additional, the procedure of the cost-benefit analysis is outlined in Chapter 3. The paper concludes with a summary and an outlook in Chapter 5.

2 EB, EB tresos®, and the Quality Measurement System

2.1 Who is Elektrobit Automotive?

Elektrobit Corporation develops advanced technology and transforms it into enriching end user experiences. The company specializes in demanding embedded software and hardware solutions for the automotive industry and wireless technologies. The EB Automotive Business segment offers an extensive range of standard software products and professional tools that support the whole process of the in-car software development. Besides developing pioneering products, the company has also specialized in services and consultancy for the automotive industry to which it is a supplier of a wide range of software ready for series production for AUTOSAR, infotainment, navigation, HMI, driver assistance, and FlexRay.

Elektrobit Automotive GmbH was established in 1988 as 3SOFT GmbH and since 2004 has been part of the global Elektrobit Corporation. Since 2006 3SOFT GmbH has been operating under the name of Elektrobit Automotive GmbH (from now on referred to as EB). EB has branches in Munich Böblingen, Braunschweig,

Gaimersheim (all located in Germany), Vienna (Austria), Paris (France), Novi/Michigan (USA), Tokyo (Japan), Bothell/Washington (USA), Beijing (China). At the end of 2008, 720 employees worked at EB [4].

2.2 What is EB tresos®?

EB tresos® (from now on referred to as EB tresos) is a family of collaborating electronic control unit (ECU) software tools centered around a basic software core. The name EB tresos stands for the compliance with software standards in the automotive industry, such as AUTOSAR, FlexRay and CAN. The EB tresos product family uses open interfaces and supports standardized file formats. This is also true for the common data basis, which allows company-specific tools or even third-party software to be integrated in the development environment.

Developing basic software and integrating it from multiple suppliers is a work-intensive task. The AUTOSAR [5] consortium has standardized basic software for ECUs. However, approximately 140 AUTOSAR specification documents and many different configurations need to be managed for basic software development.

Within the EB tresos product family, EB tresos AutoCore is the implementation of AUTOSAR-compliant basic software. The production-ready software delivers a complete infrastructure for running complex control strategies in a multi-bus network environment, including FlexRay, CAN and LIN. AUTOSAR OS and RTE, together with modules for memory management, diagnostic services and ECU mode management round off the integrated software package. Strategic partnerships guarantee the availability of EB tresos AutoCore for the most popular automotive microcontrollers, providing the most comprehensive and mature collection of AUTOSAR software on the market today. Several modules of EB tresos AutoCore are already employed in series production [6].

2.3 The QMeaS

EB tresos aimed at developing a quality measurement system that fulfills two major objectives. First, the quality measurement system must be able to cope with the complexity of the software development [7]. Currently, everyday software is built for twenty platforms and forty modules. If we assume only one quality metric for each module, we would already have collected 800 data points per day. Second, the system should act as common communication base for everyone involved in the software product development. The QMeaS as described in this paper was first developed for the EB tresos AutoCore modules. Currently, the usage of the QMeaS is extended to other members of the EB tresos product family.

In Section 2.3.1, we explain the quality model that is implemented with the QMeaS. Section 2.3.2 provides an overview to the QMeaS and Section 2.3.3 shows the flow of quality around the QMeaS.

2.3.1 Quality model

We did not want to measure and display thousands of data points for a set of metrics commonly known in the software development. On the contrary, we wanted to define quality goals for which appropriate metrics were selected. Therefore, to attain a goal-oriented quality model the concept of the Goal Question Metric Paradigm (GQM) [8] was applied. Furthermore, the Software engineering - Product quality ISO Standard [9] was used to accomplish a common terminology. Figure 1 shows an excerpt of the

quality model for the quality goals Usability and Reliability. The quality goal Reliability is broken down among others to the quality sub-goal of Maturity. Maturity is determined by one or more quality criteria. A quality criterion is a formula consisting of one or more metrics. For quality goals, sub-goals, and quality criteria, thresholds are predefined to decide in the evaluation upon the attainment of the quality levels. The actual data is collected according to the metrics.

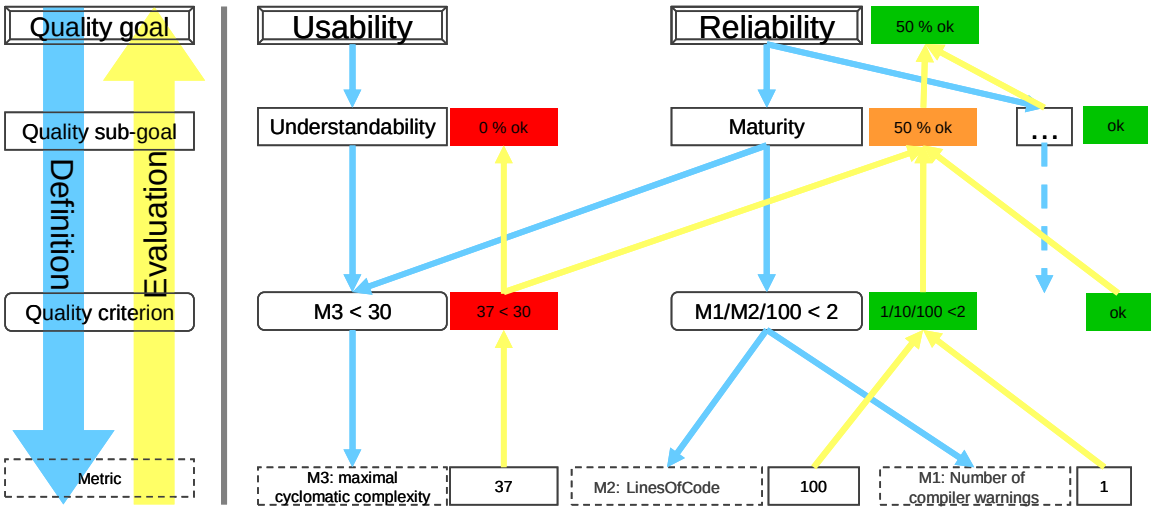

Figure 1: Excerpt of the quality model for EB tresos

The quality model has been implemented step-by-step in parallel to the ongoing software product development. Currently the standard quality model consists of 5 quality goals, 13 quality criteria and 16 metrics [10]. These metrics are collected in every build and the degree of compliance with the five quality goals is reported automatically.

2.3.2 Overview to the QMeaS

Figure 2 shows an overview of the QMeaS. One part is the development database. Among others code, specifications, bug reports, and test cases are stored in the development database. Furthermore, we consider test systems as part of the QMeaS. The idea behind the test systems is that software developers develop on standard PCs with MS Windows and the QMeaS performs the platform-specific tests. Each test system is individually configured to the needs of a dedicated platform. A test system encompasses a standard PC and the platform development environment (including hardware boards, compiler, and debugger). Each test system individually pulls the data needed from the development database and uses the data for the continuous integration and testing of all supported platforms.

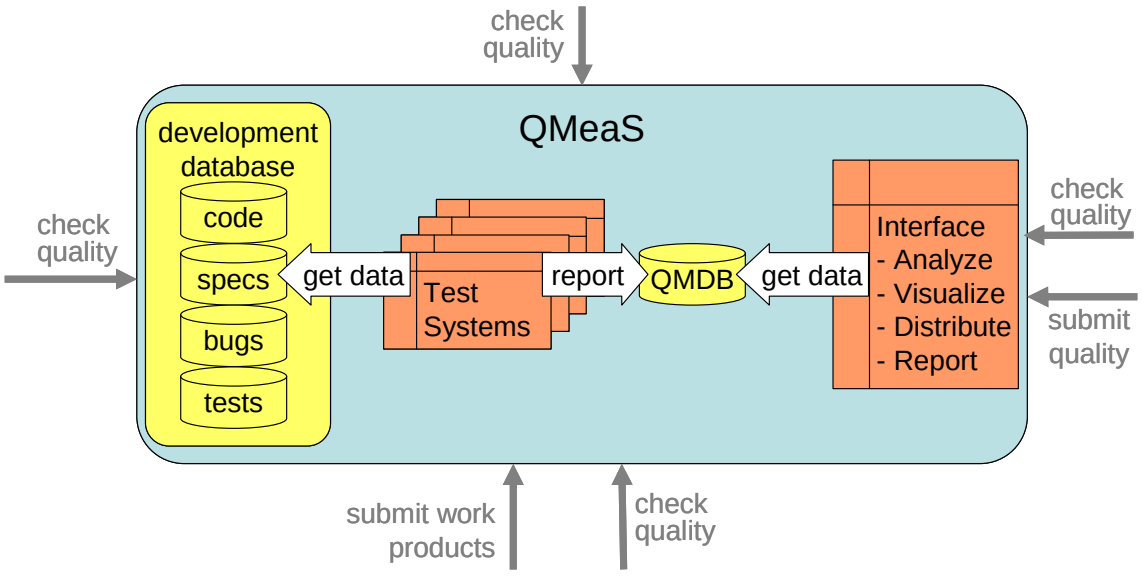

Figure 2: Overview to the QMeaS

One run of a test system is called build. Thus, with currently twenty platforms at least twenty complete new builds are generated every day. Additionally, after every committed change of the developers, a module build is started on all test systems ensuring instant feedback.

For each of these builds and additionally for the product builds and instant module builds quality data are gathered. This includes, for example, test results itself, static code analysis, dynamic code analysis as well as statistics of open problems. Then, this data is filtered and stored in the QMDB, which is the basis for automated reporting, analysis, project reporting and acceptance of a configuration of the product.

The web-browser interface provides views for each platform and module. Every software development role is able to focus on their needed views. The state of quality goals or quality criteria is visualized with colors. Not fulfilled quality criteria can be listed in a separate view as well as changes and trends among builds can be displayed.

The the test systems and the servers for the databases are kept in a separate room. The room is equipped with an air conditioning system, the network infrastructure, and appropriate furniture and is called laboratory.

The software that runs the test systems, the QMBD, and the interface is developed by EB staff.

2.3.3 Flow of quality around the QMeaS

Figure 3 depicts the flow of quality around the QMeaS. The product management (ProdMmgt) receives the quality requirements from the customer and after delivery the customer accepts the quality. The product management orders the quality from the project manager who in turn assigns the required quality to the software developers. The developers produce test cases and software code and submit them to the QMeaS. The QMeaS as well as the integrators provide feedback to the developers. In the course of the development cycle, everyone checks the status of quality in the QMeaS. Before the delivery, the quality assurance controller (QAC) finalizes the quality report and sends the quality report to the product management.

The product management informs the integrators about the attainment of the quality, so the integrators can submit the quality levels to the QMeaS.

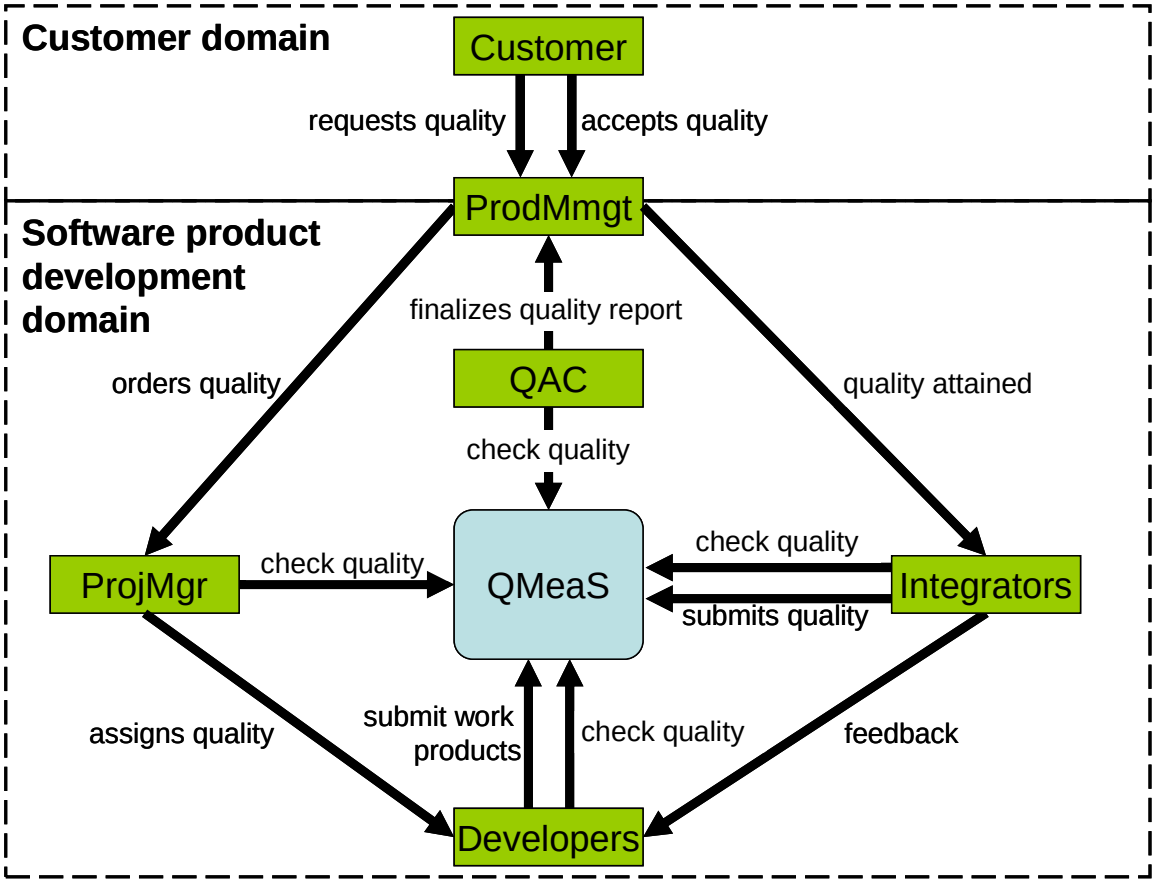

Figure 3: Flow of quality around the QMeaS

3 Procedure to the cost-benefit analysis of the QMeaS

We considered the cost-benefit analysis as measurement activity with a specific scope. It was our goal to attain valuable and reliable results from the cost-benefit analysis. Therefore, we aimed at a thorough planning and goal-oriented execution of the cost-benefit analysis. When we planned our procedure for the cost-benefit analysis, we followed the principles of the GQM paradigm [8] (see Figure 4), and an instantiation for software improvement programs [11].

As first step, we formulated our measurement goal that reads “Evaluate cost and benefit of the QMeaS as applied to EB tresos”. In order to clarify what cost and benefit means, we formulated questions. To quantify the cost-benefit value, the question reads “How long does it take until the investment costs of the QMeaS amortizes?”. Furthermore, we were interested in qualitative statements about the benefit that people involved in the product development experienced. Therefore, we asked the director, product management, project management, software developers, and quality assurance controller about positive changes because of the QMeaS. This question was asked regarding customer related activities, internal product development, and their personal work.

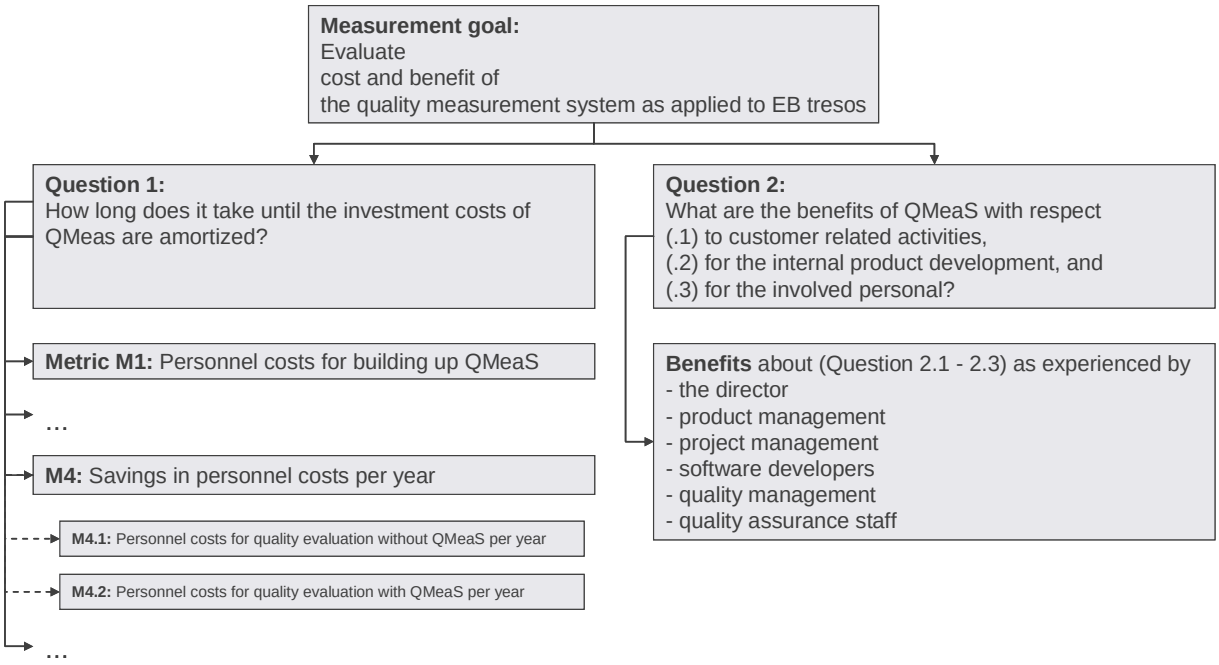

Figure 4: Measurement model for the cost-benefit analysis of the QMeaS

In the second step, we identified the sources to collect the needed information. For most of the cost data, the time recording system of EB was used. For the product development of EB tresos a detailed cost unit structure is set up with separate cost units for the QMeaS. To collect the benefits from the different people, we decided to conduct interviews. An interview guideline was created that covered all the needed information. As third step, we performed the interviews with the different people by means of the interview guideline. Furthermore, we collected the cost data from the time-recording system. Finally, we validated the data and information collected, evaluated the point in time of amortization and the interview results.

4 Results of the cost-benefit analysis of the QMeaS

In Section 4.1, the qualitative results from the interviews are described. In Section 4.2, the quantitative cost-benefit analysis is presented.

4.1 Qualitative results regarding the benefit of the QMeaS

We looked at the benefits of the QMeaS on three different product scopes and from the point of view of different product development roles.

- The first scope looked at the benefits when communicating with customers (see Section 4.1.1).
- The second scope refers to the benefits for the product development in general (see Section 4.1.2)
- Third scope encompassed the personal benefits of the interviewed person (see Section 4.1.3).

Interviews were conducted with representatives of the roles director, product manager, project manager, software developer, and quality assurance controller. We do not want to individually refer every statement to a particular role. However, it is in the nature of the different roles that product manager and director were able to say more about the first two scopes, whereas the software developer contributed more to

the third scope. The project manager and the quality assurance controller were able to equally contribute to all three scopes.

4.1.1 Benefit of the QMeaS when communicating with customers

The QMeaS can be seen as a business enabler for EB tresos. Although it is hard to quantify to which extend the existence of the system influenced the customer decision, the absence of the system would have meant loosing business.

In the course of acquisition processes, potential customers ask how EB tresos is tested. The question arises because of the known complexity of the AUTOSAR standard and customers want to know how we deal with the complexity. Potential customers are OEMs, Tier-1s, or semi-conductor manufacturers. The ideal point in time to demonstrate the QMeaS is as soon as questions arise that concern the software development. The demonstration may take more than half a day. We are able to show the results from the test runs of the latest code changes in a web browser. This means the data is always current and accessible on every computer with intranet access. The transparency of EB tresos' software development and software quality impressed and often convinced the potential partners. EB is able to trustworthily demonstrate its competence by means of this system. In these demonstrations, potential customers said statements such as: "We also want such a system". Or: "This system must be a benchmark for others".

The communication and decisions with the customers about the development states of a release became more transparent. EB now has reliable data for all three parameters in the triangle of cost, schedule, and quality. Customer surveys reflect the increasing satisfaction in connection with the maturing of the QMeaS.

4.1.2 Benefit for the product development

The order of the subjects represents the numbers of mentions by the interviewed people. Interestingly, the first subject was mentioned by everyone immediately after having asked the question about the benefit for the product development.

Transparency of the software development
The transparency of the QMeaS for everyone involved in the product development bases

- on the accessibility via intranet with a web browser.
- the availability of current data. Software developers get feedback within one hour about the correctness of code changes and management is able to inform themselves every day with current data.
- the uniform understanding of the quality required.

Quality can be ordered
Quality levels can be defined by product management. Product management knows the customers' demands and is able to set the quality levels platform-dependently, and also according to different stages of the releases. The predefined quality levels or thresholds can be controlled and monitored by viewing the reports that are automatically created by the QMeaS.

Easy identification of critical module parts
The identification of critical modules or module parts is free from bias, because the identification is based on objective data. However, the gut feeling of software developers about a critical module should not be underestimated.

Increases motivation
Developers are motivated by the system to improve the quality of their module. If the quality of their module is red while the modules' quality of the other developer is green, then developers are encouraged to improve their module

Quality of code attains delivery state earlier than with a semi- or not-automated processing
In a semi- or not-automated quality check processing, quality of code is mostly considered very late in the development. With the QMeaS, quality of code is taken care of from the start of the development. Code changes are qualified according to the predefined quality criteria within one hour. Side effects of a code change in other parts of the code are noticed with the next test run.

Deterministic software development
Due to the QMeaS, the software development has become deterministic: Process steps are performed in a uniform way. The uniformity has enabled the software developers to comply with processes and work instructions.

Scalability of the software development team
The reporting functionality of the QMeaS helps to settle new employees into their new job. New employees learn through the instant (i.e., next test run) feedback about the quality of their implemented code.

Costs of target testing can be reduced
Hardware boards for testing the software is not available in a sufficient number. This bottleneck is bypassed via the QMeaS (see also Section 4.2.2).

Analysis of trends is possible
The quality measurement results of every build are stored permanently in the QMDB and thus can be compared. Analysis of trends is possible among different releases, but also among different platforms.

4.1.3 Personal benefits

The personal benefits are strongly linked with the benefits described in the Section 4.1.2. Therefore, in the following only enhancing statements from the different interviewed people are presented.

Benefit for the software developers
Interestingly, one software developer talked about a psychological aspect of the QMeaS. The software development process as applied gives him the feeling that he really finished his work at the end of the day. In the morning, after having developed the tests, the test reports were red. In the evening, after the code changes, the test reports are green. Furthermore, the software developer very much appreciates the uniform understanding of quality. In that way, the personal demand for quality is tightly linked with the management's demands for quality.

According to the software developers, they have improved their efficiency by 20 to 60 per cent. This increase can be justified by the following observations:

- The settling-in period for a new developer decreases.
- The software developers receive feedback within one hour. The reports support the software developer in the identification of faulty code parts.
- They are not disturbed by separate test activities. The tests run at the touch of a button.

- They are not annoyed by assembling the setting for a new platform development environment.
- The manual work of tracing the requirements to the code and back is automated.

Benefits for the project manager
The QMeaS gives the project manager a safer feeling, because the project manager is able to compare the statements of his colleagues with the results from the QMeaS and may identify critical modules in an early phase of a release. This allows the project manager to better plan resources, to shift resources early, or to think of a redesign of a constantly “red” module. In that way the project manager is able to act proactively. The project manager benefits from the project members’ benefits by the system.

Benefit for the product management
At the point in time of a release, the state of the software can be determined by detailed test results. This increases the confidence in the software development. Furthermore, it is a good feeling to know that spot checks are always possible because the data is current.

4.2 Quantitative cost-benefit analysis

First, the costs of building up and maintaining the QMeaS are calculated (Section 4.2.1). Second, the cost savings that are made due to the system are explained (Section 4.2.2). Third, the costs are compared with the cost savings (Section 4.2.3).

4.2.1 Costs of the QMeaS

The total costs of building up the QMeaS are considered 100 per cent (see Table 1). All other monetary amounts are presented relatively to the costs of building up the QMeaS. Therefore, the unit is per cent. The total costs consist of 92 per cent for personnel cost and 8 per cent for the laboratory and hardware (see Section 2.3.2). The personnel costs mainly arise from the development of the software that runs the test systems, the design and implementation of the QMBD, and the interface (see Figure 2).

For the yearly maintenance, 19 per cent per year must be considered. 16 per cent fall to personnel costs to maintain the QMeaS and to familiarize the users with the QMeaS. Cost for the laboratory and hardware write-off amount to 3 per cent each year.

	Building up	**Maintenance per year**
Personnel costs	92 %	16 %
Laboratory and hardware	8 %	3 %
Sum	100 %	19 %

Table 1: Costs of the QMeaS

4.2.2 Savings due to the QMeaS

The savings are attained by reducing the personnel costs per release and by reducing the costs for the platform development environments per year.

Saving of personnel costs per year
Savings in personnel costs amount up to 66 per cent per year (see Table 2) assuming 60 releases per year. This number is derived through the following calculation: In comparison to the situation without the QMeaS, the automated quality assurance tasks of the QAC and the integrators save an average amount of 1.1 percent per release. These savings stem from the reduced effort for data collection and aggregation, processing of the quality reports, performance of quality checks, performance of tests, and building the product. If we assume 60 releases per year, then 66 per cent are saved with each year. If we assume 80 releases, more can be saved.

Savings of personnel costs	**Savings**
Per release	1.1 %
60 releases per year	66.0 %
80 releases per year	89.0 %

Table 2: Savings of personnel costs per year

Cost savings for the platform development environment
Cost savings for the platform development environment (PDE) amount up to 44 per cent per year (see Table 3). This number results from the savings of two PDEs. Without the QMeaS, four PDEs were needed. With the QMeaS only two PDEs are needed. On average in a conservative assumption[1], one PDE costs 1.1 per cent. Currently, 20 platforms are supported by EB tresos, so that 44 per cent are saved.

	Without the QMeaS	**With the QMeaS**
Number of PDEs per platform	4	2
Costs for PDEs per platform	4,4 %	2,2 %
Costs for 20 platforms	88.0 %	44.0 %
Savings per year		44.0 %

Table 3: Savings of costs for the platform development environments per year

4.2.3 Comparison of costs and savings

The costs of the QMeaS amount to 119 per cent (building up costs of 100 per cent plus 19 per cent maintenance costs per year) after the first year. The savings amount to 110 per cent in the first year. This means that in a conservative calculation the break even point is attained after 13 months (see Figure 5).

[1] Penalties in case EB broke a board due to high physical exposure are not taken into account. However, this incident happens regularly and sums up, too.

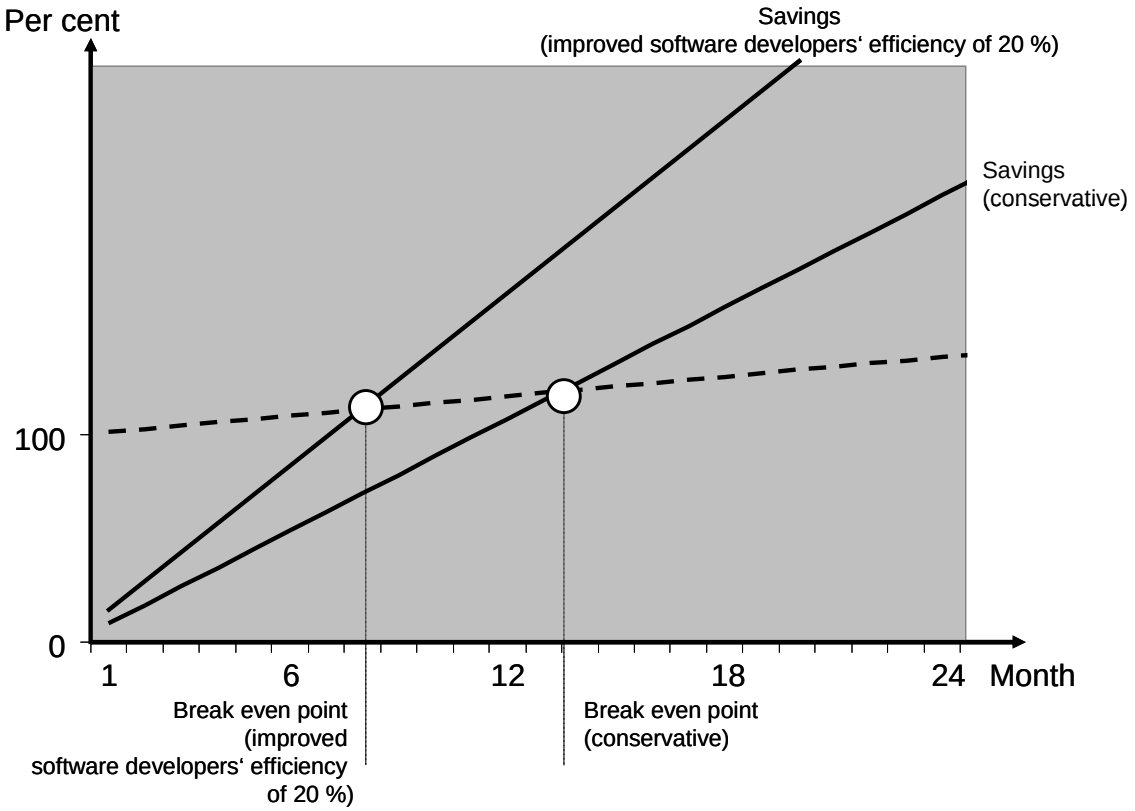

Figure 5: Break even points for the QMeaS

The conservative calculation is based on the current number of sixty releases per year. However, the calculation does not consider the increase of efficiency of the software developers as found out in the survey (see Section 4.1.3) as well as the expected increasing number of releases in the next years. Therefore in reality, the break even point for the amortization is earlier. Figure 5 depicts the savings with and without the software developers' efficiency increase of 20 per cent per year. The later break even point assumes the conservative calculation, whereas the earlier break even point includes the improvement in the software developers' efficiency which results from the QMeaS.

After applying the QMeaS for two years, a total amount of approximately 500,000 Euro can be saved (difference between building-up and maintenance costs and cost savings) not considering the increase of efficiency of software developers.

5 Summary and Outlook

We report about the success of a quality measurement system as built up and applied in the software product development of EB tresos at EB Automotive GmbH. The system automates the software builds and tests, enables the traceability from requirements to code, checks predefined quality levels, and informs everyone involved in the product development of the current status of the development. The system was built up in parallel to the ongoing product development, which was a challenge for the entire team; by now software developers consider the break down of the system as torture. The results of the cost-benefit analysis are presented in four topic areas. First, the benefits of the system in the acquisition and communication with customers are explained. Second, the benefits for the product development are listed. Third, the personal benefits are summarized that different people involved in

the product development experience. Last, we make a quantitative statement about the cost-benefit ratio of the system. We found out that the money spent for building up and maintaining the system pays off in less than two years only by looking at the savings of personnel costs and costs for the platform development environment. The costs will pay off significantly earlier when taking into account the improvement of the software developers' efficiency.

It is now the objective of the management to broaden the application of the quality management system. It shall be applicable also for customization projects in which EB tresos modules are customized to the individual needs of a customer. An additional aspect for quality management staff is that successfully applied quality approaches may open up new business fields.

Acknowledgements

The authors thank Stefan Duda, Felix Fastnacht, Tobias Jordan, and Roman Pallierer for their valuable input in the cost-benefit interviews.

References

[1] James Herbsleb, Anita Carleton, James Rozum, Jane Sigel, and David Zubrow. Benefits of CMM-Based software process improvement: Executive summary of initial results. Software Engineering Institute (SEI), CMU/SEI-94-TR-13 http://www.sei.cmu.edu/pub/documents/94.reports/pdf/sr13.94.pdf, page 14, 1994.

[2] Automotive SPICE©, http://www.automotivespice.com, April 2009.

[3] Markus Müller, Klaus Hörmann, Lars Dittmann, and Jörg Zimmer. Automotive SPICE in der Praxis. dpunkt.verlag, Heidelberg, 2007.

[4] EB webpage, http://www.elektrobit.com, April 2009.

[5] AUTOSAR. http://www.autosar.org/, April 2009.

[6] EB tresos® AutoCore: http://www.elektrobit.com/file.php?1515, April 2009.

[7] Felix Fastnacht. Immer auf dem neuesten Stand. Elektronik Automotive Sonderheft AUTOSAR, Seite 42-44. October 2007.

[8] V. Basili, G. Caldiera, H. D. Rombach: The Goal Question Metrics Approach. In: Encyclopedia of Software Engineering. John Wiley & Sons, S. 528–532, 1994.

[9] ISO/IEC 9126-1:2001, Software engineering - Product quality – Part 1: Quality Model, 2001.

[10] Susanne Hartkopf and Bernd Hardung. Qualität von Anfang an. automobil elektronik, http://imperia.mi-verlag.de/imperia/md/content/ai/ae/fachartikel/ael/2009/02/ael09_02_030.pdf, April 2009.

[11] Andreas Birk, Dirk Hamann, and Susanne Hartkopf. A framework for the continuous monitoring and evaluation of improvement programmes. Proceedings of the Second International Conference on Product-Focused Software Process Improvement (PROFES 2000), Springer, Berlin, 2000.