Test-case Driven Inspection of Pre-project Requirements
- Process Proposal and Industry Experience Report

Tony Gorschek
tony.gorschek@bth.se

Nina Dzamashvili - Fogelström
nina.dzamashvili@bth.se

Department of Systems and Software Engineering, Blekinge Institute of Technology, PO Box 520, S-372 25 Ronneby, Sweden

Abstract

Requirements inspections can be used not only for defect removal in projects but also applied pre-project. This to assure that managers have good-enough requirements for product and subsequent project planning activities, such as requirements selection for realization and estimation. This paper introduces an inspection process designed to address the needs of companies with limited resources operating in a market-driven environment. The inspection technique presented here utilizes well-known and recognized concepts like perspective based reading, while also introducing new application ideas. The reuse of testing expertise and inspection artifacts help spread the cost and benefit of inspections over several development phases. Initial experiences from industry application report on positive reactions from managers, testers and developers.

1. Introduction

Inadequate[1] (low quality) requirements can have severe consequences for a product development effort. These consequences are mainly due to the fact that problems in requirements filter down to design and implementation [3-5]. Davis published results indicating that it could be up to 200 times as costly to catch and repair defects during the maintenance phase of a system, compared to the requirements engineering phase [6], and several other sources indicate that inadequate requirements are the leading source for project failure [7-12].

The importance of producing good-enough requirements can be considered as crucial to the successful development of products, whether it be in a bespoke [5] or market-driven [3] development effort – this is generally not disputed. There are however clear indications that requirements quality is lacking in industry and thus low quality requirements being a main contributor to project failure. Both from the perspective of being inadequate as basis for development (in projects), but also in terms of being inadequate as decision support material for pre-project requirements selection and estimations [7-14].

Inspections, first presented by Fagan in 1976 [15], are recognized as a powerful tool with regards to finding and removing defects, thereby increasing quality in development artifacts (e.g. requirements). There are reports indicating that 50-90% of the defects can be caught [16] using inspections.

Equally importantly – inspections offer the possibility to catch defects in the early stages of development thus reducing cost of rework. This is the main reason why there is a general consensus amongst most experience reports and research that inspections on requirements is very beneficial [17, 18], and recommended as a part of the development process, see e.g. CMMI [19] and ISO/IEC 15504 [20].

Furthermore, the recommendation given to organizations with limited resources for reviews and inspections is to prioritize requirements inspections over e.g. design and code inspections [16, 17].

In spite of the seemingly obvious benefits and explicit recommendations inspections are not commonplace in industry, Ciolkowski *et al.* [21] reports that only about 40% perform any type of reviews (on any development artifact). The number for inspections on requirements is even lower.

The explanation to this could be the fact that inspections are often considered as labor intensive and therefore costly process [22]. Moreover the return of investment is not immediate causing skepticism among e.g. industry managers [23].

This paper offers a proposal for an alternative inspection technology, i.e. *Test-case Driven Inspection of Pre-project Requirements* (TCD Inspections). It utilizes test-cases as a tool for inspection efforts - involving the Verification and Validation department (or more precisely “testers”) at an early stage of the development process (pre-project).

The main idea behind TCD Inspection is reuse, i.e. making the inspection efforts and artifacts beneficial to several development stages (both before and during the projects) and thereby “spread” the cost and time demands of the inspection process.

[1] The word ”inadequate” is used as a term denoting requirements that are not good-enough, i.e. of low quality in terms of being incomplete, unambiguous, incorrect, conflicting and so on. All of these issues can be seen as defects In this paper increasing “quality” denotes addressing these mentioned issues i.e. removing defects [1] "IEEE standard for software verification and validation," *IEEE Std. 1012-1998,* 1998.

In summation the TCD Inspection can be characterized by:

- Involving experienced testers in the inspection process, thus reducing the educational and training costs of inspectors.
- Using tester's competence and effort for double purposes, i.e. testing and inspections. In software development projects it's common that testers review requirements specification in order to plan and execute testing activities. Thus using testers in pre-project inspections can be seen as an effective usage of company's resource.
- Producing reusable inspection artifacts: test-cases produced during inspections are to be used in subsequent stages of the project such as during implementation (as an appendix to requirements offering another view to e.g. developers) and testing.
- As managers (writing the requirements) and testers are directly involved in TCD Inspections the learning effect (becoming better at writing and formulating requirements) is achieved.
- By performing inspections pre-project (prior to final selection of requirements for realization) managers get better requirements as input for requirements selection and cost estimations.

This paper presents both the TCD Inspection process and an experience report from industry application.

The structure of this paper is as follows. Section 2 gives an overview of traditional requirements inspections. Section 3 provides analysis of benefits and drawbacks of inspection techniques identified by industry and academia, as well as motivation for finding new approaches for performing inspections. In Section 4 the TCD Inspection process is presented, showing how early requirements inspections are performed using test-cases. In Section 5 a discussion is presented where the traditional and alternative inspection technologies are compared and put against each other in order to clarify the contribution of this paper. The conclusions are in Section 6.

2. Inspection - general overview

This section starts by presenting the classical inspection process as defined by Fagan in 1976 [15]. Next the basics of the Perspective Based Reading (PBR) technique are explained. The reason for focusing PBR is twofold: PBR is identified as an effective reading techniques while performing requirements inspections [24, 25], and second, the TCD Inspection process presented in this paper (see Section 4) is related to the ideas behind PBR.

2.1. Inspection process

Inspections in general can be defined as a process of visual examination of software products with the goal of identifying defects [1, 24]. Software products, in this context, are different documents produced in software development, such as requirements specifications, high level designs, source code, test plans and so on. Defects cover errors, deviations from specifications, software anomalies and so on.

The inspection process itself can be characterized by inspection team members (who they are and their roles) and inspection team size, inspected artifact(s), utilized reading technique and the number of inspection meetings held. Classical Fagan Inspection process includes steps described below [15, 17, 21, 24].

2.1.1. Planning. The inspection team is formed and roles are formulated and assigned. These roles can be summarized in the following manner:

A. **Moderator** - Team manager and inspection coordinator. Selects team members, sets schedule, and assigns resources and so on.

B. **Producer** (a.k.a. Author) – The individual who produces the product (e.g. design, requirements, code and so on). It is the producer's responsibility to assure that the product is ready for inspection. The producer also supports the other parties in the inspection by e.g. answering questions and offering support.

C. **Reader** – The individual who paraphrases the design or code during the meeting. I.e. reads the product being inspected at the inspection meeting and thus supporting the moderator and the inspection process by taking focus of the producer (e.g. alleviating potential problems with the producer becoming defensive).

D. **Inspector** – The individual inspecting the product. The inspector has the responsibilities of thoroughly familiarizing him/herself with the product and inspecting it in order to find and document defects.

E. **Tester** – The individual who inspects the product from a testing point of view.

F. **Manager** – This role helps establish what products (or maybe what parts of a product) is to be inspected and selects the moderator. In some instances this role also takes some managerial weight of the moderator by taking over issues such as resource allocation and training. In addition to role assignments the overall goal of the inspection is formulated [17, 21].

2.1.2. Overview meeting (optional). This optional step is used for familiarizing the inspection participants with the product if needed.

2.1.3. Defect detection (a.k.a. preparation). Each team member reviews the material independently to get an

understanding of the product. This is the phase were the defects are found and documented. Looking at the roles described above roles C, D and E are the ones performing the actual reviews [17].

Subsequent to Fagan's introduction of inspections several reading techniques have been examined and developed as a part of the inspection technology. An overview of these can be seen in Section 2.2.

2.1.4. Inspection meeting (a.k.a. examination). This is the traditional approach for collecting and discussing the defects (not the solutions to a defect) to reach consensus about a joint number of defects. A defect can be accepted as a defect or refused as a false positive. A false positive is an item suspected to be a defect but later on turns out not to be. The moderator documents the findings to enable all issues be passed along to the rework and follow-up stages.

2.1.5. Defect correction (a.k.a. rework). The defects are corrected by the author. Some products may need to be re-inspected and reworked several times before all the defects can be considered to be corrected.

2.1.6. Follow-up. The moderator verifies that all defects are corrected. During this phase there is generally also feedback to the inspection participants as to the inspection results.

2.2. Reading techniques

There are several reading techniques available, and they are mainly used as a way to improve defect finding efficiency in the defect detection step during the inspection. Some of the most well known reading techniques are ad-hoc reading (not really a "technique" per see), checklist-based reading, scenario-based reading and perspective based reading [18, 21, 22, 24, 25].

According to number of studies, out of the above mentioned reading techniques scenario-based reading and perspective-based reading methods have shown to be superior compared to ad-hoc and checklist-based reading, see e.g. [18, 25, 26].

Looking at the specific case of requirements inspections Basili [27] states that two types of reading techniques have been found suitable, i.e. scenario-based reading and perspective-based reading. The reasoning behind this being that the structure and dimensions (perspectives) of these two are superior to checklist-based and ad-hoc reading when it comes to inspections in general, but especially for requirements as they are used by multiple stakeholders (that have multiple perspectives) and are the foundation for the development effort.

Perspective-based reading focuses on conducting inspections using the perspectives of different stakeholders when inspecting a document [24]. A distinguishing feature of this reading technique is that the reviewers build and use models to uncover defects. For example using this technique an inspector can review a requirements specification utilizing the perspectives of a system developer, a system end-user and/or a system tester. In the case of using the system user perspective the created model would consist of e.g. a user manual or use-cases, in the case of the developer perspective e.g. a high level design, and in case of the tester perspective e.g. a set of test-cases [26]. Thelin et al. report on the efficiency and effectiveness of using PBR with the perspective of users (thus creating use-cases for the purpose of performing inspections) [28].

TCD inspection, which is presented in this paper, uses test-cases created by testers for requirements inspection. The detailed description of TCD inspection is found in Section 4.

3. Benefits and issues – inspection experiences from industry and academia

As all technologies, inspections have both benefits and issues (drawbacks/problems). This section presents an analysis of some negative issues and positive benefits related to inspections. The result of this discussion builds a motivation for suggesting TCD inspection, presented in Section 4.

3.1. Benefits of inspections

The overall most significant benefits of inspections is that 50-90% of defects can be caught [16], and equally importantly - they can be caught in the early stages of development thus reducing cost of rework. This is the main reason why there is a general consensus amongst most experience reports and research that requirements inspection is very beneficial [17, 18]. I.e. the recommendation given to organizations with limited resources for reviews and inspections is to prioritize requirements inspections over e.g. design and code inspections [16, 17], this mainly due to the filtering down-effect of defects in requirements. The reasoning being that the earlier problems like incompleteness, ambiguity, errors, conflicts, and so on can be caught the less effort has to be put on fixing the problems and reworking parts that have been influenced by the problems in question. I.e. problems filter down from requirements.

This is not to say that defects can not be introduced later in development, i.e. design, code and so on may also benefit a great deal from inspection.

Another obvious benefit with inspections is that all products (from requirements to code) can be inspected,

whereas other V&V activities, e.g. tests can not be performed on all products as they demand some sort of executability (e.g. execution of code or a formal specification). In addition to these benefits there is a positive spin-off effect with inspections, namely the learning effect. As individuals in an organization perform (or are a part of) inspections they learn about defects, and subsequently can avoid making the same mistakes repeatedly [16, 17].

3.2. Issues with inspections

Seeing the benefits with inspection one might assume that inspections are used extensively in industry as a way to detect and remove defects, thus increasing quality. This is however not the whole truth. Ciolkowski et. al. [21] reports that only about 40% perform any type of reviews.

The main issues with inspections seem to be related to a number of main points that to some extent may explain the reluctance of organizations to adopt inspections in their regular process [21, 29-31]:

- **Time pressure and Cost.** Development projects are under constant time pressure, thus performing inspections (which are generally time-consuming) are not perceived as a clear benefit over e.g. decreasing time-to-market for the product.
  In addition to being time-consuming inspections are labor intensive. An inspection team consisting of e.g. 5 people tie up crucial resources over an extended period of time. In addition inspection result is often tied to the competences of the inspectors, resulting in that good results demand good inspectors. This means that the best results demand that the best personnel be tied up in an inspection during development, not an ideal situation for organizations with limited resources.
  In addition to this the preparation of inspections, documentation and inspection meetings are both labor intensive and time consuming.
- **Difficulty in quantifying the ROI.** Looking at the issues described above the logical response is that the benefit of inspections outweigh the cost regarding needed resources and time for both training and inspection execution. The main issue seems to be that the benefits of inspections are not easily quantifiable. E.g. how is the value of a removed defect in a requirement specification ascertained? There are proponents of stringent metrics collection during inspection as a way to measure benefit [24], but this entails having measurement practices in place over a period of time (which in itself can be very costly [32]) to quantify inspection benefit. There is also the question of organizations with limited resources; on what basis do they initially adopt inspections as a way to increase quality, especially if an alternative investment could be faster time-to-market or the realization of more requirements?
- **Lack of training.** As mentioned earlier, a certain amount of training may be necessary in order for inspectors to master the different reading techniques. In addition to this the moderators need to be trained in the inspection process.

3.3. Motivation to find alternatives

High cost and time-intensive work can be seen as issues that create concern when it comes to the adoption of inspection technology, especially for Small and Medium Sized Enterprise (SME). This is supported by the fact that most experience reports from industry describing inspections are from large organizations with (in comparison to SMEs) vast resources, see e.g. Basili's work with SEL and NASA [25, 26, 31, 32].

The difficulty to measure and quantify benefit contributes to raising the bar, i.e. making inspections a harder sell to management. Without the possibility to convince management of the benefits inspections will not be implemented.

Most recommendations to solving this problem in academia are centered on gathering metrics that will prove benefit. The problem is that this in turn implies that measurement programs have to be in place, i.e. the collection and analysis of metrics over an extended period of time. This in itself is costly and not commonplace in SMEs [32].

The alternative inspection technology suggested in this article aims to address the problem connected with inspection costs via effective use of company's resources. The main idea behind this technology is reuse, i.e. making the inspection's results and artifacts beneficial to as many development stages and people as possible.

This makes it possible to "spread" the cost and time demands of the inspection over several development stages, in an attempt to lower the bar for organizations with limited resources to use inspections as a tool.

In addition the inspection technique aims at producing better quality requirements and thus providing better decision support that can be used directly by company management.

4. Test-case driven inspection

TCD inspection was developed as a part of a SPI activity conducted at Danaher Motion Särö AB (DHR).

The DHR development organization was faced with a market-driven product centered development situation (see Figure 1), characterized by the need to have high quality requirements pre-project. This was crucial in order to provide decision support for requirements

selection and estimation of future projects, i.e. enable product planning activities.

Introducing traditional requirements inspections at DHR, as described in Section 2, (and performing them pre-project – and not after project initiation), would be connected with high cost due to large number of incoming pre-project requirements. In addition, the requirements at this initial product planning stage are not allocated to a project, implying that the explicit decision to implement the requirement has not yet been taken. In order to meet the above described restrictions an alternative inspection process was developed. The overall goal was to achieve high quality requirements pre-project and at the same time keep inspection costs low through reuse. The following section provides an overview of the inspection process introduced and piloted at DHR (for details on the SPI activity at DHR see [33, 34]).

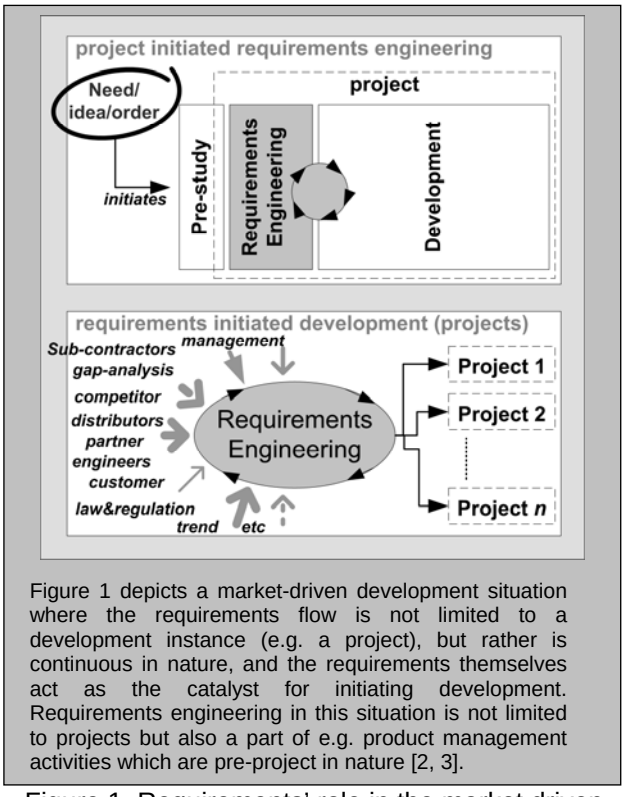

Figure 1 depicts a market-driven development situation where the requirements flow is not limited to a development instance (e.g. a project), but rather is continuous in nature, and the requirements themselves act as the catalyst for initiating development. Requirements engineering in this situation is not limited to projects but also a part of e.g. product management activities which are pre-project in nature [2, 3].

Figure 1. Requirements' role in the market driven development process.

4.1. TCD inspection

The TCD inspection process introduced and piloted at DHR consists of three steps as can be seen in

Figure 2. In Step 1 a product manager(s) initially reviews and selects requirements as an iterative part of the initial formulation/specification. Requirements come from multiple sources (see Figure 1), and some requirements are discarded during this initial step. This is mostly an ad-hoc process where the product manager utilizes personal as well as coworkers' expert knowledge and experience to perform an initial "trial" of the requirements, deciding which requirements will be passed on to Step 2. As a backdrop to this initial trial the product manager is governed by the limitations and goals set up by management through product strategies. They explicitly or (more commonly) implicitly govern the goals and direction of development activities, see [34].

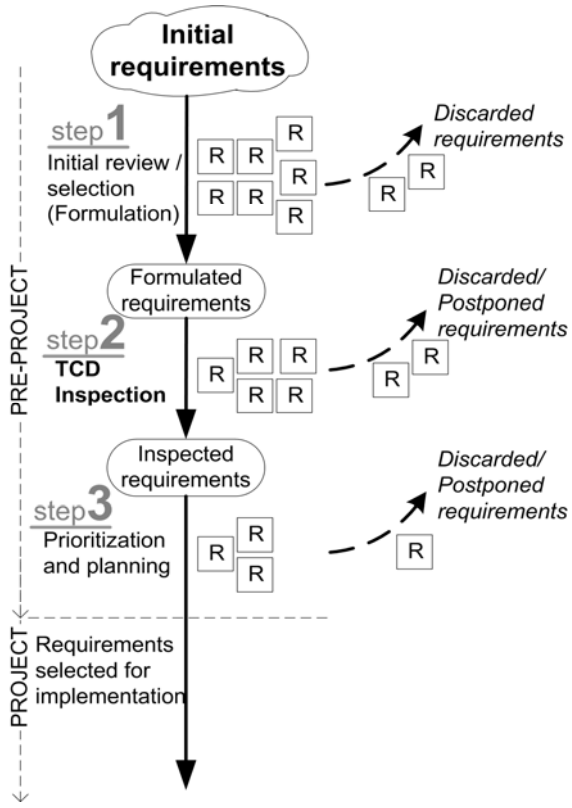

Figure 2. TCD inspection process steps.

Formulated/specified requirements are the input to Step 2. These requirements are specified according to a template and each includes attributes such as ID, Title, Description, Benefit/Rationale, Source, and so on. It is during Step 2 that the TCD inspection itself is performed creating test-cases on the requirements. As the inspection is performed some additional requirements may be discarded or (more often) postponed as a result of the refinement of the requirements (thus adding to the decision support for the product manager). This enables him/her to remove requirements that are inappropriate or of obvious low priority at the time of the inspection. Step 2 will be described in further detail in the following sections.

In Step 3 the inspected requirements are prioritized and project planning is performed. Some requirements

may be discarded or (more often) postponed during this phase also.

The initial review of the requirements in Step 1 is meant to ensure that mostly viable requirements are inspected in Step 2. The requirements inspected refined through test-case creation, enabling dismissal and postponement of some requirements at an early (pre-project stage), as well as passing requirements of higher quality to be passed to Step 3.

4.1.1. TCD inspection roles. The following roles are relevant for TCD inspections:

Product Manager (PM) – this role was chosen based on the fact that PMs elicit and specify requirements. In addition PMs are product experts with an overview of both technical and non-

The PM role in a TCD inspection is comparable to both the Producer and Manager roles as described in Section 2.1.1. The PM is the "producer" of requirements, and in addition decides what requirements are to be inspected (see Section 0).

Tester – this role was chosen due to the fact that testers are also product experts as they test the system continuously and have to understand and possess an overview of it as well as knowledge about details, without which the creation of test-cases and execution of tests is difficult.

An additional major reason for choosing testers to take part in the inspection of requirements is the fact that they are (or at least should be) used to reading requirements as system tests are to be based on requirements [17].

The tester role in a TCD inspection is comparable to both the Inspector and Tester roles as described in Section 2.1.1. The tester inspects the requirements and assures testability.

4.1.2. TCD inspection process (Step 2). The TCD inspection (Figure 2, Step 2) can be divided into several sub-steps comparable to the ones presented in Section 2.1:

Planning - This step involves the PM. The main function of this step is to ascertain what requirements are to be inspected (which requirements are passed to Step 2, see Figure 2), and to allocate resources for this inspection, the main resource being a tester.

Defect Detection - This step involves the tester(s) that inspects the requirements. The tools for the inspection are the creation of test-cases based on the requirement, thus ascertaining:

- Testability - if it is possible to create test-cases based on the requirement, the requirement can be considered as testable.
- Completeness - as the tester is an expert in usage of the system, missed functionality, whether it is whole requirements or parts in a requirement, can be caught.
- Non-conflicting – the tester goes through each requirement thoroughly as test-cases are created. During this the tester can catch issues such as conflicts and inconsistencies.

As the tester performs the inspection test-cases are produced. In addition an inspection protocol is created documenting defects caught during the inspection.

Inspection Meeting - As the inspection is completed a meeting between the inspector (tester) and the owner (PM) takes place. During this meeting the inspection protocol is reviewed and the test-cases created are gone through.

The inspection protocol serves as specification of what is to be corrected, and the defects are confirmed or dismissed.

The test-cases are gone through to establish that a joint understanding of what the requirements entail. The main reasoning behind this is to assure that the requirements are correct, i.e. if both the tester and the PM have the same understanding of a requirement the formulation of it is assumed good-enough. Whether or not it is correct from the requirement's sources view is up to the PM.

Defect Correction - The PM corrects the defects agreed upon during the inspection meeting. If there is a need the requirement's source is elicited for additional information.

Test-case Completion - As the PM completes the corrections the updated/new requirements are delivered to the tester. The tester then completes the creation of test-cases, effectively confirming the corrections and re-inspecting the requirements in question.

In some instances there might be a need for additional corrections if new defects are caught during this step.

As this step is completed the test-cases are saved as attachments to the relevant requirements. The reason for this is twofold. First, the test-cases can be used to augment the requirements, e.g. a developer can use the test-case in addition to reading the requirement to ascertain that he/she understands what is meant. Second, if a requirement is changed, the likelihood of updating the test-case in question is greater if it is directly visible and assessable.

5. Discussion and study results

Looking at the TCD inspection there are several main differences from a traditional inspection described in Section 2. These can be summarized as follows:

1) Inspection roles and team size – TCD inspection has two distinct roles, PM and tester.

2) Reusable artifacts – the artifacts produced during TCD inspections are intended to be reused in later project stages.
3) Pre-project requirements – the TCD inspection was initially developed to make inspections of pre-project requirements feasible.

The combined effects of these features can result in lowering inspection costs, whereas contributing to increased quality of inspected requirements.

5.1. TCD vs. traditional approaches

In this section the main features of TCD inspections are explored and motivated in relation to the generic inspection technology described in Section 2.

5.1.1. Inspection roles and team size. The roles of the TCD inspection are comparable to traditional inspections, but with some differences. The minimal size inspection team in TCD inspection consists of only two persons, PM and a tester. This is not new as such, two-person inspections have been used before [24], here traditionally one person is the Author and one the Inspector. In a TCD inspection the roles of the participants are "double" in nature and explicitly described pertaining to each role's responsibilities.

The PM is both Author of the requirements and Manager (the PM owns and interprets the requirements and often specifies them as well). This is logical since the PM is a manager responsible for the product planning (what requirements are selected for implementation) and the requirements themselves.

The reading technique used during TCD inspections can be seen as perspective-based. The tester inspects the requirements from the perspectives of a tester, i.e. ascertaining testability of the requirement, but there is an added perspective, namely the end-user perspective. As the tester inspects the requirements through the creation of test-cases, functionality is inspected, giving the perspective of system end-users, the benefit of this has been presented in previous studies, see e.g. [28].

To assign a tester the end-user perspective seems logical as the tester is not only seen as a testability expert, but also as a system (functionality) expert and an expert at reading and interpreting requirements, fully capable of determining other perspectives than just testability.

It is important to highlight that in TCD inspections the role of a tester is occupied by a tester, not e.g. a developer inspecting requirements from a tester's perspective, which is usually the case in traditional inspections [18]. This is considered as a benefit for the reasons expressed above, i.e. the competence of a tester regarding inspecting requirements, creating test-cases and ascertaining testability was considered superior to that of e.g. a developer.

It should be noted that two-person inspections have been reported to be as effective as traditional inspections, especially if the two-person team can be considered an expert-pair [29]. The PM and the tester do constitute an expert pair in TCD inspections.

5.1.2. Creation of reusable artifacts. A main benefit of TCD inspections is that the artifacts are not created solely for the purpose of the inspection, but test-cases can be used for several purposes in addition to the inspection. The obvious use of the test-case is of course during the testing of the system. There is however another potentially positive effect, namely that the test-cases are attached to the requirements as they are sent to implementation. The test-cases can be used to augment the requirements offering a better understanding and perhaps even decreasing the chance of the requirements being interpreted differently than was originally intended by the PM.

In some traditional inspections the creation of models (e.g. UML models) have been reported as beneficial as the modeling itself increased understanding and thus increased defect detection rate [16]. In the case of TCD inspections test-case are created, using the same basic idea of inspection through the creation of "models". There is however one basic difference, UML models are generally aimed at inspecting *HOW* something is to be done, not *WHAT* as in the case of requirements and test-cases. Attaching a UML diagram to a requirement and offering it to e.g. a designer could be potentially beneficial, but it could also limit the designer as a UML diagram can be seen as a suggestion of how something can/should be realized, as well as taking focus away from the requirement itself, which should be the basis for what is designed. It goes without saying that models in the form of e.g. UML diagrams are not directly usable during system test, as is the case for test-cases.

5.1.3. Pre - project requirements. As mentioned previously, there are several advantages to concentrate inspection effort on requirements. However, the recommendation of inspecting requirements generally applies to requirements within projects, in which case there is commitment to implement the requirements. TCD inspection could be applied within projects as well, but in the case of the SPI activity at DHR (of which the TCD inspection implementation was a part) the requirements inspected were in product planning state (see Step 2 in Figure 2). This had the advantage of increasing the quality of requirements, and complementing them with test-cases improving the decision support material used for both product planning (selection) and project planning (estimations).

The fact that the inspection was performed at an early stage could also be seen as a disadvantage, as effort was put into inspecting requirements that ultimately may not be implemented (not selected for realization). This is however countermanded by the fact that improved requirements quality increased the chance of irrelevant/not appropriate/high risk requirements were discarded or postponed pre-project (instead of later within project).

Whether or not Test-case Driven Inspection of Pre-project Requirements is advantageous depends on two things, pre-project benefit and cost. Benefit is as the name indicates how much benefit there is to having high-quality requirements as decision support for product planning/project planning activities. Product planning activities include e.g. selecting and packaging of requirements into releases with regards to coupling and cohesion (i.e. cohesion within a package should be maximized, while minimizing coupling between packages).

Project planning activities involve estimations of time and effort with regards to implementation of a requirement, as well as estimating e.g. risk etc.

Cost relates to the ability to reuse the inspection artifacts produced by TCD inspections (test-cases) in projects during development and testing, as well as how beneficial it is to get inspected requirements as input to projects.

5.2. Study results

During the pilot study of the TCD inspection the general view of the participants, which included experienced personnel (one PM and one tester), was that the benefit was substantial. The PM felt that the refined requirements were of very high quality in comparison to the requirements before the inspection, offering much better decision support material for product planning and project planning activities. As the completeness and understanding of the requirements was improved during the inspection it was also possible to discard and/or postpone requirements during Step 2 (see

Figure 2), enabling fewer unviable requirements to pass to Step 3, and thus wasting less resources on planning and estimation activities.

An additional benefit perceived by the PM was a learning effect, making the PM better at specifying requirements in terms of the perspectives inspected.

The tester felt that the possibility to see and ascertain testability at an early stage was far superior to getting the requirements post-implementation, as this offered the possibility to faithfully base system test on requirements that were good-enough for this purpose. In addition, a positive spin-off effect was that the activity of creating a realistic test-plan could be started earlier as the requirements were available and testable.

A drawback identified was that non-functional requirements were hard to inspect. Not necessarily in terms of testability, but from the perspective of conflicts. It was hard to ascertain if e.g. a performance requirement could be in conflict with a security requirement without actually running tests.

The creation of test-cases at an early stage also entails the potential risk as they are vulnerable to change as the requirements may change during development. The test-cases may thus be subject to re-work. On the other hand, as the requirements are inspected, the chance that they are changed due to e.g. misunderstandings may be smaller.

An additional potential benefit of TCD inspections is that “inspection knowledge” can be passed on to parties not directly participating in the inspection. This is achieved through the attachment of test-cases to the requirements. The test-cases, if used by the developers as a requirement complement, are the means of this inspection knowledge transfer.

6. Conclusions

The main goals of TCD inspections are aimed at enabling pre-project requirements inspections (increasing the quality of the requirements) and at the same time keeping time and cost at a minimum.

The process of TCD inspections may involve just two persons, utilizing already existing expertise (tester and PM) thus minimizing the need for training. But more importantly, the major artifacts of the inspection, test-cases, are manufactured during the inspection and can be reused to augment requirements, spread inspection knowledge, enable test-plans to be created at an early stage, and ultimately for performing system-tests.

A noticeable benefit of TCD inspections is that the early quality and completeness increase of pre-project requirements improves the decision support material for market-driven development organizations.

TCD inspection technology is in its initial stages. There is need for additional pilot tests in industry and collection of metrics to measure and quantify potential benefits. The results from the initial pilot were promising, and the work of improving this inspection technology will continue through cooperation with industry, in an attempt to make inspection technology accessible to organizations with limited resources.

7. References

[1] "IEEE standard for software verification and validation," *IEEE Std. 1012-1998*, 1998.

[2] R. Wieringa and C. Ebert, "Guest editors' introduction: RE'03: practical requirements engineering solutions," *IEEE Software*, vol. 21, pp. 16-18, 2004.

[3] L. Karlsson, Å. Dahlstedt, J. Natt och Dag, B. Regnell, and A. Persson, "Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study," presented at Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Essen, Germany, 2003.

[4] G. Kotonya and I. Sommerville, *Requirements engineering: processes and techniques*. New York: John Wiley, 1998.

[5] I. Sommerville, *Software Engineering*, 6 ed. Essex: Addison-Wesley, 2001.

[6] A. M. Davis, *Software requirements: objects, functions, and states*, Rev. ed. Englewood Cliffs NJ: Prentice Hall, 1993.

[7] I. K. Bray, *An Introduction to Requirements Engineering*. Dorset: Addison-Wesley, 2002.

[8] R. L. Glass, *Software Runaways*. Upper Saddle River NJ: Prentice Hall, 1998.

[9] T. M. Connolly and C. E. Begg, *Database systems: a practical approach to design, implementation and management*, 2. ed. Harlow: Addison-Wesley, 1998.

[10] M. Jirotka and J. A. Goguen, *Requirements engineering social and technical issues*. London: Academic Press, 1994.

[11] J. van Buren and D. A. Cook, "Experiences in the Adoption of Requirements Engineering Technologies," *Crosstalk - The Journal of Defense Software Engineering*, vol. 11, pp. 3-10, 1998.

[12] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. Chichester UK: John Wiley & Sons, 1999.

[13] C. Clegg, *The performance of information technology and the role of human and organizational factors - The OASIG Study*. Sheffield: University of Sheffield, 1996.

[14] B. Regnell, L. Karlsson, and M. Host, "An analytical model for requirements selection quality evaluation in product software development," presented at Proceedings of the 11th International Conference on Requirements Engineering, Los Alamitos CA, 2003.

[15] M. E. Fagan, "Design and Code Inspection to Reduce Errors in Program Development," *IBM Systems Journal*, vol. 15, pp. 182-211, 1976.

[16] O. Laitenberger, T. Beil, and T. Schwinn, "An industrial case study to examine a non-traditional inspection implementation for requirements specifications," presented at Proceedings of the Eighth IEEE Symposium on Software Metrics, Los Alamitos CA, 2002.

[17] S. R. Rakitin, *Software Verification and Validation for Practitioners and Managers*, 2. ed. Boston MA: Artech House, 2001.

[18] F. Shull, I. Rus, and V. Basili, "How perspective-based reading can improve requirements inspections," *Computer*, vol. 33, pp. 73-79, 2000.

[19] CMMI-PDT, "Capability Maturity Model Integration (CMMI), Version 1.1," in *CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1)*. Pittsburgh, 2002.

[20] ISO/IEC, "Software Process Assessment TR 15504:1998," vol. 2004. http://www.sei.cmu.edu/iso-15504/: ISO/IEC, 1998.

[21] M. Ciolkowski, O. Laitenberger, and S. Biffl, "Software reviews: The state of the practice," *IEEE Software*, vol. 20, pp. 46-51, 2003.

[22] G. W. Russell, "Experience with Inspection in Ultralarge-Scale Developments.," *IEEE Software*, vol. 8, pp. 25-32, 1991.

[23] D. A. Wheeler, B. Brykczynski, and R. N. Meeson, *Software inspection : an industry best practice*. Los Alamitos, Calif.: IEEE Computer Society Press, 1996.

[24] A. Aurum, H. Petersson, and C. Wohlin, "State-of-the-Art: Software Inspections after 25 Years," *Software Testing Verification and Reliability*, vol. 12, pp. 93-122, 2002.

[25] V. Basili, O. Laitenberger, F. Shufl, and I. Rus, "Improving software inspections by using reading techniques," *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pp. 836, 2000.

[26] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sorumgard, and M. V. Zelkowitz, "The empirical investigation of perspective-based reading," *Empirical Software Engineering - An International Journal*, vol. 1, 1996.

[27] V. R. Basili, "Evolving and Packaging Reading Technologies," *Journal of Systems and Software*, vol. 38, pp. 3-12, 1997.

[28] T. Thelin, P. Runeson, C. Wohlin, T. Olsson, and C. Andersson, "Evaluation of Usage-Based Reading-Conclusions after Three Experiments," *Empirical Software Engineering*, vol. 9, pp. 77-110, 2004.

[29] C. Sauer, D. R. Jeffery, L. Land, and P. Yetton, "The effectiveness of software development technical reviews: A behaviorally motivated program of research," *IEEE Transactions on Software Engineering*, vol. 26, pp. 1-14, 2000.

[30] L. G. J. Votta, "Does every inspection need a meeting?," presented at Proceedings of the 1st ACM SIGSOFT Symposium on Foundations of Software Engineering, New York, 1993.

[31] A. A. Porter, L. G. J. Votta, and V. R. Basili, "Comparing detection methods for software requirements inspections: A replicated experiment," *IEEE Transactions on Software Engineering*, vol. 21, pp. 563-576, 1995.

[32] V. R. Basili, F. E. McGarry, R. Pajerski, and M. V. Zelkowitz, "Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory," presented at Proceedings of the 24th International Conference on Software Engineering (ICSE02), Orlando, 2002.

[33] T. Gorschek and C. Wohlin, "Requirements Abstraction Model," *Submitted to Requirements Engineering journal.*, pp. N/A, 2004.

[34] T. Gorschek, "Software Process Assessment & Improvement in Industrial Requirements Engineering," in *School of Engineering - Department of Systems and Software Engineering*. Ronneby: Blekinge Institute of Technology, 2004, pp. 154.