Published in Software Quality Professional, volume one, issue three, june 1999

PRACTICAL QUALITY ASSURANCE FOR EMBEDDED SOFTWARE

Drs. E.P.W.M. van Veenendaal CISA

Inspections are generally accepted as a means to improve the quality of software products in an effective and efficient way. However, inspections are not a standard practice in a great number of software projects and software organisations. Introducing and implementing inspections is often a tedious and difficult task, because software engineers must be personally convinced of the effectiveness of new methods before they will consistently use them.

This paper describes the applications of inspections and structured testing as measures for quality assurance, in a television set software project at Philips Semiconductors. Inspections were used to reduce a number of project risks and to assure the quality of the products. The way inspections and structured testing were carried out is described and of course the achieved results, including data. The aim is not to show a "theoretical correct" and optimal application of inspections, but a successful application in practice in a critical and difficult project. The implementation and introduction process of these "new" methods within the project runs as a central theme throughout this paper, closing off with a number of experienced critical success factors.

SOFTWARE INSPECTIONS

In every software development phase defects are introduced, found and rework is being carried out. However, often most defects are only found when the software product is almost finished, e.g. during the system and acceptance testing phase, or even during operation. Defects found during the testing phase have the disadvantage that their rework on the almost finished software product is very time consuming. It would have saved the development organisation a lot of time if these defects where found during an earlier development phase. Inspections are an effective and efficient measure that can be introduced to improve the quality of the products at an early stage. Besides finding a defect at the earliest possible moment, the prevention of defects is the important issue. Inspections can also be used as a means for defect prevention. Based on an analysis of the defects that were found, the software development processes can be adapted and optimised to prevent these defects from occurring in the future (as far as possible). Engineers that are involved in the inspection process can learn from their defects or the defects that were made by someone else.

Inspections can be defined as a structured review of an engineers' software work product carried out by his colleagues to find defects and to enable the engineer to improve the quality of the product. The reason for carrying out inspections can be explained by using Wienberg's concept of egoless engineering [11]. Weinberg refers to cognitive dissonance as the human tendency to self-justify actions. Since we tend not to see evidence that conflicts with our strong beliefs, our ability to find errors in our own work is impaired. Because of this tendency many engineering organisations have established independent test groups that specialise in finding defects. Similar principles have led to the introduction of software inspections.

CONSUMER ELECTRONICS

Within the world of consumer electronics the amount of embedded software is growing rapidly. The amount of software in high-end television sets has increased by a factor of about eight over the last six years [10]. The increasing complexity of software in consumer electronic products calls for a high level of quality assurance activities. The challenge is even greater if one thinks of the fact that in principle no

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

field defects can be allowed. For it is almost impossible to recall consumer products from the market since the location of these products is generally unknown. Implementing software product quality is not an option anymore, it has become mandatory.

In contradiction to the need for a high quality product, which requires time and effort to develop, are the demands that come from a highly competitive market. The time-to-market for consumer electronic products is decreasing rapidly and is putting extra emphasis on software development to deliver on time, e.g. to reduce the throughput time. Being one month late on market introduction will result in a substantial reduction of the calculated profit. In addition, the market of consumer electronics has been faced with a 5 to 10 percent price erosion per year [10]. The price of a product is, amongst a number of other things, determined by the microcontroller used. Therefore the use of ROM and RAM remains under high pressure in consumer electronic products, leading to severe restrictions on code size. Time-to-market and price are thus putting extra tension upon the objective to develop a high quality software product.

This paper describes a specific project within consumer electronics that had the objective to develop the leading versions for a low-end TV software product. The paper focuses on the way the software quality activities, e.g. testing and inspections, were implemented and performed. The TV software development project was carried out by a software group within Philips Semiconductors. Philips Semiconductors is one of the many companies in the electronics industry that is in the transition from the hardware stage to the embedded software stage. Companies that are in this position are starting to add software to their hardware products as added value and to achieve a higher level of flexibility within their products. However, development is still dominated by hardware [3].

The low-end TV software project was carried out at various locations throughout the world. Eindhoven (The Netherlands) and an Asian location (Singapore) were used as the main development sites, two other sites were used for reasons of specific knowledge and skills on respectively teletext and closed captioning. From the start the project was faced with tight deadlines and difficulties in getting adequate staffing. The final staffing was a mixture of experienced engineers, both in software development in general as in the application domain, experienced engineers with no practical knowledge of the application domain and less experienced engineers who just finished their education program. During the development a quality system, certified according to ISO 9001, was used and within the software organisation the Capability Maturity Model [4] is used for continuous software process improvement.

Trying to meet its objectives, i.e. a high quality product (“zero” field defects) on time according to the agreed functional requirements, the project was faced with a number of risk factors:

- *partly less experienced staffing*; as stated before only a part of the staffing was experienced. Little time was available for explicit training, so most of the training had to be carried out as training-on-the-job. Additional attention was considered to be needed for ensuring the quality of the various software work products.
- *more than one location*; the project was carried out at various locations, especially Eindhoven and Singapore. Again this provided a number of challenges considering methods and standards used, interfaces between components, planning and tracking and quality in general. These challenges are enlarged by the cultural differences that exist between Europe and Asia.
- *project size*; although the software organisation had experience in developing TV software, the size of the project introduced a new dimension. Project planning, tracking and oversight principles had to be expanded for this project towards a “new” environment. A number of project management activities were to be carried out for the first time at this level.

As an answer to these risk factors it was decided by project management and the project sponsor that quality needed explicit attention. A separate Software Quality Plan had to be written, in which the

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

quality strategy would be one of the most important issues, and quality engineering was defined as a dedicated role within the project.

THE QUALITY STRATEGY USED

One of the most important elements in the software development process is the definition of the quality strategy. The quality strategy defines what quality activities, e.g. testing and inspections, will be carried out on which software work products and how thorough the various quality activities will be. Choices have to be made because it is impossible to evaluate a software product completely, e.g. 100% coverage on all components and all quality characteristics is perhaps possible in theory but no organisation has the time nor the money to achieve this. The quality strategy of the TV software project was determined via a process of communication within the software development team, trying to identify the most important components and quality characteristics of the software product and making use of existing (testing) experiences at previous TV software projects. The aim is to have the most feasible coverage on the most important and complex components and quality characteristics of the software product.

The V-model (see figure 1) was adapted as a starting point for evaluating the software product. This means that in parallel to the phasing model for software development, staring with the Customer Requirements Specification (CRS), a sort of phasing model is used for the software quality activities, e.g. testing and inspections. The arrows in the V-model show which documentation was used as a reference basis for a certain type of testing. To improve the quality of the starting documentation, thus providing a solid basis for testing and trying to detect defects as early as possible in the development process, the V-model was enriched with various inspection activities. The way the various types of testing were carried out is described in the next paragraph before focusing on inspections throughout the rest of this paper.

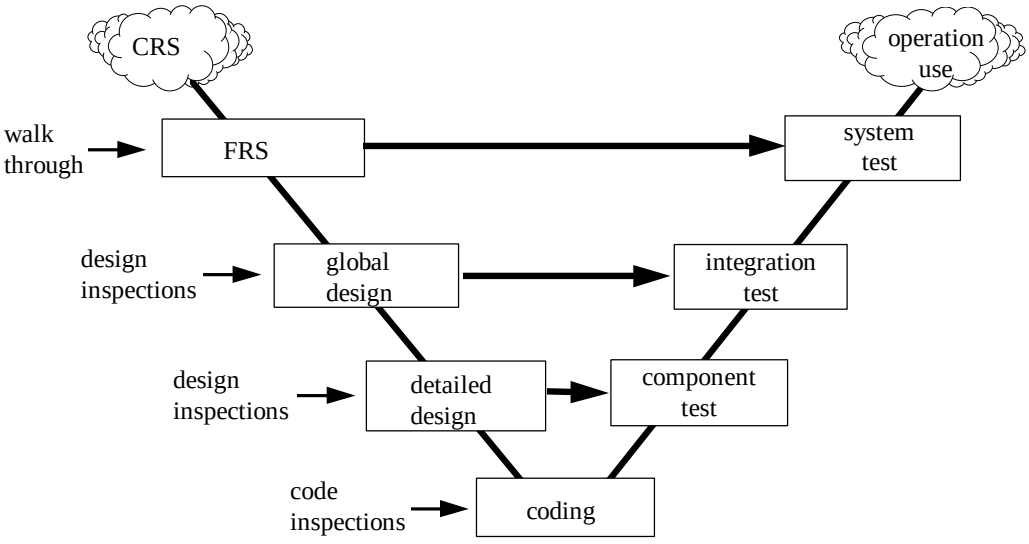

Figure 1: "the V-model"

Requirements walkthrough

At the time the quality strategy was defined and inspections were introduced the Functional Requirements Specification (FRS) document was almost finished. The tight deadline, that was put on the delivery date of the FRS, left little time for internal reviews or inspections by the software development team. However, the FRS was reviewed by means of a walkthrough with representatives of the customer, e.g. product management. Walkthroughs can cover more material than inspections and reviews because the presenter is the author, and the other participants do not have a heavy participating

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

work load. They therefore provide an opportunity for large numbers of people to become familiar with the material. Walkthroughs are used for purposes of communication rather than for discovering defects. Software may be “inherited”. We don't know exactly what’s there, so we organise a walkthrough to go through it, page by page, with the key people in the room. If defects are found, that’s fine, but the main objective is to get familiar with the product. In the TV software project the walkthrough served as a means to get formal agreement with the customer on the documented functional requirements.

Important quality characteristics

In addition to the quality strategy based on the V-model a number of quality characteristics, according to ISO 9126 [6], was identified that needed special attention during software development:

- *efficiency*; as stated before price and therefore code size are important issues within consumer electronics software products. During software inspections special attention was given to the efficiency of the software code. In the final stages of the project a tool was used to track the code size after compilation for the various leading versions daily.
- *maintainability*; due to business requirements like time-to-market and reduction of production cost the importance of re-usable and easy-to-maintain software is high. Within the project a static analyser tool was used to evaluate the number of code lines, number of function calls, cyclomatic complexity, maximum number of control structures and comment density of the developed software components and functions. Acceptance criteria per metric were defined based on commitment by the software engineers. As a result 92% of the software code complied to all criteria and was classified as being excellent.
- *portability*; for the various leading versions three different variations of a certain microcontroller were used (non teletext, teletext and closed captioning). This meant that the interface functions for hardware pinning had to be developed separate from the standard components. To fulfil the customer requirement of one “standard” user interface disregarding the microcontroller used, meant that the basic functionality for user interface was limited to the possibilities of the non-teletext microcontroller. This customer requirement was thus putting constraints on the specifications and design to achieve the required portability.
- *usability*; usability is of course an important quality characteristic within consumer electronics, however this quality characteristic was not mainly dealt with by the software development team. A specialised corporate design unit designed and specified the user interface and human interactions as part of the FRS. The same unit also performed usability testing based on a prototype user interface to improve to usability of the final software product.

Note, reliability was not treated as a separate quality characteristic, the reliability of the software product was considered to be sufficiently covered by the various inspection and testing activities defined in the V-model.

SOFTWARE TESTING

Component testing

Initially is was decided that all software components that were newly developed would be subject to component testing by the software engineers, preferably not being the one who initially wrote the software code. No formal testing techniques would be used, the software engineers themselves were to decide how component testing was going to be carried out using the global and detailed design document as a reference base. However, component test cases that were identified would be embedded in the source code, to enable re-use and reproducibility of the tests performed. A test case can be defined as a set of test inputs, execution conditions, and expected results developed for a particular objective. To assist the software engineer in doing a thorough component test, a test coverage tool was made available. A test coverage tool can provide information on the level of statement coverage that has been reached during testing, which gives the software engineer an understanding of how thorough testing has

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

or has not been. It was decided by the project team that as an acceptance criteria for component testing a statement coverage of 75% was required. However, if the number of defect that was found was relatively high, the requirement statement coverage was raised to 85%. Statistics show that the number of defects found on 75% coverage will double when the coverage is increased from 75% to 85% [7]. The main problem for component testing is having an adequate testing environment, for embedded software this means having dummy components for both hardware and software interface. Developing these dummy component is often very time consuming. In practice, due to time constraints and the problem related to dummy components, component testing was limited to a small number of critical and complex components. To reduce the risk associated with not performing component testing according to the defined quality strategy, it was decided to spend more effort on code inspections and to enlarge the number of test cases for integration and system testing. By defining these actions the risk was considered to be under control, enabling the software development team to nevertheless deliver a quality product on time. The test coverage tool was not used during the few component tests that were carried out. However, it was successfully applied for the reasons stated earlier during integration and system testing.

Integration testing

The integration test aims at testing the interfaces between the various components in an integrated hardware/software environment. For the integration tests a more formal approach was used. The integration test was prepared by the software engineers together with independent test engineers defining test cases for each of the interfaces specified in the global design document and the Hardware Software Interface (HSI) document. For specifying the test cases the data flow testing technique was used [8]. The starting point for the data flow test are the data (triggers) that flow from one component to another by means of a function call. The technique focuses on the various possible values the parameters can have, that are present in the function call. A test case consisted of an identification of the interfaces that are being tested, the input and the expected results (both on a logical level). Relevant specific environmental needs and preconditions were also specified. Test execution was done by the software engineers using the prepared (and inspected!) test cases and logging the results on a test report form.

System testing

The system test had the objective to the test the entire software product against the functional requirements, treating the software product as much as possible as a black-box. Also for the system test a more formal approach was used. The system test was prepared by the software engineers together with independent test engineers defining test cases for each of the functional requirements defined in the Functional Requirements Specification (FRS) document. For specifying the test cases the decision table testing technique was used [8]. The decision table technique is a very formal technique, which means that the test cases are derived from the documentation (FRS) on a deterministic way according to a fixed set of rules. The decision table technique focuses on the completeness and accuracy of the processing. The technique was originally developed for white-box testing, but has also been applied successfully during black-box testing (system and acceptance testing), especially in embedded software environments. The test cases based on this technique were enriched with “error guessing” test cases focusing on invalid and unexpected conditions. To improve the quality of testing, test cases were subject to inspections. Again test execution was mainly done by the software engineers themselves using the test report form as a logging mechanism. All test cases were put under configuration management to assure re-usage of the test cases for regression testing and future related projects.

INSPECTIONS

Implementation

While the importance and benefits of inspections for software projects is well understood within the software industry, only few engineers apply the inspection technique to their personal work. Even when statistic evidence exists, the introduction of improved software methods, e.g. inspections, is often slow

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

because software engineers must be personally convinced of the effectiveness of new methods before they will consistently use them. In software this is particularly true because [5:

- software engineers' methods are largely private and not obvious from the products they will produce. Thus, if they do not use proper methods, it is unlikely that anyone else will know;
- software engineers are generally not trained to follow the planning and measurement disciplines needed to rigorously evaluate the methods they use;
- even when software groups have a common set of defined practices, these practices are not consistently followed;
- the current industrial environments do not as a prerequisite require the use of the best-known software engineering methods.

A principal issue, therefor, is how to motivate and implement inspections within a software organisation. Also within the TV software project a number of software engineers initially showed a somewhat critical attitude towards inspections. Amongst other this was due to the fact that in previous projects they had experienced a somewhat theoretical introduction of inspections, mainly resulting in finding a large number of minor, e.g. textual, defects. As a result inspections were introduced within the project in a pragmatic and realistic way identifying two types of inspections: team review and formal inspections. Both types of inspection were carried out using initiation, preparation, meeting, rework and follow-up as the main phases (a description of the inspection procedures is provided in the next paragraph). However, within the formal inspection a moderator was appointed, roles were assigned and a separate logging and causal analysis meeting were carried out. These issues and aspects were not part of the inspection process when a review was carried out. For both types of inspections a procedure was developed and *maintained* throughout the project by the quality coordinator with supporting forms for the inspection process and defect logging. In principle the author of the design document or software code to be inspected had to decide which of the two inspection methods was going to be used. An exception to the this rule was made when the object to be inspected was larger then 15 pages or when the quality strategy had identified the component as highly important and complex. In these cases the inspection was performed by means of a formal inspection.

However, the implementation of inspections within the Singapore team was another matter. It was experienced within the TV software project that the Singaporese team members were not used to comment software work products made by another team member. Things are taken personal very quickly, which is even stronger during an inspection meeting when more people are present. As a result the decision was made to perform all inspections at Eindhoven. For each software work product that was developed at Singapore a co-author was assigned from the Eindhoven team. The co-author coordinated the inspection at Eindhoven and gave feedback on the results in writing to the original Singaporese author. This way of working proved to be beneficial throughout the project. Note that the observations regarding the Singapore team are derived from one practical experience and can therefore not automatically be generalised towards other co-operations with Asian or Singaporese software development teams.

Inspections carried out

Within the project all design documents, both global and detailed, were inspected. Especially for inspections on the detailed design documents, because of their number, much attention was given to the authors' objective of the inspection, e.g. the specific questions that had to be answered. Emphasis was put on "best practice", discussions on functionality or architecture were not allowed since this would not be very cost effective during inspections. Also all software code was inspected, giving special attention to the issue of code size (efficiency). When performing code inspections optimisation of code size was the main objective, next to conformance to standards. Between the two inspection methods that were distinguished, the team review method was used for code inspections.

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

In total 123 inspections were carried out throughout the projects' life cycle. The total effort spend was approximately nine person weeks for finding the defects and another nine person weeks for doing rework and follow-up. In total no less than 1465 defects were found during inspections. Within the project the defects were classified into three groups:

- *critical*: a defect that is allowed to get through to testing or operational stages if not found during the inspection and has a scope beyond the component that is being inspected;
- *major*: a defect that is allowed to get through to testing or operational stages if not found during the inspection and only has an impact on the component that is being inspected;
- *minor*: a defect that "only" has an impact on the document under inspection; if this defect would not have been found during inspection it would not have resulted in a testing or operational defect.

Note, the more common categorization for defects relating to risk or the amount of rework effort it is not adhered to. In stead the focus is on prevented test defects [10] to facilitate calculation of inspection savings. The category distribution of the defects found, is shown in the table hereafter:

Defect category	**Total number of defects**	**Percentage**
Critical defects	50	4%
Major defects	691	47%
Minor defects	724	49%
Total	1465	100%

Table 1 : "defect distribution"

It can be calculated from the data presented that on average 1 critical or major defect was found per hour spend (including rework). Looking at the differences between team review and formal inspection it is very interesting to see that the average number of defects found per hour for formal inspections is 1.5. This means that although there was some initial opposition towards formal inspections, they showed to be more effective during the project. Data resulting from the engineers' own organisation or project will convince them on process changes needed, e.g. more formal inspection in stead of team reviews, more easily. It was also calculated that on average 0.9 defects were found per page and 27 defects were found per KLOC, only taking the critical and majors defects into account.

THE INSPECTION PROCEDURE

The inspection procedures are based on the generic inspection procedure described in the quality system of the organisation. Adaptations were based on the specific project environment. The procedure for formal inspection consisted of eight steps and the team review procedure consisted of five step. Hereafter, the formal inspection procedure that was developed is described, identifying also the differences with team review procedure. At all places within the procedure where the terminology "document under inspection" is used, this implicitly also refers to "software code under inspection".

- *Request: Initiating the inspection process*
  The inspection process begins with a "request for inspection" by the author to the quality coordinator. The quality coordinator assigns a moderator to the inspection, who will perform an initial entry check. The purpose of the entry check is to reduce the probability that the team will waste scarce resources, only to discover items which could easily have been corrected by the author before the inspection began. In the case of a team review no moderator is assigned to the inspection, the author and the quality coordinator, take care of moderators' activities.
- *Planning the inspection*

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

When the design document or software code has passed the entry check an Inspection process form is supplied by the quality coordinator to the moderator (or the author in the case of a team review). During the planning phase the dates, times and place are set. The participants are selected, informed on these issues and are assigned to one or more so-called “roles”. Roles are used to assure that each participant has a more or less unique viewpoint and discovers unique defects, thus making the inspection process more effective. An inspection team generally consisted of two or three team members, having a mixture of both experienced and less experienced software engineers.

The following basic roles can be distinguished, according to the 1 to 4 model, for inspections:
- focus on higher level documents, e.g. does the design comply to the FRS;
- focus on related documents on the same level, e.g. interfaces between software functions;
- focus on standards, e.g. “best-practice”, internal consistency, clarity, naming conventions;
- focus in use, e.g. for design testability, programmability and maintainability.

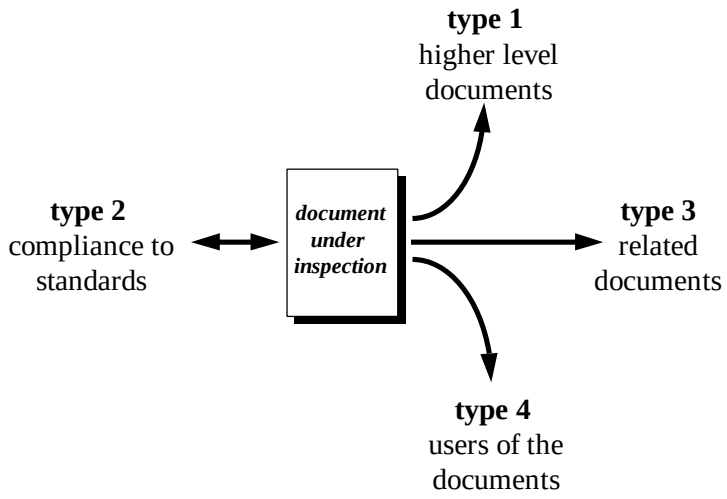

Figure 2 : “the 1 to 4 model”

The author can raise specific issues (roles) that have to be addressed during the inspection process. However, during a team review the concept of roles is not mandatory. It is up to the author to raise specific questions and issues during the planning phase. Because no separate kick-off is held during a team review, the author distributes the document to be inspected and other relevant information at the end of the planning phase and provides the participants with the necessary explanation.

- *Kick-off*
  The moderator distributes the document to be inspected and other relevant documents. The participants receive a short introduction on the objectives of the inspection and the documents. Role assignments and other questions are also discussed during the kick-off meeting. The kick-off meeting may also include feedback on inspection process changes and training on the inspection procedure. As stated before a kick-off meeting is not held during a team review.

- *Preparation*
  The participants work separately on the document under inspection using the related documents, procedures and checklists provided. The defects are identified according to the understanding of the individual participant, and are recorded on the document under inspection. If necessary, these copies of the document will be provided to the author at the end of the logging meeting. This step is identical during a team review.

- *Logging meeting*
  During the meeting the individual defects that have already been identified during the individual preparation are logged on the Logging form by the author. To assure progress and efficiency no real discussions are allowed during the logging meeting. Items that need to be discussed and questions

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

that are raised are logged on the Logging form and are subsequently addressed in the discussion meeting. However, in a team review discussions are allowed to take place during this meeting. At the end of this logging meeting of a team review the author will ask for remarks on improvements, possibly leading to change requests on related documents, standards and the review process. Also the way the follow-up should take place is discussed and agreed upon at the end of the logging meeting during a team review.

- *Discussion meeting*
  The discussion meeting normally takes place immediately after the logging meeting. The discussion items that have been identified during the logging meeting are now addressed. This meeting may include issues on how to solve the detected defects. At the end of this meeting the moderator will ask for remarks on improvements, possibly leading to change requests on related documents, standards and the inspection process. Also the way the follow-up should take place is discussed and agreed upon, if possible including througput time.

  There are four options to choose from:
  - the follow-up is done by the moderator;
  - the follow-up is done by the moderator in Cupertino with the participants;
  - a new team review is required for the document;
  - a new formal inspection has to take place on the same document.

  A separate discussion meeting is not present during a team review.

- *Rework*
  Based on the defect logging the author will improve the product. The author may also make other, not based on the defect logging, improvements or corrections to the document. This step also applies to the team review.

- *Follow-up and exit*
  The moderator, or the quality coordinator in case of a team review, checks whether satisfactory actions have been taken on all logged defects, improvement suggestions and change requests. Although the moderator checks to make sure that the author has taken action on all known defects, he does not have to check the corrections themselves. If decided at the end of the discussion meeting, the design document is once again distributed among the participants to check thoroughly whether the defects are solved in a correct way.

  Finally the document leaves the inspection process and is put under formal configuration management. The moderator completes the Inspection process form and presents it to the quality coordinator, who will use these forms as a basis for reporting and for gathering metrics.

RESULTS

Of course inspections and testing are only a means for achieving the project objectives. Therefor, the most important questions that have yet to be answered are:

- Did the project meet its objectives?
- How did the defined and implemented quality strategy contribute to these objectives?
- What other results were achieved by carrying out inspections and structured testing?

A quality product on time

The primary project objective “a quality product delivered on time” was met by the software development team. Inspections and testing improved the quality of the software product without delaying the project. This is especially true since inspections were used at various development stages, such as global design, detailed design and coding. Using inspections during the various stages helped to catch defects in later stages that had slipped through inspections at earlier stages. Structured testing, by

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

means of well prepared test cases, proved to be successful in finding the remaining defects and showed an extremely high level of coverage. Since the test cases were prepared during an earlier phase, making the test execution phase a very efficient one, the overall throughput time of testing did not exceed it's initial planning. Project statistics also showed that the project did not loose any time on inspections but in fact gained some time (see table 2 "time saved" hereafter). Delivering a quality product on time ultimately meant customer satisfaction, which is very likely to result in an assignment for software development of another TV range.

Reduction of the project risks

One of the main reasons for having a thorough quality strategy and using inspections was to be able to control the identified project risks. In practice inspections proved to be a very powerful means for controlling these risks. The risks that were identified earlier in this paper are again briefly discussed:

- *partly less experienced staffing*; by having both experienced and less experienced engineers on each inspection team a lot of knowledge was transferred during the inspection meetings. Inspections were thus used as a training-on-the-job mechanism.
- *more than one location;* since all software work products developed in Singapore were inspected by the Eindhoven team, information was provided on product quality at a detailed level to project management. This information enabled project management to track product quality and take appropriate actions whenever necessary. Although not initially intended this way, carrying out all inspections at Eindhoven, because of the implementation difficulties at Singapore, turned the problem into an opportunity and became beneficial to the project.
- *project size*; the inspection phases "initiation" and "follow-up and exit" were used as checkpoints for project tracking and oversight at a detailed level. In the beginning of the project there was a lot of discussion about tasks that were "almost done" or "90% done" according to the software engineers. Using the inspection checkpoints as a basis for project tracking made tracking more objective and tangible since the results of a certain task, e.g. a design document or software code, has to be available at the beginning of an inspection. Inspection thus provided a effective means for tracking the progress of the project.

Earlier and cheaper defects

Another reason for applying inspections is to improve productivity. The longer a defect remains in a product, the more costly it is to remove. The only other widely applicable technique for detecting and eliminating defects is testing and since inspections can eliminate defects more cheaply than testing, they can also be used to improve productivity. Within the TV software project data has been collected on the rework effort for critical and major defects during the various development phases. As shown in the graph below the average rework effort (shown in minutes) on testing defects is significantly higher than during earlier phases.

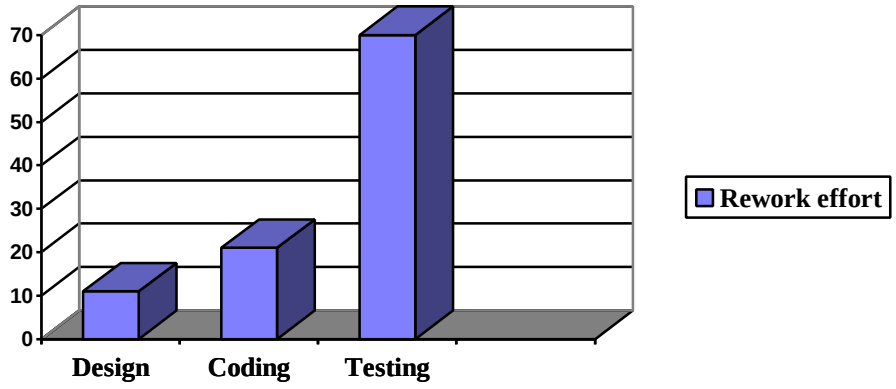

Figure 3 : "rework effort on defects"

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

It is even possible to calculate the time that was saved because inspections were carried out, thus giving an indication of the improved productivity. Although inspections were used for a number of reasons, amongst others improved product quality and risk reduction, the project ultimately did not loose time on inspections. The calculation on the "time saved" is shown in the table below.

Total number of inspection defects found	741
Average time to solve a testing defect	1 hour 10
Potential effort "spent" to solve inspection defects	864
Total effort spent on inspections	720
Effort saved by carrying out inspections	144

Table 2 : "person-hours saved"

A structured testing process

Another benefit, in addition to improved product quality and higher productivity, of using inspections is a more structured testing process. Since a lot of defects have already been detected during earlier phases by means of inspections the initial input quality of the software product to be tested is higher. By doing inspections the chaos that usually exists when testing is starting is substantially reduced. The chaos is most prominent when the "big bang" approach is used and will include things like non-reproducible defects, tests scripts that cannot be completed, unexplainable system failures, test cases that cannot be executed and test cases that are inconsistent with specifications and design. Using inspections within the TV software project meant that most test scripts were fully executed during the first test run.

Testing with the prepared test cases is a prerequisite for reproducible testing defects, making it easier for the software engineer to solve the problem. All versions of the TV software product were subject to a very thorough evaluation cycle, which included at least three full integration and system tests. This of course meant the test scripts and test cases were highly re-used giving a satisfactory return on the initial invested effort for the identification and specification of the test cases.

While performing an integration and system test for a specific software version the code coverage tool was used to measure the statement coverage and thus the effectivity of testing by means of structured testing techniques. All tests (full integration and system test) showed a statement coverage of at least 95%, which is very high compared to the average 40% for commercially released software stated in literature [2]. Note that statement coverage requires every statement in a program to be executed at least once, it is a necessary but in no way sufficient criteria for a quality test. Much stronger criteria are decision coverage or branch coverage.

Improved quality awareness

Both on a managerial and an operational level the project, due to its results, created additional awareness on software product quality issues. On the operational level two engineering groups can be distinguished: the engineers involved in the project and the other engineers. The engineers in the project experienced themselves the benefits of using inspections and performing structured testing. For them it has now become a way of working that goes without saying. Also the other software engineers from the software organisation involved have started to get really interested. Talking to their colleagues who were involved they want to know more about the way the various quality activities were carried out and how to use them in their projects. A separate presentation has been held with them being the audience, explaining and discussing in detail the way of working, results, pitfalls etc.

Since the project was of strategic importance to the organisation, management was watching the project very closely. When the project turned out to be a success, e.g. a quality product delivered on time and a satisfied customer, a presentation, allowing for discussion, was made to management on this topic. Both senior as middle management attended a two hour meeting on software product and project quality! At

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

the end of the meeting is was decided that a number of additional Software Process Improvements (SPI) actions were to be defined and carried out for which budget would be made available. One of the three SPI actions that has been defined, proposed and accepted is the implementation of inspections and reviews throughout the entire software organisation.

Metrics and data

Although often underestimated a very important aspect of each Software Process Improvement program should be measurement. SPI programs often require substantial investments. In the beginning these investments can be justified by pointing to the successes of other organisations. However, after a certain period an organisation should be able to justify the investments by showing that the SPI program has been effective, i.e. that the targets set for the SPI objectives have been met. This can be achieved by defining and implementing corresponding metrics. The TV software project delivered a number of metrics that will be used as a zero-hour measurement, thus describing the current situation of software development within the organisation. Two types of metrics have been delivered:

- quality metrics: metrics related to testing, inspections and the number of defects found per development phase;
- predictability metrics: metrics relating to slip on lead-time, effort and code size.

Defect data that has been gathered throughout the project, both as a result of inspections and of testing, can also be used for SPI. An analysis of the defect data by means of classification by type and cause will result in more knowledge on the development process. It will clearly identify the problem areas thus enabling the organisation to improve the process in a more effective and efficient manner.

CRITICAL SUCCESS FACTORS

It will be clear from the information provided in the previous paragraph that the usage of inspections and structured testing has been a major success within the project. It is of course interesting to find out the reasons for success, especially how the implementation and introduction of new methods and techniques became accepted by the software development team. These issues were discussed with a number of team members, which ultimately resulted into the five most important critical success factors:

- *a separate quality engineer and quality consultant*; no one will ever argue whether quality is important. Nevertheless the required quality is often not achieved. This is because quality is everyone's second highest priority. Unfortunately people have to do overtime to fulfil their first priority, leaving no time left for their second. Hence, the main reason for having someone in a project team that can focus on quality as his first priority. Of course quality is still everyone's responsibility, its the role of the quality engineer to assist the other engineers in being able to fulfil that responsibility. As for any other role quality engineering requires skills, knowledge and experience. Since the project quality engineer did not have all of these requirements, an external consultant was asked to support the project. By doing this project management showed it was really taking product quality seriously and looking upon quality engineering as a true profession.
- *supporting procedures, forms and tools*; nowadays a great number of software organisations have a quality system in place, sometimes certified according to ISO-9001. However, each project has different characteristics demanding a tailored version of the organisations' standard software process to address these specific characteristics. The quality engineer developed and *maintained*, in close co-operation with the development team, specific project procedures and supporting forms. This enabled the engineers to perform their tasks in a more efficient and effective manner. Only the things that were really needed were developed thus restraining from a bureaucratic procedure manual. Supported tools were implemented and used, especially for engineering, testing and configuration management activities.
- *realistic implementation*; as stated every project has its own characteristics, meaning that the standard procedures are not always hundred percent applicable. In each situation it has to be

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

evaluated which parts of the procedures are meaningful and provide added value. Bureaucratic application of procedures only results in opposition. A procedure is only a means and not an objective by itself. However, if the way of working differs from the standard procedures, this has to be documented including the reasons for doing so.

- *management commitment*; no matter how hard one tries, the implementation of inspections and structured testing will not be successful if no management commitment exists. Commitments are not met by inspections, procedures, or tools, however; they are met by committed people. Within the project both the project manager and the project sponsor showed great commitment towards achieving product quality. Commitment is not something that can be taken lightly; it means one has to get involved. Within the TV software project, management demanded inspections and testing on every software product even when time was running out, participated in reviews, attended software quality meetings and arranged additional budget for quality support.
- *the right people*; perhaps the most critical success factor are the people working on a project. A quality product is always realised by means of a number of people working together as a team. The TV software project consisted of a number of engineers who throughout the projects' life cycle evolved to become a strong and solid team. It has been experienced within the project, that having the right mixture of engineers helps in a team building process, when one also want to make changes to the way of working. The combination of experienced engineers, not always willing to make a change but highly skilled and less experienced engineers very enthusiastic and willing to try and learn new things supported by open minded project management and experienced quality engineering proved to be very successful.

If one looks carefully at the above mentioned factors, all of them seem to relate to the implementation and the support delivered to the engineers. Undoubtedly inspections and structured testing were successful in every way in the project. However, making it work in the end all comes down to how to motivate, and implement the methods and techniques in a software organisation with thorough management support. Remember "Willing people make failing systems work, unwilling people make working systems fail".

About the author

Drs. Erik P.W.M. van Veenendaal CISA has been working in as a practitioner and manager within the area of software quality for a great number of years carrying out assignments in the field of quality management, project control, EDP-auditing and software testing. Within this area he specializes in software testing and is the author of a number of papers and books, e.g. "Testing according to TMap® " and "Software quality from a business perspective".

As a consultant to Philips he has been involved in the mentioned TV software project, being responsible for the implementation and coordination of the various quality activities. He is a regular speaker at conferences and a leading international trainer in the field of software quality and testing. Erik van Veenendaal is the founder and managing director of Improve Quality Services, a company that provides services in the area of quality management, usability and testing.

At the Eindhoven University of Technology, Faculty of Technology Management, Erik is part-time involved in lecturing and research activities. He is on the Dutch standards institute committee for software quality.

REFERENCES

1. Fagan, M.E. (1986), Advances in software inspections, in: *IEEE Transactions on Software Engineering*, vol. 12, no. 7, July 1986

2. Fenton, N.E. (1991), *Software metrics: a rigorous approach*, Chapman & Hall, ISBN 0-412-40440-0

3. Gal, R. and M. van Genuchten (1996), Release the embedded software: the electronics industry in transition, in: *International Journal of Technology Management*

4. Humprey, W.S. (1989), *Managing the software process*, Addison Wesley, ISBN 0-201-18095-2

5. Humprey, W.S. (1995), *A discipline for software engineering*, Addison Wesley, ISBN 0-201-54610-8

6. ISO/IEC 9126 (1991), *Information Technology - Software product evaluation - Quality Characteristics and guidelines for their use*, International Organisation of Standardisation

7. Kit, E. (1995), *Software testing in the real world*, Addison-Wesley, ISBN 0-201-87756-2

8. Pol, M., R.A.P. Teunissen and E.P.W.M. van Veenendaal (1995), *Testing according to TMap*$^{\hat{a}}$ (in Dutch), UTN Publishing, The Netherlands, ISBN 90-72194-33-0

9. Pol, M. and E.P.W.M. van Veenendaal (1998), *Structured Testing of Information Systems: an introduction to TMap*$^{\hat{a}}$, Kluwer, The Netherlands, ISBN 90-267-2910-3

10. Rooijmans, J., H. Aerts and M. van Genuchten (1996), Software Quality in Consumer Electronic Products, in: *IEEE Software*, January 1996

11. Weinberg, G.M. (1971), *The psychology of Computer Programming*, New York, Van Nostrand Reinhold

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl