Can using Fagan Inspections improve the quality of specification in 2011? A Case Study

S. Jeary, K. Phalp, F.Milsom, L.Hughes, S. Webster, J.Holroyd

Software Systems Research Centre, Bournemouth University,
Fern Barrow, Poole Dorset, BH12 5BB
kphalp;sjeary;fmilsom;lhughes;swebster;jholroyd@bournemouth.ac.uk

1 Abstract

In this paper, we explore why Fagan Inspections have become obsolete in the software industry, given the body of evidence which supports their use to improve the quality of software artefacts and the software development process.

Since the late 1970's, much has been written about how Fagan Inspections improve the quality of both processes and outputs of the software development process. The literature indicates that the Fagan Inspection technique can improve quality of software (or other software development artefacts) by a reduction in defects of 60 – 90%. However, recent literature suggests that inspection techniques in general and Fagan Inspections in particular, are no longer used. A study in 1998 found that respondents used inspections either irregularly or not at all. Teams often review artefacts informally, but believe that they are performing an inspection or formal review. The lack of rigour in the review process results in reduced benefits and more defects in the artefacts.

To explore this situation, we conducted a case study with a local enterprise and we report on the early findings. These suggest that the introduction of Fagan Inspections may have a number of benefits before they have even been introduced fully, including recognition of flaws in the current development process, development of technical knowledge relating to the software process domain, and improved team relations and a 'quality' culture. In addition, the personnel using Fagan Inspection gain experience in the production of 'quality' artefacts.

1.0 Introduction

Inspections were originally created as a measure of validation and verification for software by Fagan in 1972 in an attempt to improve software quality and programmer productivity [1]. The software development process was, at that time, considered by Fagan to lack discipline and rigour. He, thus, introduced statistical quality and process control which he had previously used in industrial hardware [2] to address these problems. He initially introduced the use of a rigorous and disciplined process for source code and pseudo code production, and later realised that the same methods could be used for further 'upstream' processes such as architecture and requirements, to give even greater efficiencies; although he gives neither details nor examples [3]. Fagan reports that experience has shown that in the first ten years, results of inspection usage reported that 60-90% of defects were found [4].

There are two studies which provide specific evidence on the use of Fagan's inspection on upstream components, one on planning and requirements [5] and a second on requirements [6]. Both studies show that the use of inspection for specification brings both economic and intangible benefits. Interestingly, whilst software inspections are agreed by most software developers to be a foundation for software quality [7] and our own informal research agrees; many newer developers have never actually applied any inspection techniques in their software development process. Indeed, in our experience many of the more senior developers, who know of inspection techniques and have used them in the past, do not use them now. This is confirmed by Johnson who carried out an informal survey in 1998 and found that 80% of respondents either practiced inspection irregularly or not at all [8] and is backed up by [9].

This paper will discuss the different kinds of inspection in the literature in Section 2, before describing the Fagan Inspection process itself. It will point out the benefits that result from using Fagan Inspection at the requirements stage of the development process. Section 4 will discuss, within a case study, the setting up and early stages of the use

of Fagan for requirements in an enterprise before Section 5 gives details of early findings where the current software process is described by one of the business analysts as 'chaotic'. Finally, in Section 6 we will conclude and describe the next stages of the process.

2.0 Overview of inspections

This section first delineates reviews, walkthroughs and inspections, before giving an overview description of Fagan Inspection. Descriptions of some of the most popular methods are also given and we suggest how they appear to have built on, or echo the work of Fagan.
There has been some confusion regarding the terminology of reviews, walkthroughs and inspections. This has given rise to some issues with the perceived efficacy of inspections because reviewers have actually been completing a walkthrough.
The IEEE standard 1028-1997 [10] is still relevant here, and in Section 3 a number of different types of reviews are defined.

- The basic review is:

'A process or meeting during which a software product is presented to project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval'.

Also described are variants on this, the management review that is a systematic evaluation of the process and the technical review that is a systematic review of the product.

- A walkthrough is defined as:-

'A static analysis technique in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems'.

- An inspection is defined as:-

'A visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications. Inspections are peer examinations led by impartial facilitators who are trained in inspection techniques. Determination of remedial or investigative action for an anomaly is a mandatory element of a software inspection, although the solution should not be determined in the inspection meeting'.

Weigers believes that reviews are more rigorous than walkthroughs and describes how a walkthrough is often only a presentation by the author, and that of the three review types they are the least successful at identifying bugs [9]. A walkthrough though can, be effective at removing some defects and thus be considered to improve software quality as can a badly completed inspection. If either technique leads to an initial improvement in software quality, many companies do not explore the further potential for improvement [2] or they get mediocre results that reflect their informal approach [4]. The point is that each type of review has a different purpose. Walkthroughs are very useful early on in the development process not only to identify defects but also to determine how to solve problems. They are quick and cheap to perform. They can be very useful for certain things but they should not be used on their own. They should be used in conjunction with other techniques, or used for 'low risk' artefacts.

The inspection is the most formal of the three validation and verification techniques and is based on Fagan's concepts of a disciplined process to detect defects. The use of different techniques must take into account the resources, the time available and the need for quality. Interestingly, we found no empirical work describing which technique to use in a given situation. However, the expected benefits of inspection over other forms of review include measurably higher product quality, greater productivity from people in development and maintenance, and shorter more predictable development times.

Fagan inspection is the most cited inspection method in the literature and is the basis upon which all the other methods are designed (or at least appear to be). It can be used in any phase of the development lifecycle and involves a team with a moderator. The moderator is a key role and not only manages the team but provides leadership; and should be trained. There are a number of defined roles including the moderator, the author of the code, a reader who translates the design into code and finally the tester for writing or executing the tests. These roles were later broadened to include the other phases of the lifecycle when it was realised that defects introduced at earlier stages of the lifecycle, cost more money to re-work [4].

The Fagan Inspection process involves five phases.

1. Overview which consists of group education of the participants as to what is to be inspected and roles are

assigned.
2. Preparation where participants learn the material and prepare for their roles.
3. Inspection consists of finding the defects.
4. Rework of the defects by the author.
5. Follow up where verification is made by the moderator that the fixes are effective.

Whilst the inspection session should last no longer than two hours the whole process from end to end may take up to 100 people hours [1]. The inspection process, roles and content should be rigorous in their planning and use as shown in Figure 1, and if the criteria are not met then it is not an inspection [11].

```
|A Fagan Inspection, or simply 'Inspection', is a formal and        |
|rigorous review.                                                   |
|Its performance is budgeted and planned.                           |
|Its primary objective is to identify defects in artefacts so they  |
|can be removed.                                                    |
|The artefacts to be inspected are believed complete and correct by |
|the artefact's originators.                                        |
|It uses a defined process with five steps: - overview (optional),  |
|preparation, inspection, rework, and follow-up.                    |
|It uses knowledgeable peers...                                     |
|...who are trained to perform their inspection roles.              |
|It has defined entry and exit criteria.                            |
|It uses checklists and (optional) standard s to define correctness.|
|                                                                   |
|It is well defined in IEEE Std. 1028-1997 'Standard for Software   |
|Reviews', pp 13-20.                                                |
|Any procedure that does not satisfy all these criteria may be an   |
|effective technical review or walkthrough, but is not an           |
|inspection.                                                        |
```

Figure **1: Definition of a Fagan Inspection [11].**

Whilst there has been considerable work done to show how the introduction of inspections is effective at removing defects there is little to show that it is cost effective and there is little empirical evidence [12].

There have been a number of surveys of Inspection Methods and each has a different perspective for example a review on the cost of using the different methods [12] or, as another example, a review on the state of the art [7]. However the most complete survey is that by Laitenberger and DeBaud who give useful information for both academics and practitioners [13].

A summary of three of the most commonly known methods [7] is shown here for completeness, each method building on criticisms of Fagan Inspections. First, the Active Design Review (ADR) [14]. This was created by Parnas and Weiss [14] to address what were believed to be the problems with Inspections in the design phase, which they considered shallow and ineffective. The problems included such things as overloaded reviewers being given too much information and not having time to understand the information they were given, reviewers being unfamiliar with the goals of the design of the artefact they were inspecting and reviewers inspecting areas of the artefact they were unfamiliar with at the expense of those they were. In addition there were the issues of holding large meetings (which need to be controlled [14]. The ADR consists of several smaller reviews and involves reviewers working in a single technical area (usually of their expertise), alone and being asked questions by the designer to check their comprehension. Each mini-review has three stages, firstly brief details of the product are provided, this material is studied by the reviewers following the guidelines and finally the reviewer meets the author and other relevant reviewers to discuss issues in small meetings. There is no evidence of its use.

Second, the *N*-Fold Inspections method [15]. The *N*-Fold method is carried out the same way as a Fagan Inspection but N teams inspect a requirements document using checklists. Martin and Tsai recognised that all the methods of finding defects worked well with code, but they were not as successful with requirements documents; and in addition they believed that early detection of defects increased both quality and reliability of the subsequent design and code [15]. A moderator is appointed and supervises all the teams. The teams inspect the requirements document and are expected to find some of the same faults. Each fault is recorded by the moderator once in a database. Only one team is responsible for the subsequent re-work. They recognised that the process could be done for documents and code

further downstream. The findings of their work show that the best value for *N* depends on the availability and cost of additional teams and the potential cost of not finding a defect during the inspection. Later work by Schneider, Martin and Tsai [16] conducted an experiment and found that a single team found 35% of defects in a requirements document but nine teams detected 78% of defects.

Finally, the Phased Inspection method [17, 18]. Knight and Myers [17, 18] stated that inspections, such as Fagan Inspections, could identify defects but that there was a lack of rigour. They suggested that whilst Fagan Inspections may have been effective at finding defects the quality of the final product could not be determined. They listed a series of other issues with Fagan Inspection including the fact that inspection concentrates on defects and does not focus on other important points such as maintainability and portability. In contrast, the Phased Inspection method ensures that the product is inspected in a series of phases that determine if the product has the desirable properties necessary at completion of that phase; the inspection cannot move to the next phase until this phase is complete. A Phased Inspection consists of six phases. The most important point is that any inspection is rigorous, and should be defined by a rigorous planned process; this allows inspectors to understand exactly what is needed and ensures the process is repeatable.

Whilst all the additional methods have some merits, there is little documentary evidence of the use of the methods. In view of this, and the fact that one of the authors is a Fagan Moderator and trainer, the method used for the study will be the standard Fagan Inspection.

4.0 Case study: application of Fagan Inspections

Whilst the use of Fagan Inspections can give very good improvements in defect detection, there has been little work on the use of Fagan Inspections in an Agile environment except that of Gilb [19] whose method is more about sampling a part of the specification and motivating requirements engineers to produce less defects. One of the unique properties of Fagan Inspections is that it encourages the writers of artefacts to learn and understand the defects that they are likely to make and to recognise a good quality artefact. Much of the work that has been carried out in the area of inspections is quantitative in nature and is either proving the benefit of the use of inspections or trying to show the cost-effectiveness. There is very little that looks at the qualitative features and shows the human costs and benefits, the problems that occur and the issues that are involved in changing the culture to introduce inspections. The studies as part of Gilb and Graham [2] are the exception; however they are not necessarily still valid (for example one is still using tape input). The aim of this work is thus to explore the introduction of Fagan's inspection into an enterprise using more qualitative principles.

The enterprise under scrutiny is a large organisation with a discrete IT department that develops both its own applications and uses external products that meet the requirements of the users within the enterprise. The organisation of the IT department is based on projects and each team of developers has a project manager. One team describes itself as working in a lean and agile way while the other uses a more traditional approach. The team of five business and systems analysts wish to introduce Fagan Inspections to see if it will be a useful technique. They believe that if they can show an improvement in specification, then it will demonstrate its usefulness to both project managers and developers.

The work is based on a case study approach as defined by Yin [20] and, as he suggests at the outset of the study, the research question should be articulated. The research question is thus; will the introduction of Fagan Inspection into an organisation in 2011 reduce the number of defects and increase the confidence of the developers in their product?

Yin's [20] framework identifies the unit of analysis and the propositions as the next steps in further defining the boundaries and scope of the case study. The unit of analysis ensures that the researcher explores the problem of outlining the case for the study. The unit of analysis becomes apparent when the research question that needs to be answered has been accurately identified (Yin, 2009). The unit of analysis for this study is the introduction of Fagan Inspection into one enterprise. By creating questions in proposition format, Yin's framework forces the researcher to identify what should be studied (2009). The propositions for this study are:

- Introduction of Fagan inspection will reduce the number of defects in specification
- Introduction of Fagan inspection will increase designer and developer confidence
- There will be issues when Fagan Inspection is used with an agile development team
- The use of Fagan Inspection will improve departmental team work

For our ongoing study data will be collected using a variety of different methods including the use of interviews, the study of documents and the contents of discussion boards and finally observation. Clearly, there will be some degree of subjectivity in analysis of the evidence to support (or refute) our propositions. For example, we will attempt to rate confidence of the team before and after Fagan inspection use, Similarly, effectiveness within an agile team will depend upon opinions of a variety of stakeholders. In seeking quantitative evidence we intend to note the effectiveness of inspections (in terms of amount and rate of defects found) and to compare defects found with similar internal projects.

However, even at this early, preliminary stage (pre-implementation of inspection) the organisation has noted a number of benefits (now reported as our initial findings).

5.0 Initial findings

This study is still at an early stage, but the findings are already interesting and show that introduction of something like Fagan Inspections is a challenge. Firstly, during initial interviews, the enterprise development process was described by one analyst who observed it as being '*like CMMi level 1….chaotic and…. 'ad-hoc'*. The IT department had been reorganised over the past few years and all the business analysts had been with the Enterprise for less than 18 months except one who has completed 2 ¼ years.

The way the developers worked had changed from a version of SCRUM to being Agile and Lean with one team being more traditional. '*Projects come into existence in a variety of ways, some of which are controlled via a Change Management Board (CMB)…… there is no clear shared understanding of this process….. and it is currently under review. There are a variety of business change projects involving IT. Some are procurement and implementation of major IT systems, while some are agile web service developments'*.

The analyst produces a requirements document following interviews and creates it as a specification that meets the stakeholder's needs, it is then agreed by those stakeholders, and the analyst is usually not involved with it again. '*I don't know what issues there were with what I had written or the individual requirements…. I don't know if the final product meets their* [stakeholders] *needs…..I don't know how those requirements are evaluated…. or tested'*. Therefore there are no starting figures of the number of defects, no measures of failed tests, nor figures of the number of requirements not being met by the final product.

Bush reports that Fagan inspection works well when 'sold' from both bottom up (by training developers and demonstrating how useful inspections can be) and 'top down' (by training managers in the benefits) [21]. However, the introduction of Fagan inspection in this enterprise is likely to be neither top down, nor bottom up – it has emerged the analysts are unsure of exactly where they are in the IT department. Interestingly a diagram drawn by one of the analysts shows the analyst sitting in the centre, but having no link to the company strategy, the enterprise architecture, the project portfolio management nor surprisingly, the systems development life cycle.

The first step for any inspection is to identify a set of inspection rules, what would make a good inspection checklist? The analysts did some research and one produced the list shown at Figure 2 based on a useful web page [22]. Having been circulated to the analysts there was a good discussion about the points that the list showed. Interestingly, the first point related to the list having a project-based perspective, and the analysts believing that they needed a broader perspective that took account of strategy, architecture and portfolio management and that this perhaps needed to be reflected somehow in the list. One of the analysts showed a sound understanding of the principles of specification by raising a number of issues for discussion.

- Vocabulary
  Different business areas have their own locally well understood, vocabularies which exist in 'pockets'. A common, controlled vocabulary could be developed across all projects; thus adding use of a common vocabulary as a check across all projects.
- Traceability
  With business requirements, consideration needs to be give to the traceability of the requirement and it needs perhaps to be formalised as part of the inspection process. The possibilities are: owner, user, source, enterprise objectives and project objectives. The analyst argued that each business requirement should be traceable to a specific project goal supportive of the project vision and in turn that goal should trace back to an Enterprise goal supportive of the Enterprise vision. These would need to be agreed and signed off by the project sponsor.

- Readiness for review
  The point at which the inspection should occur was highlighted. There was a lack of a reusable methodology for the analysts across different projects. Where would a technical review take place?
- Links to other products
  Which other products should be linked to the requirements document and should they be part of the inspection? Products include business rules, stakeholder analysis, process models, controlled vocabulary, user stories, conceptual model, and textual overview.
- Feasibility
  Boehm [23] believes that feasibility is essential, 'to establish the consistency and conceptual integrity of other elements' as requirements evaluation criteria and architectural design criteria. Where should the architectural review happen? Should a completed review be an entry criterion to the inspection?
- The requirements set
  The final set of inspection criteria hides complexity in its simplicity, and completeness could be a discussion on its own. However, prioritisation as a project attribute could be an important criterion to assess.

```
|1.    Is a simple, complete, well-structured sentence that         |
|a.   States one thing and states it well                           |
|b.   Does not contain conjunctions (and, or, but, . . .)          |
|c.   Avoids the use of limiting phrases (unless, except, . . .)    |
|d.   Does not depend on other sources of information              |
|e.   Contains subject, verb, object and appropriate modifiers      |
|f.   Defines what an actor should or should not do OR is an        |
|external      constraint that must be enforced                    |
|2.    Emphasises "what" should be done, not "how" to do it and it  |
|a.   Avoids preconceived solutions                                 |
|b.   Describes business logic (rules), not the technology needed   |
|c.   Expresses the what (destination), not the how (journey)       |
|3.    Targets components that are in scope for your project        |
|a.   Defines a desired behaviour or feature of a component of the  |
|system                                                             |
|b.   Is within your authority to implement                         |
|c.   Does not impact out-of-scope components                       |
|4.    Is understandable, unambiguous and clear                     |
|a.   Has a single possible interpretation                          |
|b.   Is easily understood by knowledgeable peers                   |
|c.   Avoids confusion                                              |
|d.   Is written to the readability level of the target audience    |
|5.   Is objectively measurable                                     |
|a.   Clearly states what the solution has to do                    |
|b.   Defines acceptable behaviour for the solution in measurable   |
|terms                                                              |
|c.   Specifies what data the solution creates or consumes          |
|d.   Relates constraints to the functional, performance or         |
|informational elements that they impact                            |
|6.   Is one of a complete, internally consistent, non-redundant,   |
|set of prioritised requirement statements                          |
```

Figure **2: Initial requirements checklist. Adapted from [22].**

One analyst, in the style of Barnard [5] showed a recent set of outline high level requirements and by applying the criteria to them, it became apparent that the criteria will be extremely useful.

There followed considerable discussion about the best way to introduce Fagan Inspection in terms of getting buy in from the teams. There is no further budget to train the whole department at this stage; therefore it was decided to identify a suitable project to pilot the inspection principles.

6.0 Conclusions and further work

The project is ongoing but the findings at this early stage demonstrate a number of points that have a resonance with the findings of Doolan [6]. He reports that inspections give employees on the job training in standards, technical material, the culture is improved by open working processes and finally inspection – the learning by doing it. Finally, he believes that inspection can identify the root causes of defects and thus modifications can be quickly introduced to change the software development process to ensure the defects do not happen again.

In our case study, the process of introducing Fagan Inspections has given a focus for discussion that has shown the lack of standards and is improving the body of knowledge amongst the different team members. In addition, the team are realising the issues that they have with the development process as it currently stands, and already have ideas about how to improve it. These ongoing discussions will add to an improvement in the quality of business and system analysis, requirements, specification and process. Finally, the discussion and the meeting series have created a stronger team with a better knowledge of the standard of work that they wish to achieve.

7.0 References

1. Fagan, M.E., Design and code inspections to reduce errors in program development. IBM Systems Journal 15 (1976)
2. Gilb, T., Graham, D., Software Inspection. Addison Wesley, Harlow, England (1993)
3. Fagan, M.E., A History of Software Inspections. In: Broy, M., Denert, E. (eds.): sd&m 2001. Springer (2002)
4. Fagan, M.E., Advances in Software Inspections. IEEE Transactions on Software Engineering 12 (1986)
5. Barnard, E., One Person Getting Started. In: Gilb, T., Graham, D., Finzi, S. (eds.): Software Inspections. Addison Wesley, Harlow, England (1993)
6. Doolan, E.P., Experience with Fagan's Inspection Method. Software Practice and Experience 22 (1992) 173-182
7. Aurum, A., Petersson, H., Wohlin, C., State of the Art: Software Inspections after 25 years. Software Testing, Verification and Reliability 12 (2002) 133-154
8. Johnson, P.M., Reengineering Inspection. Communications of the ACM 41 (1998) 49-52
9. Wiegers, K.E., Improving Quality Through Software Inspections. Software Development (1995)
10. IEEE, IEEE Std1028 -1997: Standard for Software Reviews The Institute of Electrical and Electronics Engineers, Inc., New York, USA (1998)
11. SPIN, SPIN UK Fagan Inspection Workshop Report. In: Shelley, C. (ed.). British Computer Society: Software Process Improvement Network, London (2010)
12. Porter, A., Siy, H., Votta, L., A Survey of Software Inspections. Advances in Computing 42 (1996) 70-76
13. Laitenberger, O., DeBaud, J.-M., An Encompassing Life-Cycle Centric Survey of Software Inspection. Journal of Systems and Software 50 (2000) 5-31
14. Parnas, D.L., Weiss, D.M., Active Design Reviews: Principles and Practice. International Conference of Software Engineering (ICSE). IEEE Computer Society, London, England (1985) 132-136
15. Martin, J., Tsai, W.T., *N*-Fold Inspection: A Requirements Analysis Technique. Communications of the ACM 33 (1990) 225-232
16. Schneider, G.M., Martin, J., Tsai, W.T., An Experimental Study of Fault Detection in User Requirements Documents. ACM Transactions on Software Engineering and Methodology 1 (1992) 188-204
17. Knight, J.C., Myers, E.A., Phased Inspections and their Implementation. ACM SIGSOFT Software Engineering Notes 16 (1991) 29-35
18. Knight, J.C., Myers, E.A., An Improved Inspection Technique. Communications of the ACM 36 (1993) 50-69
19. Gilb, T., Agile Specification Quality Control: Shifting emphasis from cleanup to sampling defects. Proceedings of INCOSE Conference, Rochester NY, USA (2005)
20. Yin, R.K., Case Study Research: Design and Methods, Vol. 5. Sage, Los Angeles (2009)
21. Bush, M., Improving Software Quality: The Use of Formal Inspections at the Jet Propulsion Laboratory. Software Engineering, Nice, France (1990)
22. RSG, (2011). Six Simple Rules for Writing An Effective Business Requirement, [online]. Available from: http://www.requirementssolutions.com/WhitePapers/RulesForEffectiveRequirements.pdf [Accessed: 17 January 2011
23. Selby, R.W., Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research. Wiley-IEEE Computer Society Press (2007)