A case study of Agile practices applied to a project seeking DO-178 certification

Kobus Coetzee
Graduate School of Technology Management
Faculty of Engineering, Built Environment
and Information Technology
University of Pretoria
kobus@embeddedfool.net

Dr Fritz Solms
Department of Computer Science
Faculty of Engineering, Built Environment
and Information Technology
University of Pretoria
fsolms@cs.up.ac.za

ABSTRACT

This study measured the feasibility of applying agile software development practices to high reliability software projects seeking DO-178 certification. We surveyed the current uptake of agile practices within DO-178 practicing communities, and did a case study comparing two student developer teams completing two projects according to the same specifications. One team used traditional waterfall development methodologies while the other used and agile methodology we proposed for use in safety critical and regulated industries. The teams were compared on the Robustness, Functionality delivered, Solution complexity, Lines of code, Word count and Quality of the documentation delivered. We found difficulties with the agile methodology, and provide a reference for further studies.

CCS Concepts

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Keywords

Project management, Agile, Waterfall, DO-178

1. INTRODUCTION

Software development practices have progressed through a great many iterations since the first conference on software engineering sponsored and facilitated by NATO in 1968 [2]. Since the 1970's the development of software for military use has been standardised by the US Department of Defence, eventually leading to the generation of thousands of military standards of the form MIL-STD's [13]. These MIL-STD's for software developed eventually led to the creation of derivative standards for software development.

But since software engineering spans such a wide field of application domains, several best practices have developed for each field, with standards bodies regulating standard implementation of the practices for a particular domain; for example the RTCA (Radio Technical Commission for Aeronautics) specified standards being used in the aeronautical industry, MISRA (Motor Industry Software Reliability Association) specified standards being used in the automotive industry and the ISO (International Standards Organisation) regulating medical device software. Both these standards bodies has as their objective to guide their industries to ensure the development of products that are safe, reliable, robust and adheres to minimum operational performance requirements. There is a tradeoff and that is that this comes at the cost of flexibility to adapt to changing requirements, cost increases and speed to market.

The RTCA has published the DO-178C [15] specification for "Software Considerations in Airborne Systems and Equipment Certification". MISRA has published the MISRA-C:2012 [14] specification for "Guidelines for the use of the C language in critical systems". ISO has published the IEC 62304 [11] specification for "Medical Device Software - Software Lifecycle Processes".

These standards prescribe a strict adherence to process and fixed design and implementation life cycles. However software development practices in other application development domains has moved towards increasing flexbility, improved software delivery times, software quality and client satisfaction through agile development methodologies[1]. Software development methodologies such as Scrum and Kanban can in some ways be considered the agile parallels to the RTCA and MISRA standards.

Even so, agile development is only practiced in a certain subset of software industries, mainly consumer software development [7]. Two software industries that have remained largely immune to agile development practices and still mainly done with 1970's era waterfall methodologies is military and safety critical software. Contrast for instance RTCA's and MISRA's focus on safety, reliability, robustness and minimum operational performance mainly through rigid process and verification, with the agile manifesto:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Coetzee, Solms '16 Johannesburg, South Africa

© 2016 ACM. ISBN TBD.

DOI: TBD

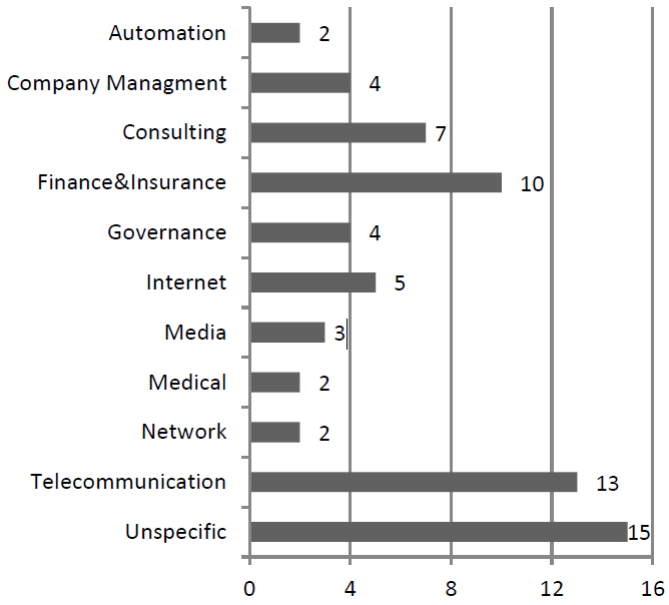

Figure 1: Agile distribution across domains [7]

- *Individuals and interactions over Processes and tools*
- *Working software over Comprehensive documentation*
- *Customer collaboration over Contract negotiation*
- *Responding to change over Following a plan*

That is, while there is value in the items on the right, we value the items on the left more." [3]

As such, applying agile methodologies to safety critical and regulated industries requires careful considerations of the tradeoffs that agile ethodologies bring to the software development process.

2. LITERATURE SURVEY

Very little research has been generated on the application of agile methods to highly regulated industries. Diebold [7] did systematic study on agile penetration in various industries.

Note the low occurrence of agile in the automation and medical industries. But problematic also is getting to a clear definition of what exactly agile software development entails. If the difference can only be regarded as iterative vs. incremental development practices, then agile methods have been in development since the 1970's [12]. The current most popular agile development methodology is SCRUM, a process which specifies short development sprints of roughly two weeks in which certain developmental goals are specified and the team attempts to satisfy these goals. Cardazo [5] did a literature review and found SCRUM did to lead to productivity improvements, but did not investigate if organisational make-up or the industry the company operates in have any effect. Also these findings are still very recent, and more in depth analysis is required.

A couple of case studies have been performed on implementing agile methodologies in regulated industries. Fitzgerald [8] found tension points when implementing a SCRUM process in a case study where the company and product had to comply with several FDA, ISO, IEC and ISPE regulations. These tension points were in quality assurance, safety and security, effectiveness, traceability, verification and validation. This is not surprising since agile trades off flexibility and time to market for decreased performance in these very attributes. As such a pure agile process is not suitable for safety critical or regulated software development, but agile methodologies can be made suitable and even bring about improvements to these industries if tailored correctly.

Shenvi [17] also investigated if improvements can be made in the development of medical equipment software when employing techniques from six sigma and agile, while complying with regulatory standards. A theoretical model was developed, and employed as a case study, resulting in improvements in productivity and a decrease in defects in the final product software in the period the model was used between 2010 and 2013. It is unknown if wider applicability of the model exists for other industries with other certification requirements.

Hrgarek [10] pointed to further difficulties companies face in the medical device software industry when trying to employ agile methods while staying compliant with regulations. It is unclear if the case study was successful in employing agile methods into the development process.

Glas [9] investigated a case in the application of agile methods in a aviation manufacturing setting, and identified open tool platforms and open design to be beneficial. The paper also mentioned regulatory difficulties as a problem for agile adoption, although this was not the focus of the study.

Sfetsos [16] did a literature survey and investigated which of the different agile tools does indeed provide improvements in product quality, and found TDD and pair programming to be beneficial. It should be noted that the study was performed in 2010, and a lot of improvements to agile has happened since then.

Causevic [6] investigated uptake of TDD in industrial settings, and found the following limiting factors: increased development time, insufficient TDD experience/knowledge, lack of upfront design, domain and tool specific issues, lack of developer skill in writing test cases, insufficient adherence to TDD protocol, and legacy code. It is uncertain what is meant with an industrial setting, and if the companies involved was already engaged in other agile practices.

Boehm [4] shows that a balance between planned and agile methodologies can be found and formulates a strategy to arrive at such a balance. This paper is quite dated, especially given the velocity that agile is developing at, but represents a seminal work in blending agile and planned approaches from which several derivative works were produced.

But even with the relatively slow uptake of agile in DO-178 projects, some work has been done on what high reliability agile would look like. The FDA (US Food and Drug Administration) released TIR45 [**?**], which prescribes the application of agile software development practices for the development of safety critical medical devices seeking IEC 62304 certification.

Furthermore Open-DO [**?**] is an initiative for the discussion of the application of agile methodologies to the DO-178C specification. Open-DO is facilitated a conference in 2010 in France for this purpose amongst others.

In addition to agile methodologies such as SCRUM and Kanban that can be thought of a bigger strategies that might

Table 1: Roles

Client	0
Project manager	26
System architect	7
Developer	25
Quality assurance	9
Configuration management	1
Tester	11
Certification authority	10
Independant contractor	6

be able to replace the waterfall process that is traditionally followed in safety critical software development, improvements have also been made in developmental tools that aid in agile development. These tools can be thought of as tactics and include agile centric project management software, CI (Continuous integration), CD (Continuous deployment), Unit testing, TDD (Test driven development), BDD (Behaviour driven development), pair programming and others. So while the strategies (Scrum, Kanban, Waterfall) is mostly mutually exclusive, these tactics (TDD, CI, CD etc) is mostly interchangeable and deciding on the adoption of these tactics can be evaluated on the tradeoffs involved that is important to a project.

3. SURVEY ON AGILE UPTAKE IN DO-178 PROJECTS

For the purposes of this study survey responses were collected to answer the following questions about the current state of the DO-178 certified software industry:

1. Is there a case for agile development within the DO-178
2. Is agile methods already being used within DO-178?
3. To get a benchmark to measure if agile methods improve DO-178

The survey was advertised on various DO-178 special interest groups and mailing lists, technical news aggregator sites as well emails sent to the authors personal contacts active with DO-178 development (This explains the high amount of South African respondents).

The survey was completed by 88 respondents, in various roles, with various team sizes and in various countries:

Most respondents indicated they are still following the DO-178B specification and haven't switched over to the newer DO-178C specification which was released in January 2012.

3.1 A case for agile

To determine if there is a case for the use of agile within DO-178, we looked at the current performance of DO-178 projects. This was done by asking the survey participants the following questions:

- What was your last projects budget performance?
- What was your last project's delivery date performance?
- How many features was delivered according to the original specification?

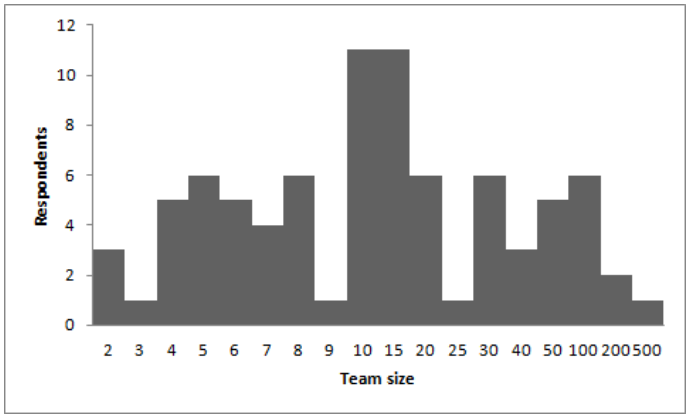

Figure 2: Team size

Table 2: Countries

USA	30
South Africa	16
France	8
Turkey	6
Europe	5
UK	5
Canada	4
Germany	3
India	3
Poland	2
Russia	2
Spain	2
Brazil	1
Israel	1
New Zealand	1
Norway	1

Table 3: DO-178B and DO-178C use

Answer Choices	Responses	
DO-178B	84.71%	72
DO-178C	15.29%	13
Total		85

#	Another (i.e. IEC 62304)	Date
1	With all of of CRI that bring thet certification perimeter equivalent to DO-178C	2/11/2015 11:13 AM
2	+ DO-297, ARP4754, ARINC653 (IMA)	2/2/2015 9:29 AM
3	DO-17B/C, IEC-61508	1/27/2015 3:51 PM

- Was there any requests for additional features during the development? (Scope creep)
- Was there any requests for additional features after certification was complete? (Scope creep)
- In your opinion, does DO-178 increase the costs to a project? By how much?

These questions all measure the traditional improvements that agile can bring to a project, and as such any difficulty with these metrics should show that there is a case for the implementation of agile methodologies. The question is off course, at what cost.

3.2 Is agile being used

We also wanted to find out to which extent agile techniques were already being practiced in the industry. But

Table 4: What was your last project's budget performance?

	< 30%	< 10%	On budget	> 10%	> 50%	> 100%	> 150%	> 200%	Total	Weighted Average
(no label)	0.00% 0	9.59% 7	21.92% 16	28.77% 21	26.03% 19	8.22% 6	2.74% 2	2.74% 2	73	4.18

#	Other (please specify)	Date
1	15%	2/25/2015 7:23 PM
2	Project was scrapped	2/15/2015 6:51 PM
3	very difficult to state as it was a very complex system with many variables	2/13/2015 7:49 AM
4	-	2/6/2015 11:49 AM
5	Unknown: no access budget information	1/29/2015 5:43 AM
6	Mostly on budget, our projects are Fixed Firm Price	1/27/2015 3:54 PM

Table 5: What was your last project's delivery date performance?

	< 70%	< 90%	On time	> 10%	> 50%	> 100%	> 150%	> 200%	Total	Weighted Average
(no label)	0.00% 0	4.05% 3	31.08% 23	31.08% 23	20.27% 15	6.76% 5	1.35% 1	5.41% 4	74	4.15

#	Other (please specify)	Date
1	5%	2/25/2015 7:23 PM
2	Project was scrapped after serious budget cuts at company	2/15/2015 6:51 PM
3	very difficult to state as it was a very complex system with many variables	2/13/2015 7:49 AM

Table 6: How many features was delivered according to the original specification?

	< 60%	< 70%	< 80%	< 90%	100 %	Total	Weighted Average
(no label)	5.33% 4	6.67% 5	6.67% 5	20.00% 15	61.33% 46	75	1.00

#	Other (please specify)	Date
1	Features was removed, but replaced with additional requirements.	5/19/2015 3:40 PM
2	Requirements did change, but were replanned and budgeted for.	2/17/2015 7:08 AM
3	Definitely scope creep.	2/13/2015 7:49 AM
4	-	2/6/2015 11:49 AM
5	Some feature details were changed, removed or added. Mostly this were refinements made to the system to be able to practically achieve the original features. However one a few features were added.	2/6/2015 10:08 AM
6	Additional features were specified by the client after the development was already contracted. All original features as well as new features were delivered in the final product. Hense the +/- 40% Over Budget/Time performance.	2/2/2015 6:57 PM
7	A complete circuit changed after flight tests.	1/29/2015 10:13 AM
8	Some features were changed during project lifetime and also new features were added. (around %10 new)	1/29/2015 9:36 AM
9	Scope creep has occurred on >90% of projects I have worked	1/28/2015 6:04 PM
10	Large amount if scope change throughout the program driven from system level changes and modification of intended functionality driven by customer	1/28/2015 3:47 PM
11	We adjust price and schedule if project changes	1/27/2015 3:54 PM

Table 7: Was there any requests for additional features during the development?

Answer Choices	Responses	
No	13.33%	10
Yes	80.00%	60
Yes it was discussed, but the costs was deemed too expensive to implement.	6.67%	5
Total		75

#	Other (please specify)	Date
1	CRI paper from certification authority	4/13/2015 12:28 PM
2	request by certification authority	2/2/2015 9:34 AM
3	Not that I was aware of (i.e. scope creep was small).	1/29/2015 5:45 AM
4	Authorized changes accepted only	1/27/2015 4:00 PM

not everybody in the industry agrees as to the same definition of what an agile development approach is. As such we asked the respondents if they were using a waterfall, rational unified process, kanban and scrum approach. We correlated this metric by asking respondents to confirm if they were using various developmental tools, some being agile orientated and some waterfall orientated. These responses were then correlated with the developmental approach responses, and measured for correlation.

We also asked the respondents if they felt the development approach was helpful or not in their completion of

Table 8: Was there any requests for additional features after certification was complete?

Answer Choices	Responses	
No	51.35%	38
Yes	39.19%	29
Yes it was discussed, but the recertification costs was deemed too expensive	9.46%	7
Total		74

#	Other (please specify)	Date
1	But was handle as a new project.	2/17/2015 7:11 AM
2	Certification is in final stage, thus not possible to say yet, but seems unlikely as code has been unchanged for quite some time.	2/6/2015 10:08 AM
3	Additionally-wanted features were planned for future releases.	1/29/2015 5:45 AM
4	Treated as a new project	1/27/2015 4:00 PM

Table 9: In your opinion, does DO-178 increase the costs to a project?

Answer Choices	Responses	
Yes	76.71%	56
No	12.33%	9
Not sure	10.96%	8
Total		73

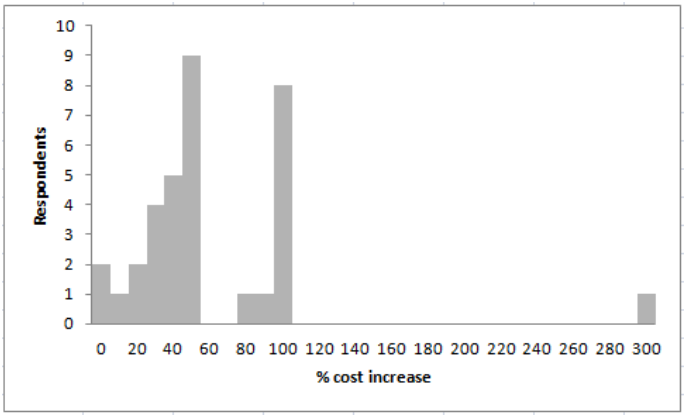

Figure 3: By how much does costs increase?

their projects:

Note the percentages do not necessarily add up, as respondents may have filled in for multiple projects or multiple approaches may have been used within the same project (subcomponents for instance).

3.3 DO-178 benchmarks

To measure the case study performance, the following benchmarks were gathered from the survey:

- What is your $ cost per line of code?
- What is your bugs per 1000 lines of code?

Off course these benchmarks presents difficulties of their own, such as inconsistent measuring, but the hope was that the benchmarks could be used as a measure to compare with the case study discussed later. Unfortunately this proved not to be that helpful.

4. CONCEPTUAL MODEL

The following conceptual model were developed, taking inputs from TIR45 [?] and Open-DO [?], but also taking

Table 10: Waterfall vs agile use

	Used	Helpful	Not sure	Not helpful
Waterfall	81%	44%	18%	19%
Scrum	27%	22%	3%	2%
Kanban	22%	15%	5%	2%
RUP	31%	23%	5%	5%

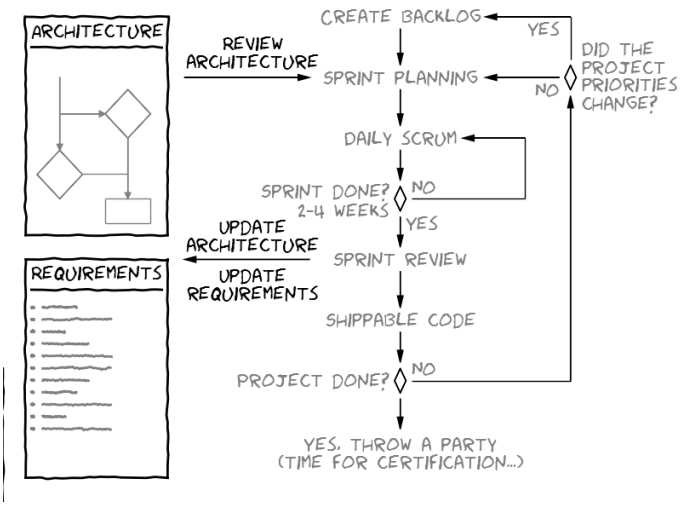

Figure 4: Agile DO-178 framework

general agile development principles, especially SCRUM and applying it to the required DO-178 outputs and objectives. Careful considerations were made to take into account the tradeoffs involved, while not sacrificing safety and traceability considerations.

The conceptual model is focused on bringing continuous integration and continuous deployment to a DO-178 environment. The purposes is to reduce costs, risks and increase flexibility through automation of repeated processes. Since DO-178B is a software quality assurance standard, not a software development standard, it does not impose any restrictions or considerations on how software is to be developed. It does however require the following list of deliverables, with the requirements for each depending on the criticality level chosen:

Organisational deliverables (These can be re-used across multiple projects, if the projects are similar enough off course):

- Software configuration management plan (SCM)
- Software quality assurance plan (SQA)
- Software requirements standards (SRS)
- Software design standards (SDS)
- Software code standards (SCS)
- Software verification plan (SVP)

At the start of a project - Specification phase:

- Plan for software aspects of certification (PSAC)

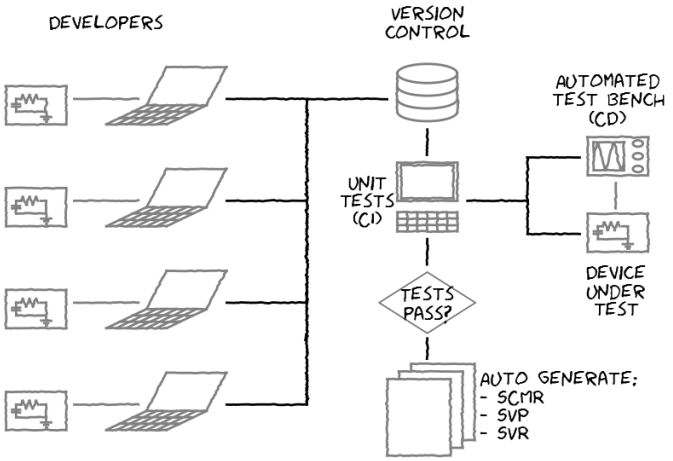

Figure 5: Agile infrastructure

- Software development plan (SDP)
- Software requirements data (SRD)
- Design description (DD)

During development - Implementation phase:

- Source code
- Object code
- Software verification Cases and Procedures (SVCP)
- Problem reports
- Software Quality Assurance Records (SQA)
- Software Configuration Management Records (SCMR)

At the end of a project - Testing and verification phase:

- Software Verification Results (SVR)
- Software Life Cycle Environment Configuration Index (SECI)
- Software Configuration Index (SCI)
- Software Accomplishment Summary (SAS)

The automated generation of some of these deliverables requires the development of infrastructure and tools that will facilitate a more agile way of working.

These automated generated outputs still has to comply with the DO-178 requirements on outputs as stipulated in the Annex A of the specification.

To limit the scope of the case study (since it was after all only a student project), only the following deliverables were required:

- Plan for software aspects of certification (PSAC)

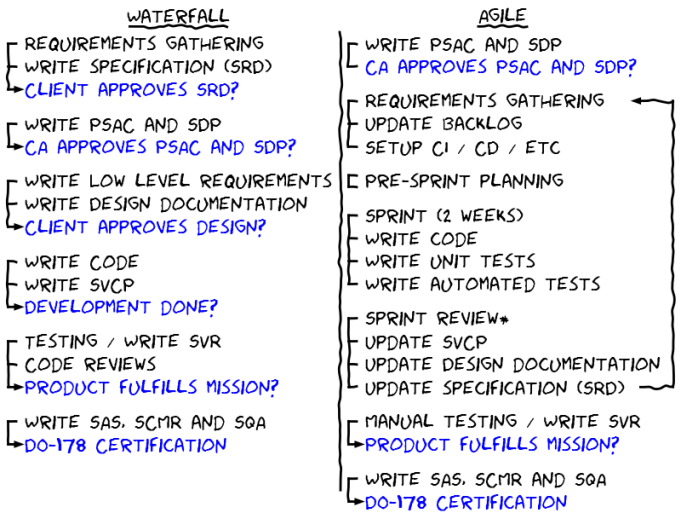

Figure 6: Waterfall vs Agile comparison

- Software development plan (SDP)
- Software requirements data (SRD)
- Design description (DD)
- Source code
- Object code
- Software Verification Results (SVR)
- Software Accomplishment Summary (SAS)

5. RESEARCH DESIGN AND METHODOLOGY

In order to test the effect that a more agile project management approach would have on a project requiring DO-178 certification, a case study was launched. In this case study two 3'rd year student teams developed and delivered a capstone project according to exactly the same specification, with the one team running the agile workflow and the other a more traditional waterfall workflow.

The teams were tasked with developing group chat functionality for the Linphone open source project [**?**], and the final deliverable were measured for robustness of the solution, functionality delivered, maintainability, code bulk and the documentation bulk delivered.

Robustment were determined with independent functional testing of the solution, as well as the amount of unit tests and functional tests each team generated during development.

Functionality delivered was a simple count against the original specification provided to each team.

Maintainability was determined by calculating the cyclomatic complexity of their code, as well as asking each team to give an estimate as to the effort required to develop additional functionality.

Code bulk is simply the count of lines of code delivered, documentation bulk is the word count of the documentation. Documentation also includes code comments.

Because the teams are not identical, in their size or their capability, a normalization task was given to each team. In this task, they were given exactly the same programming assignment to be completed in roughly half a day. The teams were measured on how long they took to complete the task and the quality of their final solution (how many bugs were still present). All case study metrics were then scaled against the results of this normalization exercise.

6. CASE STUDY

The 3rd year computer science students at the University of Pretoria has a year long subject where they complete a project as a team. This is the students first substantial project working as a team. These students are presented with a list of projects they are allowed to tender for, on this list was included two projects for this case study. Both projects listed exactly the same requirements, but it was stipulated that the one project will be run agile and the other waterfall. So some self selection of the students biased towards agile or waterfall workflows are present in the case study. For the tenders, six teams applied, three for each project. The two best applications were selected for this case study.

The teams were guided by faculty towards the completion of their projects (Dr Solms, Stacy and Freda), and monitored for their ability to work together. The waterfall team presented better in this regards with a more dilligent work ethic, and better team cohesion. The agile team had difficulties assigning roles to the team members but later in the project managed to pull together.

The agile team selected consists of six members and the waterfall team selected consists of five members. Team members from both teams have some experience with class projects they have completed, served as tutors in various subjects but haven't worked on any large scale projects (multiple team members). Neither teams have experience with DO-178 certified software projects, and have only a theoretical understanding of agile and waterfall development practices.

The teams submitted their tenders at the beginning of May 2015, and were notified of their successful application a week later. The first meeting between the teams and the client took place end of May after which the teams started their projects in earnest.

The project assigned was for the development of features on the linphone open source project (TODO reference). Linphone is a mature Android application (TODO say more about linphone). The teams had about 6 months time from then on to complete the following list of identical requirements:

- Group chat (Invite additional members to a chat, all members receive chats)
- Secure group chat (AES256)
- Creation and deletion of groups
- Voice record and send over IM
- Rework the messaging user interface
  - Improve spacing between words
  - Make the text bigger
  - Block indents required to better specify who said what

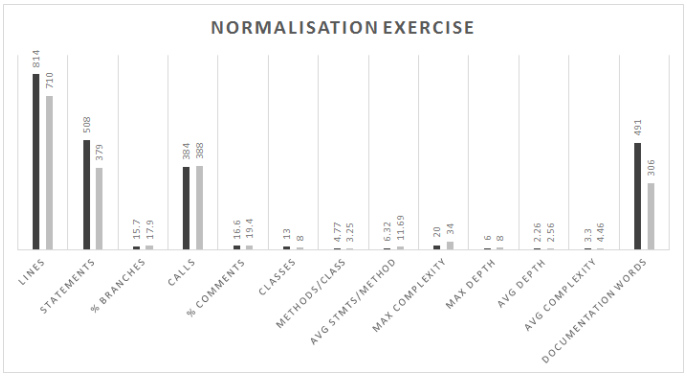

Figure 7: Normalisation results

- Presence indication to show a remote user is typing
- User picture portraits

Emphasis was placed with the student groups that the core requirement was the group chat functionality, then the encryption of the chats, and that all other requirements were of lesser importance. This was done to ensure both groups progress through the work similarly and that one group don't complete the easy work and the other the more difficult work items, which would make comparison difficult.

7. NORMALISATION OF THE GROUPS

The results from the case study were normalised to account for the differing skill levels of the student teams. The normalisation was done by having each team do a sample project taking 4 hours, and comparing their output. The students delivered the executable, source code, a user manual and the code had to be commented sufficiently.

From the normalisation exercise we can account for the differences in the teams by a factor of 26% for code bulk, and 38% for documentation word count. Qualitative comparison for instance documentation quality and robustness of their solution is difficult to account for but it is assumed the difference will be about the same as it too is a function of the teams ability and work ethic.

8. RESULTS

The final projects were demonstrated by the students, and all deliverables and project artifacts were handed in. In the figure we can see a comparison of the projects on the metrics of:

- Lines of code
- Lines of comments
- Cyclomatic complexity
- Documentation word count

From these quantitative measures the teams differed by approximately 50%, that is the waterfall team delivered twice as many lines of code, documentation and working functionality than the agile team. From the normalisation exercise to account for the difference in team strengths we would have expected a difference closer to half of that.

results.png

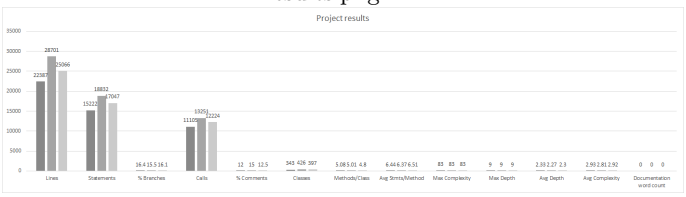

Figure 8: Project results

With project completion both teams demonstrated their projects to industry and their academic study leaders. During the demonstration the waterfall teams project delivered more functionality and held up better under testing. On the other hand the agile team project had a better user experience compared to the waterfall project, but contained an architectual weakness, a single point of failure in the design.

9. DIFICULTIES THE TEAMS ENCOUNTERED

Both teams struggled with the amount of code already present in linphone, and finding their way in the project (this case study was a brown fiels project). The teams also struggled somewhat with building and running the linphone project. Both teams also struggled a bit with git, especially with linphones use of git submodules which can be a tricky subject.

Furthermore each team struggled with different aspects specific to their project methodology:

9.1 Waterfall team

The waterfall team had what is typically known as an integration party, where the work items from each team member is added together at the end. The estimated work effort to get all these elements to work together was underestimated causing some last minute stress on the team.
The team also underestimated the amount of time required for sufficient testing at the end with proper acceptance test procedures.

9.2 Agile team

The agile team had trouble delivering functioning code with each sprint. This was due perhaps to lack of understanding of agile workflow, poor work ethic or overestimating their ability with each sprint. This was also further caused because of their difficulty with setting up a continuous integration server.
The team had difficulty setting up a continuous integration server, this was mostly due to the complicated build setup the linphone project requires.
The team also had difficulty with a test driven development methodology, and subsequently not enough unit and functional tests were written during development.

10. CONCLUSION

One cannot generalise from a single case study, and as such this study cannot speak with sufficient authority on the effectiveness of agile development methodologies versus traditional waterfall methodologies. Nevertheless this study does hint that agile development is not a silver bullet for all project types, and this might explain its poor adoption in highly regulated and safety critical industries.

There is value in the methodology that waterfall forces on a team, especially in getting a team to do proper scoping and architecting of the solution, and in doing so getting the team comfortable with the application domain. This study proposed a variant of agile that was followed in the case study. The agile team had shorter iteration cycles than the waterfall approach, but had an architectual weakness. Since quality attributes of the software is so important for safety critical and regulated industries, it indicates that more emphasis on architectual design is required in this model.

The case study was uncontrolled for the effect that a green field vs brown field project have on the result. The case study was done on a very mature and large existing project. Furthermore some of the difficulties the teams faced stemmed from their inexperience, which might also affect the results of this study.

The study also had two further weaknesses. The first is that the survey had difficulty measuring agile uptake in the industry because of the poor understanding of the definition of agile amongst respondents. Furthermore even though the case study normalises the results to account for the differences in team strengths, it is unknown if the differences scales linearly between the normalisation exercise and the final case study results.

The true value this case study provides is as a reference for further investigation into the effectiveness of agile and waterfall methodologies, and some of the pitfalls one can avoid when conducting such a study. Additional studies can be used to compare results with this one, and in so doing build a case for the effectiveness in either agile or waterfall development practises, especially in safety critical and regulated environments.

Acknowledgment

The authors would like to thank...

- Mr Kobus Steyn for his contribution on DO-178 considerations and best practices.
- Ms Vreda Pieterse (lecturer) and Ms Stacy Omeleze (assistant lecturer) for their help with running the case study.

The following B.Sc Comp Sci students for participating in the study:

- Patience Mtsweni
- Lerato Molokomme
- Tsepo Ntsaba
- Mpedi Mello
- Lutifyya Razak
- Ephiphania Munava
- Izak Blom
- David Breetzke
- Paul Engelke
- Prenolan Govender
- Jessica Lessev

This study was completed as partial fulfilment of the requirements for the degree of:

Master of Engineering (MEM)
in the
GRADUATE SCHOOL OF TECHNOLOGY MANAGEMENT,
FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY,
UNIVERSITY OF PRETORIA

11. REFERENCES

[1] O. Armbrust and D. Rombach. The right process for each context: objective evidence needed, 2011.

[2] P. D. F. L. Bauer. Software engineering. In P. Naur and B. Randell, editors, *NATO SOFTWARE ENGINEERING CONFERENCE 1968*, 1968.

[3] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, and R. Jeffries. Manifesto for agile software development. 2001.

[4] Boehm and R. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[5] E. Cardozo, J. Neto, A. Barza, A. FranÃğa, and F. da Silva. Scrum and productivity in software projects: a systematic literature review. In *14th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2010.

[6] A. Causevic, D. Sundmark, and S. Punnekkat. Factors limiting industrial adoption of test driven development: A systematic review. In *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, pages 337–346, 2011.

[7] P. Diebold and M. Dahlem. Agile practices in practice: a mapping study, 2014.

[8] B. Fitzgerald, K.-J. Stol, R. O, #039, Sullivan, D. O, #039, and Brien. Scaling agile methods to regulated environments: an industry case study, 2013.

[9] M. Glas and S. Ziemer. Challenges for agile development of large systems in the aviation industry, 2009.

[10] N. Hrgarek. Certification and regulatory challenges in medical device software development, 2012.

[11] P. Jordan. Standard iec 62304 - medical device software - software lifecycle processes. In *Software for Medical Devices, 2006. The Institution of Engineering and Technology Seminar on*, pages 41–47, 2006.

[12] C. Larman and V. R. Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, 2003.

[13] C. McDonald. From art form to engineering discipline? a history of us military software development standards, 1974–1998. *Annals of the History of Computing, IEEE*, 32(4):32–47, 2010.

[14] MISRA. Guidelines for the use of the c language in critical systems, 2004.

[15] I. RTCA. Do-178c / ed-12c, 01/05/2012.

[16] P. Sfetsos and I. Stamelos. Empirical studies on quality in agile practices: A systematic literature

review. In *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pages 44–53, 2010.

[17] M. Shen, W. Yang, G. Rong, and D. Shao. Applying agile methods to embedded software development: a systematic review, 2012.