

2015-09-30

# An Exploration of the Use of Gamification in Agile Software Development

Alan McClean  
*Technological University Dublin*

Follow this and additional works at: <https://arrow.tudublin.ie/scschcomdis>



Part of the [Computer Engineering Commons](#)

---

## Recommended Citation

McClean, A. (2015) An Exploration of the Use of Gamification in Agile Software Development, Masters Dissertation, Technological University Dublin, 2015.

This Theses, Masters is brought to you for free and open access by the School of Computing at ARROW@TU Dublin. It has been accepted for inclusion in Dissertations by an authorized administrator of ARROW@TU Dublin. For more information, please contact [yvonne.desmond@tudublin.ie](mailto:yvonne.desmond@tudublin.ie), [arrow.admin@tudublin.ie](mailto:arrow.admin@tudublin.ie), [brian.widdis@tudublin.ie](mailto:brian.widdis@tudublin.ie).



This work is licensed under a [Creative Commons Attribution-Noncommercial-Share Alike 3.0 License](#)

# **An Exploration of the Use of Gamification in Agile Software Development**

**Alan McClean**

A dissertation submitted in partial fulfilment of the requirements of  
Dublin Institute of Technology for the degree of  
M.Sc. in Computing (Knowledge Management DT217)

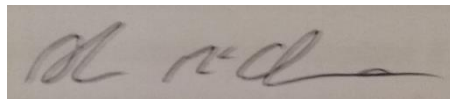
**September 2015**

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Knowledge Management), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

***Signed:***

A rectangular box containing a handwritten signature in dark ink on a light-colored background. The signature is cursive and appears to be 'D. McD.'.

***Date:***

***30 September 2015***

# 1 ABSTRACT

Although Project Management has existed for many millennia, software project management is relatively new. As a discipline, software project management is considered difficult. The reasons for this include that software development is non-deterministic; opaque and delivered under ever-increasing time pressure in a volatile environment. Evolving from Incremental and Iterative Development (IID), Agile methodologies have attempted to address these issues by focusing on frequent delivery; working closely with the customer; being responsive to change and preferring working software to extensive documentation. This focus on delivery rather than documentation has sometimes been misrepresented as no documentation, which has led to a shortfall in project metrics.

Gamification has its roots in motivation. The aim of gamification is to persuade users to behave in a manner set out by the designer of the gamification. This is achieved by adding game mechanics or elements from games into non-game applications. This dissertation examines the use of gamification in Agile projects and includes an empirical experiment that examines the use of gamification on Agile project tracking. Project tracking is an element of software engineering that acts as a de-motivator for software engineers. Software Engineers are highly motivated by independence and growth, while project tracking is seen as boring work. The dissertation experiment identifies a methodology for applying gamification experiments and then implements an experiment. The result was an overall improvement in project tracking. The experiment needs to be expanded to be run over a longer period of time and a more varied group of development teams.

**Key words:** *Software Methodologies; Project Management; Agile; Motivation; Gamification;*

## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks to my supervisor Andrew Hines, without whose guidance, I would not have been able to complete this dissertation. I would like to thank the lecturers at DIT for their support throughout the MSC programme. I would like to thank work colleagues for their assistance and willing participation, particularly during the experiment phase of the dissertation. And finally, I would like to extend my heartfelt thanks to my wife Deirdre, for support, encouragement and patience while I completed this dissertation.

# TABLE OF CONTENTS

## Contents

<b>1</b>	<b>ABSTRACT</b> .....	<b>II</b>
	<b>TABLE OF FIGURES</b> .....	<b>VIII</b>
	<b>TABLE OF TABLES</b> .....	<b>IX</b>
<b>1.</b>	<b>INTRODUCTION</b> .....	<b>1</b>
1.1	INTRODUCTION TO DISSERTATION ON GAMIFICATION IN AGILE.....	1
1.2	BACKGROUND .....	1
1.3	RESEARCH PROBLEM .....	2
1.4	INTELLECTUAL CHALLENGE .....	3
1.5	RESEARCH OBJECTIVES .....	4
1.6	RESEARCH METHODOLOGY .....	5
1.7	RESOURCES .....	6
1.8	SCOPE AND LIMITATIONS .....	7
1.9	ORGANISATION OF THE DISSERTATION .....	8
<b>2</b>	<b>PROJECT MANAGEMENT</b> .....	<b>10</b>
2.1	INTRODUCTION.....	10
2.2	HISTORY AND DEFINITIONS .....	10
2.3	TRADITIONAL SOFTWARE METHODOLOGIES .....	14
2.3.1	<i>Waterfall Methodology</i> .....	14
2.3.2	<i>V-Model</i> .....	16
2.3.3	<i>The Spiral Model</i> .....	17
2.3.4	<i>Summary</i> .....	18
2.4	DIFFICULTIES IN SOFTWARE PROJECT MANAGEMENT .....	19
2.5	CONCLUSION .....	22
<b>3</b>	<b>AGILE</b> .....	<b>23</b>
3.1	INTRODUCTION.....	23
3.2	AGILE OVERVIEW.....	23
3.3	HISTORY OF AGILE.....	26

3.4	KEY METHODOLOGIES .....	27
3.4.1	<i>Version One survey</i> .....	28
3.4.2	<i>Scrum</i> .....	29
3.4.3	<i>Extreme Programming</i> .....	31
3.4.4	<i>Kanban</i> .....	35
3.4.5	<i>Scrumban</i> .....	36
3.5	AGILE VERSUS OTHER SOFTWARE DEVELOPMENT METHODOLOGIES.....	38
3.5.1	<i>Comparison</i> .....	38
3.5.2	<i>Issues</i> .....	39
3.6	STUDIES IN AGILE .....	41
3.6.1	<i>Migrating to an Agile methodology</i> .....	41
3.6.2	<i>Estimation in Agile</i> .....	42
3.6.3	<i>Agile and global software development</i> .....	43
3.7	CONCLUSION .....	43
<b>4</b>	<b>MOTIVATION OF SOFTWARE ENGINEERS .....</b>	<b>45</b>
4.1	INTRODUCTION.....	45
4.2	OVERVIEW OF MOTIVATION .....	45
4.3	SELF-DETERMINATION THEORY .....	48
4.4	PERSUASIONS MODELS.....	49
4.5	MOTIVATING SOFTWARE ENGINEERS.....	50
4.6	CONCLUSION .....	53
<b>5</b>	<b>GAMIFICATION .....</b>	<b>54</b>
5.1	INTRODUCTION.....	54
5.2	APPROACH .....	54
5.2.1	<i>Approach</i> .....	54
5.2.2	<i>Results</i> .....	55
5.3	OVERVIEW .....	56
5.3.1	<i>Definitions of Gamification</i> .....	57
5.3.2	<i>Situating Gamification</i> .....	62
5.3.3	<i>Issues with gamification</i> .....	64
5.4	GAME ELEMENTS .....	65
5.5	EXISTING PAPERS .....	69

5.3.4	<i>Education</i> .....	69
5.3.5	<i>IT</i> .....	71
5.6	GAMIFICATION AND AGILE.....	72
5.7	CONCLUSION .....	75
<b>6</b>	<b>EXPERIMENT OVERVIEW .....</b>	<b>76</b>
<b>7</b>	<b>HISTORICAL ITERATIONS.....</b>	<b>78</b>
7.1	INTRODUCTION.....	78
7.2	EXPERIMENTATION.....	78
6.2.1	<i>Identifying Projects</i> .....	78
6.2.2	<i>Metrics to capture</i> .....	79
7.3	EVALUATION.....	90
7.3.1	<i>Projects</i> .....	91
7.3.2	<i>Data Filtered</i> .....	98
7.3.3	<i>Metrics Results</i> .....	100
7.3.4	<i>Context data</i> .....	103
7.3.5	<i>Discussion</i> .....	103
7.4	CONCLUSION .....	106
<b>8</b>	<b>MONITORED ITERATION .....</b>	<b>107</b>
8.1	INTRODUCTION.....	107
8.2	EXPERIMENTATION.....	107
8.2.1	<i>Methodology</i> .....	107
8.2.2	<i>Interview with the Scrum Master</i> .....	109
8.2.3	<i>Project Selection</i> .....	109
8.3	EVALUATION.....	110
8.3.1	<i>Project details</i> .....	110
8.3.2	<i>Metrics</i> .....	110
8.3.3	<i>Discussion</i> .....	111
8.4	CONCLUSION .....	113
<b>9</b>	<b>GAME ITERATION .....</b>	<b>114</b>
9.1	INTRODUCTION.....	114
9.2	EXPERIMENTATION.....	114
9.2.1	<i>Aim</i> .....	114



9.2.2	<i>Game</i> .....	115
9.2.3	<i>Other Game Ideas</i> .....	117
9.2.4	<i>Methodology</i> .....	117
9.3	EVALUATION.....	124
9.3.1	<i>Survey results</i> .....	124
9.3.2	<i>Metrics</i> .....	126
9.3.3	<i>Interview results</i> .....	127
9.3.4	<i>Discussion</i> .....	130
<b>10</b>	<b>CONCLUSION</b> .....	<b>133</b>
10.1	INTRODUCTION.....	133
10.2	RESEARCH DEFINITION & RESEARCH OVERVIEW.....	133
10.3	CONTRIBUTIONS TO THE BODY OF KNOWLEDGE.....	135
10.3.1	<i>Contributions to the organization</i> .....	135
10.3.2	<i>Contributions to Academia</i> .....	136
10.4	EXPERIMENTATION, EVALUATION AND LIMITATION.....	138
10.4.1	<i>Experimentation Overview</i> .....	138
10.4.2	<i>Evaluation</i> .....	140
10.4.3	<i>Limitations</i> .....	141
10.5	FUTURE WORK & RESEARCH.....	143
10.6	CONCLUSION.....	144
	<b>GLOSSARY</b> .....	<b>145</b>
	<b>BIBLIOGRAPHY</b> .....	<b>146</b>
	<b>APPENDIX A</b> .....	<b>154</b>
	<b>APPENDIX B</b> .....	<b>157</b>
	<b>APPENDIX C</b> .....	<b>161</b>
	<b>APPENDIX D</b> .....	<b>165</b>

## TABLE OF FIGURES

FIGURE 1: PROJECT PYRAMID. REPRODUCED FROM CONSTRUCTION.COM (2015).....	12
FIGURE 2: THE CLASSIC WATERFALL METHODOLOGY. EACH PHASE OF THE WATERFALL METHODOLOGY FEEDS INTO THE NEXT PHASE, AND MUST BE COMPLETE BEFORE MOVING ONTO THE PHASE. (ROYCE,W.W, 1970) .....	14
FIGURE 3: THIS FIGURE SHOWS THE V-MODEL. (RUPARELIA, NAVAN B, 2010) .....	16
FIGURE 4: THE SPIRAL MODEL: THE DEVELOPMENT MOVES FROM THE CENTRE OUT AND PRODUCES A PROTOTYPE AT THE END OF EACH CYCLE. (BOEHM, B, 1988) .....	17
FIGURE 5: SHOWS THE MAIN ELEMENTS OF SCRUM. THE SPRINT EXECUTION IS A TIME BOXED PERIOD IN WHICH THE TEAM MEETS DAILY TO DISCUSS THE PROGRESS OF THE WORK TAKEN ON IN THAT PERIOD. THE OUTPUT OF A SPRINT IS WORKING SOFTWARE COMPONENTS .....	29
FIGURE 6: THIS SHOWS THE PLANNING LOOP FOR XP PROJECTS. (BECK K, 1999).....	34
FIGURE 7: MASLOW'S HIERARCHY REPRODUCED FROM SIMPLYPSYCHOLOGY, (2015) ..	45
FIGURE 8: SELF-DETERMINATION THEORY REPRODUCED FROM GAGNÉ, M. & DECI, E.L. (2005).....	48
FIGURE 9: SHOWS WHERE GAMIFICATION IS PLACED IN RELATION TO OTHER SUBJECT AREAS (DETERDING S, 2013). .....	62
FIGURE 10: LUDIFICATION OF CULTURE (DETERDING S, 2013) .....	63
FIGURE 11: EXPERIMENT INFOGRAPHIC .....	76
FIGURE 12: ESTIMATES VERSUS ACTUALS SAMPLE.....	85
FIGURE 13: VELOCITY SAMPLE .....	86
FIGURE 14: REWORK SAMPLE .....	86
FIGURE 15: ITERATION BURNDOWN SAMPLE .....	87
FIGURE 16: ITERATION BURNDOWN ACROSS PROJECTS SAMPLE .....	88
FIGURE 17: SHOWS THE ACTUAL VERSUS ESTIMATE; VELOCITY; REWORK AND ITERATION BURNDOWN FOR PROJECTS A, B AND C.....	100
FIGURE 18: THE ESTIMATES VERSUS ACTUALS; VELOCITY; REWORK AND ITERATION BURNDOWN FOR PROJECTS D, E AND F; .....	101
FIGURE 19: COMPARISON OF CURRENT PROJECT (G) WITH PREVIOUS PROJECT (F) .....	126

## TABLE OF TABLES

TABLE 1: THE ADVANTAGES AND DISADVANTAGES OF THE INDIVIDUAL TRADITIONAL SOFTWARE METHODOLOGIES.....	19
TABLE 2: REASONS FOR PROJECT CANCELLATION (EL EMAM, K. 2008).....	21
TABLE 3: SHOWS THE USE OF AGILE WITHIN THE RESPONDENT’S ORGANIZATION. WHILE AGILE IS BEING ADOPTED ACROSS ORGANIZATIONS, THE MAJORITY OF PROJECT TEAMS ARE NOT AGILE.....	28
TABLE 4: SHOWS THE TOP 5 METHODOLOGIES USED BY THE RESPONDENTS. THE OTHERS WERE USED BY 4% OR LESS OF THE RESPONDENTS THE MOST POPULAR METHODOLOGY IS SCRUM WITH HYBRIDS OF SCRUM POPULAR TOO.....	28
TABLE 5: THE CHIEF COMPONENTS OF THE SCRUM METHODOLOGY. ....	30
TABLE 6: THIS TABLE SHOWS THE XP PRACTICES. (BECK K, 1999).....	33
TABLE 7: THIS TABLE OUTLINES SOME OF THE PRACTICES AND THE ADVANTAGES THAT THEY HAVE. THE FOCUS ON THIS IS PROVIDED BY THE BENEFITS OF PAIR PROGRAMMING. (COCKBURN A AND WILLIAMS L, 2000).....	35
TABLE 8: THIS OUTLINES THE PRINCIPLES OF KANBAN.....	36
TABLE 9: THIS TABLE OUTLINES THE PRINCIPLES OF SCRUMBAN (YERET, Y, 2015).....	37
TABLE 10: A COMPARISON OF AGILE AND TRADITIONAL SOFTWARE METHODOLOGIES (AWAD, MA, 2005).....	39
TABLE 11: AGILE RESOLVES THE ISSUES OF TRADITIONAL MODELS (EL EMAM, K. 2008).....	41
TABLE 12: SHOWS THE MOTIVATION FACTORS ASSOCIATED WITH SOFTWARE ENGINEERS. SHARP, H <i>ET AL.</i> , (2009).....	51
TABLE 13: DE-MOTIVATORS ASSOCIATED WITH SOFTWARE ENGINEERS. SHARP, H <i>ET AL.</i> , (2009).....	52
TABLE 14: THIS TABLE SHOWS A BREAKDOWN OF THE PAPERS BASED ON THE INITIAL SEARCH TERMS.....	55
TABLE 15: THIS TABLE SHOWS A BREAKDOWN OF THE PAPERS BY SUBJECT AREA. THE SUBJECT AREA WAS CHOSEN BASED PRIMARILY ON THE TITLE AND THE JOURNAL THAT THE PAPER APPEARED IN. ....	56
TABLE 16: THIS TABLE SHOWS A BREAKDOWN OF PAPERS WHICH WERE CONSIDERED OVERVIEW FOLLOWING THE REVIEW OF ABSTRACT.....	56

TABLE 17: SHOWS SOME OF THE MAIN POINTS AGAINST GAMIFICATION. FOR COMPLETENESS I HAVE ADDED A REBUTTAL, WHICH IS THE AUTHOR’S OWN OPINION. ....	65
TABLE 18: THIS TABLE SHOWS THE FRAMEWORK FOR GAMES. IT WILL BE USED WHEN DEFINING THE GAMIFICATION EXPERIMENT (AVEDON EM, 1981).....	66
TABLE 19: LEVELS OF ABSTRACTION IN THE USE OF GAME ELEMENTS IN GAMIFICATION. THIS IS USED IN THE DESCRIBING THE EXPERIMENT (DETERDING, S, 2011).....	67
TABLE 20: A SELECTION OF GAME ELEMENTS OR MECHANISIMS. SOME ARE TAKEN FROM REEVES AND READ’S “TEN INGREDIENTS OF GREAT GAMES” (REEVES D AND READ TJ, 2013) WHILE OTHERS ARE TAKEN FROM GAMIFICATION.ORG WEBSITE.....	69
TABLE 21: THE FRAMEWORK FOR GAMES AS APPLIED TO PLANNING POKER (AVEDON, 1981) .....	74
TABLE 22: CHARACTERISTICS OF THE MAJOR PROJECTS .....	91
TABLE 23: THIS SHOWS THE SCRUM ELEMENT, TOGETHER WITH ITS EQUIVALENT FROM STATISTICS PROJECT. A DESCRIPTION OF THE COMPONENT IS GIVEN WHICH FOCUSES ON DESCRIBING THE ACTIVITY IN THE PROJECT. ....	95
TABLE 24: THIS SHOWS THE SCRUM ELEMENT, TOGETHER WITH ITS EQUIVALENT FROM PROJECT. A DESCRIPTION OF THE COMPONENT IS GIVEN WHICH FOCUSES ON DESCRIBING THE ACTIVITY IN THE PROJECT.....	97
TABLE 25: SUMMARY OF DATA FILTERED, SHOWING UNASIGNED; MISSING ESTIMATES AND ACTUALS .....	99
TABLE 26: SUMMARY VIEW OF THE PROJECT DATA. THE ACTUAL THAT IS USABLE FOR THE CHARTING AND COMPARISON IS SIGNIFICANTLY REDUCED FROM THE ORIGINAL DATA.....	99
TABLE 27: SUMMARIZED CONTEXT DATA FOR THE HISTORICAL ITERATIONS. THE MEAN AND (STDDEV) ARE SHOWN .....	103
TABLE 28: POOR QUALITY DEFECT TRACKING .....	105
TABLE 29: SHOWS THE ADDITIONAL METRICS CAPTURED AIN THIS ITERATION.....	109
TABLE 30: THE METRIC RESULTS CAPTURED FOR THE NEW ITERATION.....	111
TABLE 31: DESCRIBES THE LOTTERY GAME USES THE GAMES FRAMEWORK .....	116
TABLE 32: MOTIVATIONAL FACTORS FOR SOFTWARE ENGINEERS (SHARP, H., <i>ET AL.</i> , 2009) .....	119
TABLE 33: SHOWS THE OPTIONS PRESENTED TO THE PROJECT MANAGER AS SUGGESTED REWARDS .....	120

TABLE 34: FOR EACH OF THE ITEMS IN THE LIST OF REWARDS WE HAVE IDENTIFIED THE MOTIVATIONAL FACTOR ASSOCIATED WITH THIS REWARD. ....	121
TABLE 35: ITEMS SELECTED FOR PRESENTATION TO THE TEAM. THESE ITEMS WILL BE PASSED AS A SURVEY TO THE TEAM FOR SELECTION.....	122
TABLE 36: THE SURVEY RESULTS FROM THE TEAM.....	125
TABLE 37: THE WEIGHTED AVERAGES SHOW THAT THE EXTRA TRAINING WAS THE MOST POPULAR SELECTION .....	125
TABLE 38: THE METRICS FROM THE GAMIFIED ITERATION .....	127
TABLE 39: SAMPLE INTERVIEW RESULTS.....	129
TABLE 40: CAPACITY VERSUS ACTUALS COMPARISONS .....	131
TABLE 41: THE NUMBER OF USERS WHO HAVE CAPACITY IN THE PROJECT. PROJECT A AND B DID NOT SET CAPACITY FOR THE USERS. ....	161
TABLE 42: THE NUMBER OF USERS WHO HAD ESTIMATES IN THE PROJECT .....	162
TABLE 43: THE NUMBER OF USERS WHO HAD ACTUALS IN THE PROJECT.....	162
TABLE 44: THE TIME DIFFERENCE BETWEEN THE TECHNICAL LEADERSHIP AND THE DEVELOPMENT TEAM .....	163
TABLE 45: THE PUBLIC HOLIDAYS IN THE DEVELOPMENT SITE .....	163
TABLE 46: PUBLIC HOLIDAYS IN THE TECHNICAL LEADERSHIP SITE .....	164
TABLE 47: SHOWS THE NUMBER OF STORIES IN EACH ITERATION FOR EACH PROJECT ANALYSED. THE TABLE ALSO INCLUDES THE COUNT OF STORIES NOT ASSIGNED AN ITERATION.....	165
TABLE 48: SHOWS THE NUMBER OF DEFECTS IN EACH ITERATION FOR EACH PROJECT ANALYSED. THE TABLE ALSO INCLUDES THE COUNT OF DEFECTS NOT ASSIGNED AN ITERATION.....	166
TABLE 49: SHOWS THE NUMBER OF STORIES AND DEFECT THAT DID NOT HAVE ESTIMATES ASSIGNED. THESE ARE FILTERED OUT AS THEY CANNOT BE USED IN THE METRICS.....	166
TABLE 50: SHOWS THE NUMBER OF STORIES AND DEFECTS WITH MISSING ACTUALS ...	167
TABLE 51: SHOWS THE COMBINED NUMBER OF STORIES AND DEFECTS USED TO PRODUCE THE CHARTS. ....	167

# 1. INTRODUCTION

## *1.1 Introduction to dissertation on gamification in Agile*

This section of the dissertation introduces the research and experiment used to evaluate gamification in Agile Software Development. The section starts with a background review and then identifies the research problem that is being evaluated. The next section identifies the challenges encountered during the dissertation. The next sections examine the objectives, methodologies and resources used in the dissertation. The scope and limitations are then outlined. Finally, the last section details the organization of the remainder of the dissertation.

## *1.2 Background*

Historically, the management of software projects has proven challenging. Managing software development is different to other types of project. Software development is non-deterministic and opaque (Hall, N.G., 2012). Traditional software methodologies had largely ignored these issues and developed with a focus on up-front analysis and verification at the end. The customer did not see the product until it was complete.

Agile methodologies were designed to resolve many of the issues in traditional methodologies. They focus on frequent delivery, less documentation and acceptance of change (Fowler, M, 2001). Agile has become a key set of methodologies in software development. 95% of organizations now use some form of Agile (Version One, 2015).

Software development is a knowledge intensive business. The knowledge is held in the team members, whether that is the Business System Analyst who elicits and codifies the requirements, the architect who ensures the technical viability of the software, or the development team members who have deep knowledge of the code. Software development methodologies are attempts to capture some of this knowledge for sharing in the current project and reuse in future projects (Rus, I and Lindvall, M, 2002). Traditional project methodologies attempt to record minute levels of details, but

suffer from the challenge of maintaining this as requirements change. Agile project methodologies prefer to document less of the knowledge, but intrust the knowledge to members of the team. They assume that the knowledge is shared amongst the team through frequent meetings and retrospectives.

Initial definitions of gamification focused on the mechanics of games. The aim was to use game elements in a non-gaming context to motivate users to do something that they might not do otherwise (Deterding, S, 2011). An industry has sprung up around this definition. The definition of gamification has evolved and now aligns with motivation theories. These state that motivation can be intrinsic or extrinsic, with intrinsic motivation being recognized as having more influence over a person's behaviour. It has been found that some intrinsic motivators have a negative impact on the person's intrinsic motivators.

This section has highlighted that software development is a knowledge management process and that traditional methodologies and Agile both suffer issues related to the management of this knowledge. The section also examines gamification in the context of motivation. The next section examines the research problem.

### ***1.3 Research problem***

This section introduces the research problem which relates to the Agile software development. The question was:

RQ1: Can gamification be used in a manner that has a positive impact on an Agile project? This can be decomposed into a number of sub-questions. Can gamification be used to improve the tracking of an Agile project? Can gamification be used to improve the efficiency of the team? Can gamification be used to have an impact on the motivation of the team?

The reason for tackling this problem was the author's experience of Agile development with teams in the organization. Although Agile was in use in the organization, it was not clear that it was working. One principle of Agile highlighted the need to "build projects around motivated individuals", (Fowler, M, 2001). The teams did not appear

to be highly motivated, so the dissertation set out to establish if gamification could be used to motivate the teams to improve the success of the project.

This section has described the research problem. The next section describes the intellectual challenges associated with the dissertation.

### ***1.4 Intellectual challenge***

The main challenges of the dissertation were as follows:

- Access to valid data: This issue only arose during the early analysis of the project. Although it was anticipated that there would be some issues with the quality of the data, it was not anticipated that the data would be unusable. The data quality resulted in a change to the experiment. A phase was introduced to improve the quality and the scope of the experiment, which was modified to focus on a single project team;
- Time Pressure: The limitation of completing the dissertation within a set period, combined with the fixed iteration dates in the experiment projects, added an element of pressure to the dissertation. This was added to by the change in the experiment phases, resulting in the gamification experiment only running over a single iteration;
- Technologies: The main technological difficulty was use of an add-on tool to extract the project data. Although the tool provided a means of extracting the data through a query interface it was not clear how to use the technology in the most efficient manner. There was limited documentation available on the extract tool;
- Research topics: The research areas of motivation and gamification were unfamiliar to the author. Other areas had previously been covered by course modules and work experience. Gaining an understanding of these areas was difficult. In addition to this, the gamification body of knowledge was relatively new and many of the key papers only recently been added;
- Writing a dissertation: This was more challenging than anticipated. When conducting the literature review, the most difficult aspects was determining which aspects were relevant to the thesis. As a result, the first drafts were more verbose than necessary. Establishing an experiment was also more complex. A



lot of consideration was given to the team being able to manipulate the outcome, however the team did not show any inclination to do this. Having no experience of writing this volume of data, most activities were underestimated, resulting in overtime to complete the dissertation.

This section of the document has outlined the main intellectual challenges faced during the compilation of this dissertation. The next section examines the research objectives.

### ***1.5 Research objectives***

The following objectives have been achieved throughout the dissertation and contributed to the overall outcome:

- Review literature about Project Management; Agile methodologies; motivation of software engineers and gamification. This has contributed to the overall dissertation by providing an understanding of the key issues that relate to the experiment. It was used in the project design, particularly in the understanding of the metrics to measure the historical projects and the selection of the game elements. It was also used in the evaluation of the results of the experiment;
- Review previous literature using gamification in software development. This was used to assist in the design of the experiment;
- Review previous literature on Agile measurement and planning. This was used to establish metrics which were tracked as part of the experiment;
- Design the interviews regarding the team development process and the issues in the team. The interviews were used in establishing the issues in the projects and to retrieve an explanation as to the quality of the data. These interviews were conducted with the project Scrum Masters;
- Establish a means of improving the data quality. This was not part of the original research objectives. However, the quality of the data resulted in the need to establish a means of improving the data. The result was the introduction of external regulation, (Deci, E.L. & Ryan, R.M., 2012), to motivate the team to update the tracking tool. The subsequent phase of the experiment improved the data quality significantly. However, it was arduous to implement and not popular with the team members;

- Design the game. This involved selecting game elements for use with the project and establishing a set of rules of play. The necessary managerial approval for a reward was then retrieved. A selection of potential rewards was put to the team in a survey.
- Design post experiment interview to gather qualitative results of experiment;
- Conduct gamification experiment. The process ran over a fifteen day iteration;
- Gather quantitative results from the experiment and conduct interviews;
- Evaluate the results and write up the thesis experiment;

This completes the section regarding the research objectives of the dissertation. The next section reviews the research methodology.

## ***1.6 Research methodology***

This section of the document describes the methodology used in the research of the dissertation. The dissertation was composed of two separate parts; a literature review and an experiment.

The literature review was based on the secondary research category of review. A systematic review of the literature was conducted using search of online reference libraries. The search focused on searching for key words related to the topic. The key authors and papers were identified, based on the number of citations and the use in other papers. Initially, the search was to find overview papers, but specific sub-topics were then researched. The process taken for the gamification subject area is highlighted in that chapter. This extra step was taken because the available material was evolving. As a result, the results of the search were documented.

The second part of the dissertation was based on empirical research. The aim of this research was to provide evidence that gamification could be used to influence the activities of an Agile process. The experiment used an objective mixed method research. The research combined qualitative results retrieved through interviews with quantitative results retrieved from the project tracking system. The research focused on a specific project, with the intention to induce general results from the project results. It cannot be asserted that the results of the gamification experiment are repeatable. In addition it is not possible to extrapolate the outcome when running the gamification experimentation over a longer period of time. However, the methodology used is

repeatable and while running the experiment for a longer period of time was beyond the scope of this dissertation it would be possible to conduct this in the future.

This section has described the research methodology used in the dissertation. The next section of the document describes the resources used in the dissertation.

## ***1.7 Resources***

This section describes the resources in use in completing the dissertation.

**Financial Organization:** The organization in which the experiment was run. This included access to the teams which participated in the project and the data in the progress tracking tool.

**Rally Tool and Rally Tool add-in:** The data relating to project tracking are stored in a cloud computing based tool called Rally. The Rally Tool add-in was used to extract data from Rally to Microsoft Excel. To do this, the add-in had to be installed as an Excel Add-in. The tool was used directly to extract additional data manually to supplement the data extracted from using the add-in. (Rally, 2015)

**Microsoft Office products:** The dissertation was completed in Microsoft Word. The data was analysed in Microsoft Excel.

The following reference libraries were included in the search for conference and journal papers;

- ACM: Association for Computing Machinery;
- IEEE: Institute of Electrical and Electronics Engineers;
- Gartner: Gartner for Technical professionals available through organization account;
- Forrester: For Information Technology, Marketing and Strategy and Technology Industry. This is available through organization account;
- Google Scholar: Used to supplement the papers when not found in ACM or IEEE libraries.

This section has described the resources in use in the dissertation, the next section describes the scope and limitations of the dissertation.

### ***1.8 Scope and limitations***

This section describes the scope of the dissertation and then describes the limitations of the experiment.

The project evaluates the use of gamification within Agile Software Project Development. The following subject areas are therefore within the scope of the dissertation;

- **Gamification:** The dissertation must provide an understanding of gamification. The dissertation includes a literature review of the gamification field. As part of the review of gamification, a more general review of motivation was conducted. In addition to this a review of motivation in software engineers was also referenced. Specifically, gamification is defined and discussed, game elements are identified and existing projects are discussed. A gamification experiment is created and evaluated for its impact on Agile software project development;
- **Agile Software Project Development:** The dissertation provides an understanding of Agile. The dissertation extends this to include project management. This is necessary as Agile methodologies build on project management and understanding Agile is difficult without an understanding of traditional project management. The project management review describes the history of project management and the key methodologies. The issues with traditional project management are outlined. The Agile literature review describes the Agile manifesto and the history of Agile. The most widely used Agile methodologies are then described and Agile is compared to traditional methodologies. Finally, existing studies in Agile which are relevant to the thesis are discussed. The experiment focused on two Agile project teams. As part of the experiment the project tracking data was analyzed. The experiment then focused on a single team and reviewed the impact of monitoring and gamification on their Agile processes.

The initial phase of the project focused on data capture. As part of the analysis of that data it was hoped to find a trend in the data which could be influenced by applying a

gamification technique. However, the analysis of the data concluded that the data was not complete and not usable for determining the impact of gamification. The outcome of this was that the next phase of the experiment required the addition of monitoring of the team updates. As a result of this the motivation section of the literature review focused on the impact of external regulation (Deci, E.L. & Ryan, R.M., 2012).

There were a number of limitations for the experiment. Firstly, the experiment was limited to two data warehouse projects and gamification iteration focused on only one. It was necessary to limit the experiment to comparable projects, so that the metrics from the project could be compared. However, it is not clear that the characteristics of these projects had an impact on the experiment. A second limitation of the experiment was that it was time boxed. The monitoring aspect ran in isolation for one iteration. The gamification iteration only ran for the next iteration. It would have been preferable to run the experiment for more iterations. The experiment was not run in isolation. The experiment was run on a live project. This improves the experiment by making the results more realistic, however “keeping control is a challenge when realism is increased”. (Sjøberg, D.I. *et al.*, 2002 )

This section of the document discussed the scope and limitation of the dissertation. The next section describes the organization of the remainder of the dissertation.

### ***1.9 Organisation of the dissertation***

The dissertation is composed of a number of chapters. Following on from this introduction chapter, the second chapter of the dissertation relates to project management. The chapter provides definitions and a brief history of project management. It then describes examples of traditional software methodologies. The chapter outlines the difficulty in software project management.

The third chapter describes Agile software development. The chapter gives an overview of Agile and its history. The chapter then describes the key Agile methodologies. The next section of the chapter compares Agile with the traditional methodologies described in the project management chapter. It describes where Agile resolves the issues of traditional methodologies. The chapter then provides a brief review of existing studies in Agile, specifically those areas that relate to the thesis.

The next chapter describes motivation in software engineering. The chapter examines motivation in general and then reviews Self Determination Theory (SDT) and persuasion models. Finally, the chapter describes the factors that motivate software engineers.

The fifth chapter discusses gamification. There is a brief description of the approach to establishing gamification related papers to use. The chapter then discusses definitions of gamification. The chapter outlines some game elements that could be used in the experiment. A brief overview of existing papers on gamification is included. Finally, gamification and Agile are reviewed. This section concludes the literature review.

The sixth chapter provides a brief description of the experiment. The chapter's purpose is to provide an overview of the entire experiment.

The next chapter describes the historical data that was captured as part of the experiment. The chapter describes the methodology used to capture and analyse the data. The chapter then describes the projects selected for the experiment. The results of this phase of the experiment are then presented and discussed.

The eighth chapter describes the approach taken to monitor the project. As the selected projects were narrowed to a single project, the chapter examines the reasons for the selection. The chapter presents the results of the analysis and discusses the results.

The ninth chapter relates to the gamification experiment. It describes the game and the methodology used to perform the experiment. The chapter displays the results of that process and provides an evaluation.

The final chapter of the project provides the conclusion. The chapter restates the research questions and evaluates whether it has been proven. The chapter then examines the contribution to the body of knowledge for the organization and in the academic space. The chapter then evaluates the entire experiment and discusses the limitations. Finally, the chapter describes future work and research which relates to the dissertation.

This section concludes the introduction. The next chapter introduces the first literature review topic, project management.

## **2 PROJECT MANAGEMENT**

### ***2.1 Introduction***

This section provides an introduction to software project management. The purpose of the section is to give the user an introduction to the field and to provide an overview of the topic, major developments, issues and current thinking.

### ***2.2 History and Definitions***

Project Management has existed for a long time. As a process it is believed to have been applied to construction and engineering projects for millennia. For example, records from the construction of the Egyptian pyramids, completed almost 5000 years ago, show that there were managers responsible the completion of each of the different sides. The industrial revolution would have brought project management into business, as it would have been required to manage the necessary systems of transportation, storage, manufacturing, assembly and distribution. Similarly the scale of the two world wars furthered the use of the project management. The 1918 logistical operation supplying the British Expeditionary Force was the largest the world had ever seen. New disciplines, such as human resources and marketing emerged. The forms of project management used today in the business world emerged in the 20<sup>th</sup> century specifically around the period of the Second World War. At this point there was a need to organize vast quantities of resources and personnel to achieve critical objectives in specific timeframes. Post the Second World War, project management developed into a mainstream activity in business, culminating in the creation of standards and standards bodies such as PRINCEII and the project management institute (PMI).

In contrast, project management for software development is relatively new. Royce,W.W, (1970) described the analysis and coding as the “two essential steps common to all computer program development”. He then defined the more complex Waterfall methodology which added requirements, design, testing and operation.

Boehm, B, (1989), stated that software project management was an art form. “The skillful integration of software technology, economics and human relations in the specific context of a software project is not an easy task”. Boehm defined the

manager's problem as the need to simultaneously satisfy many diverse interested parties, including customers, management, developers and production support. Since Boehm's paper, software project management has grown significantly. "The growth is largely attributable to the emergence of many new diverse business applications that can be successfully managed as projects". (Hall, N.G., 2012)

Today, project management is well documented. There are many definitions but perhaps the most prominent come from the Project Management Institute, in particular in the Project Management Book of Knowledge (PMBOK, 5th Edition). In PMBOK, they say you must firstly define a project in order to define project management. PMBOK states a project is defined as:

Any series of activities, with a specific objective, that has start and end dates. It may have a fixed budget, utilize people, time and equipment, and may utilize multiple functional areas. (PMBOK, 5th Edition)

For software development, this includes most projects but excludes on-going maintenance. From this definition the PMBOK describes project management as having five process groups:

- **Project Initiation:** This stage determines which project should be tackled given the available resources. The project benefits and costs are identified and are used to determine if the project will get sanctioned. The project manager is assigned;
- **Project Planning:** This is where the project requirements are defined. The resources needed are determined, as well as the quality and quantity of the deliverables. Risks are evaluated and a project schedule is determined;
- **Project Execution:** This is the build phase of the project. The team members are assigned to the work. The focus at this stage is to ensure that the team members have what they need to complete the project. For example, environment and training;
- **Project Monitoring and Control:** This relates to the processes required to maintain project schedules and budgets as issues and risks materialize;
- **Project Closure:** This is the process of closing down the project. This process involves the administrative tasks to close the projects, verifying that all the

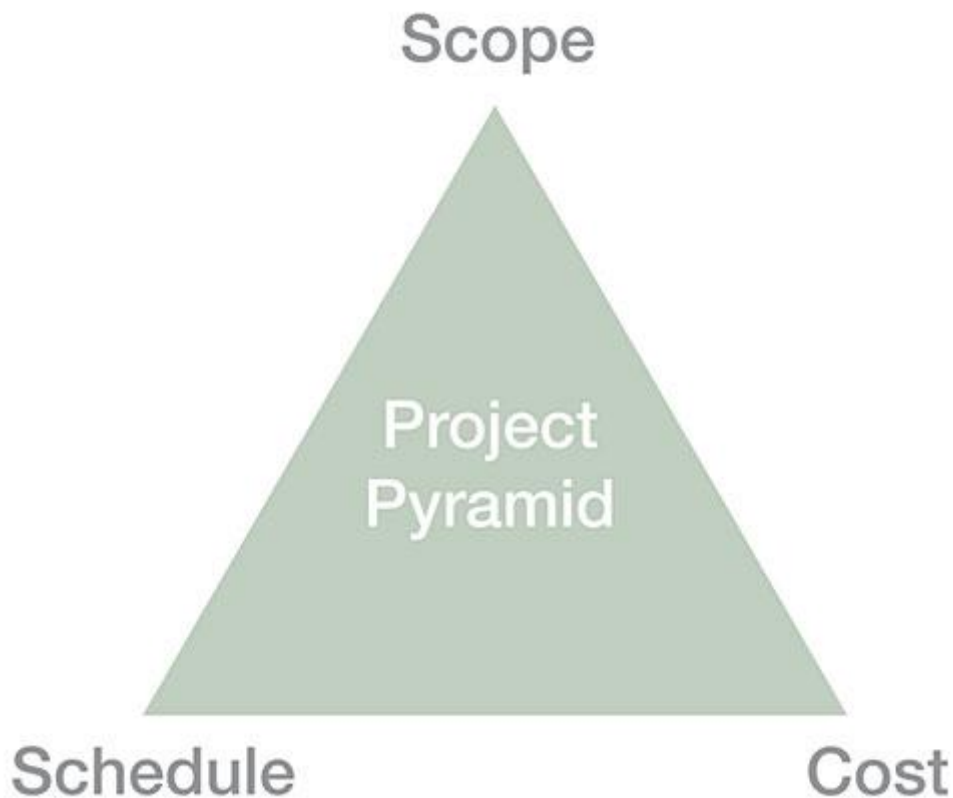


work that was part of the project has been accomplished including the changes to requirements that were encountered and committed to during the project. Project contract documents are completed and the project financials are closed.

Based on these processes, “Project Management is the planning, organizing, directing and controlling of a company’s resources for a relatively short-term objective that has been established to complete specific goals and objectives”. (Kerzner, H, 2013)

The key message here is that project management is focused on a project and works across the multiple functional areas and at different management levels.

Project success can be defined with respect to project constraints.



**Figure 1: Project pyramid. Reproduced from Construction.com (2015)**

As the figure shows the main constraints of the project are:

- Scope: The deliverables that the project team must create and the activities required to create them. Scope also includes the quality of the work or deliverables that need to be created;
- Cost: The budget or cost to deliver the project;

- Schedule: The deadline by which the project must be delivered.

For every project, the project manager needs to understand which of these can be compromised when delivering the project. If the cost and schedule are fixed then some of the scope will have to be dropped. If the scope is fixed, then the cost will have to be flexible.

In traditional software projects, cost and scheduling are based on estimates which are calculated upfront as part of the project planning phase. A number of different models exist:

- Expert judgement: This is not a formal model. In this instance the estimate is based on the judgement of the expert. The expert uses their experience in previous projects to provide estimates for the next project. If the expert's experience is not relevant to the actual work or the project is significantly different from previous projects, then the expert's estimates will not reflect the actual project;
- Least-squares linear regression: This uses the number of elements that the estimator believes to be important to the project. This will include the number of files, web pages, tables. This is then passed into a formula to produce an estimate;
- Case Based Reasoning: The approach here is to look for similar projects, based on the number of files, interfaces, web pages and table. The effort from these projects is then applied to the project being estimated;
- Wideband Delphi: This is an extension of the Delphi estimation technique, which uses more team members, not just experts and is conducted in a series of meetings. The approach is to involve the team; by first outlining the problem to be estimated and agreeing the unit of estimates in a kick-off meeting. The individual team members then prepare an initial list of tasks and efforts against those tasks. There is then an estimation meeting in which the total project estimates are shown anonymously to the team. Each participant reads out their task list, which should result in a larger set of tasks and assumptions are discussed. The participants then revise their estimates. This continues for four

rounds, unless the estimates have converged to an acceptable range. (Weigers K, 2000)

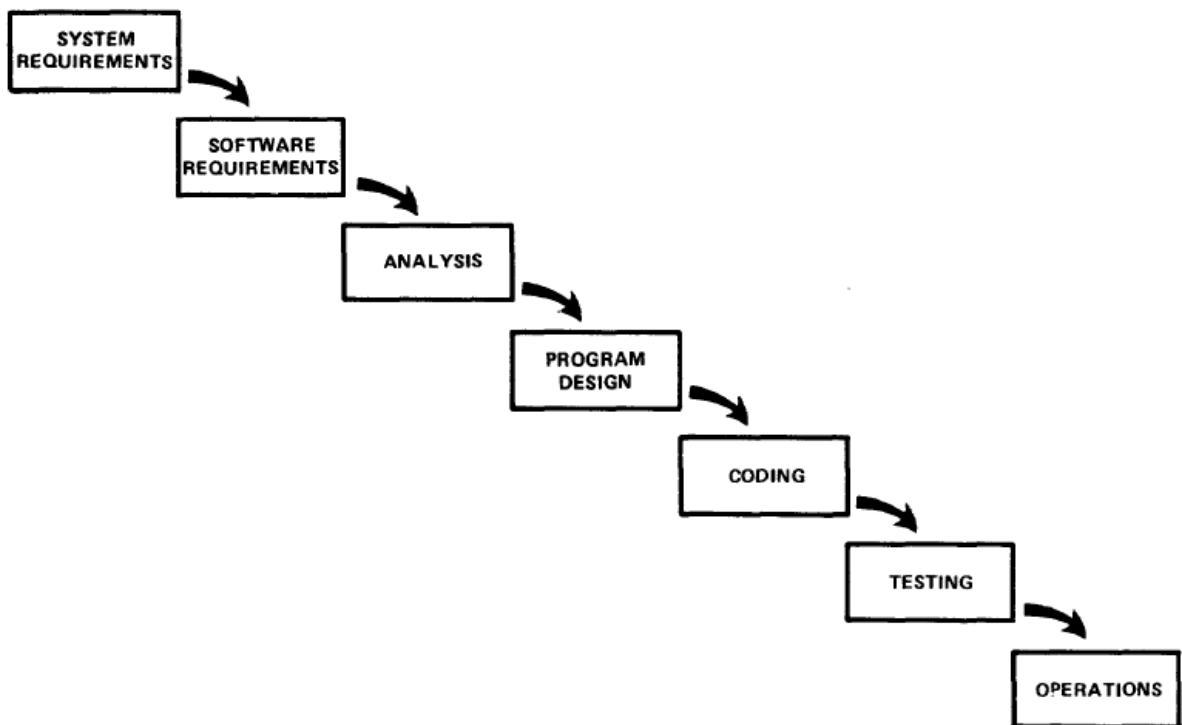
Despite of the existence of the many formal methodologies it would appear that expert judgement is still in use in project management.

In the next section we look at some of the traditional software methodologies used by project managers to manage projects.

### ***2.3 Traditional Software Methodologies***

This section of the document gives a brief description of traditional software development methodologies. These are then compared with Agile methodologies.

#### **2.3.1 Waterfall Methodology**



**Figure 2: The classic waterfall methodology. Each phase of the waterfall methodology feeds into the next phase, and must be complete before moving onto the phase. (Royce,W.W, 1970)**

System Requirements is the gathering of the requirements, or functionality the system should provide. Software requirements define how the system should look and perform. Analysis is the effort to understand these requirements. Program Design

defines how the system will be implemented. Coding is the work to implement the code for the system. Testing is the phase in which the completed system is tested to ensure that it meets the requirement. Finally, Operations is the deployment of the system and the maintenance tasks required to keep it available.

The methodology was first proposed by Winston Royce in 1970. The proposal was made in response to the general expectation that software should be a two-step process: Analysis and Coding. Royce was extending this as he believed that for larger projects, the approach was “doomed to failure”. Despite this he envisioned that “customer personnel typically would rather not pay for them and development personnel would rather not implement them”. Royce also pointed to the fact that the “implementation described above is risky and invites failure”. The main concern was that the testing phase, which occurs at the end of the development cycle, “is the first event for which timing, storage, input/output transfer, etc., are experienced as distinguished from analysed”. Royce also highlighted that any issues in one of the phases can only feed back into the previous phase, and while this was something to hope for, the more realistic approach was to assume that an issue found in one phase would most likely result in a change to the software requirements. “Either the requirements must be modified, or a substantial change in the design is required. In effect, the development process has returned to the origin and the one can expect up to a 100-percent overrun in schedule and/or costs”. (Royce,W.W, 1970)

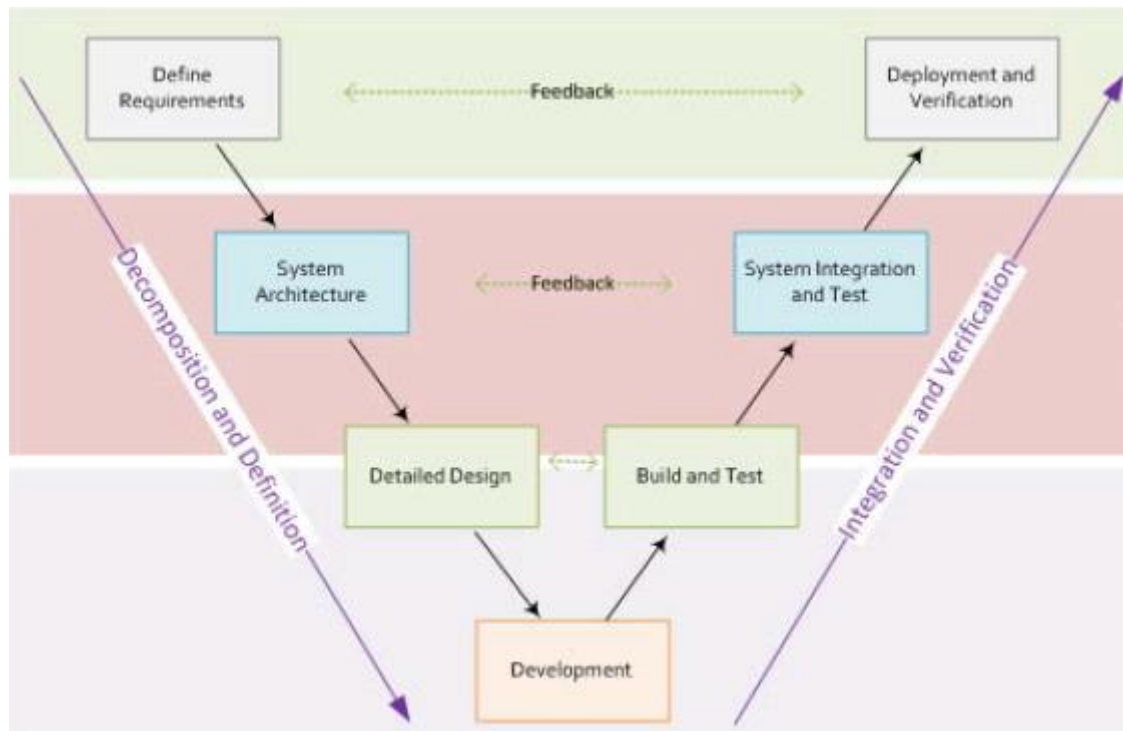
“Software development is a very young field, and it is thus no surprise that the simplified, single-pass and document-driven waterfall model of ‘requirements, design, implementation’ held sway during the first attempts to create the ideal development process”. (Larman C and Basilli, V, 2003) Other reasons for the waterfall idea’s early adoption or continued promotion include:

- Its simplicity made it easy to explain and recall;
- It aligns with management desire for an orderly and predictable process;
- It was widely promoted in texts and papers.

In summary, “the sequential document-driven waterfall process model tempts people to overpromise software capabilities in contractually binding requirement specifications before they understand their risk implication” (Boehm, 1991). Having discussed the Waterfall method, the next step is to look at the V-Model.

### 2.3.2 V-Model

The V-Model was first presented at the 1991 NCOSE symposium in Chattanooga, Tennessee. It is a variation on the Waterfall method. When reviewing the model, time should be considered as passing from left to right, however more complex versions also support iterations using a Z-axis to represent multiple deliveries.



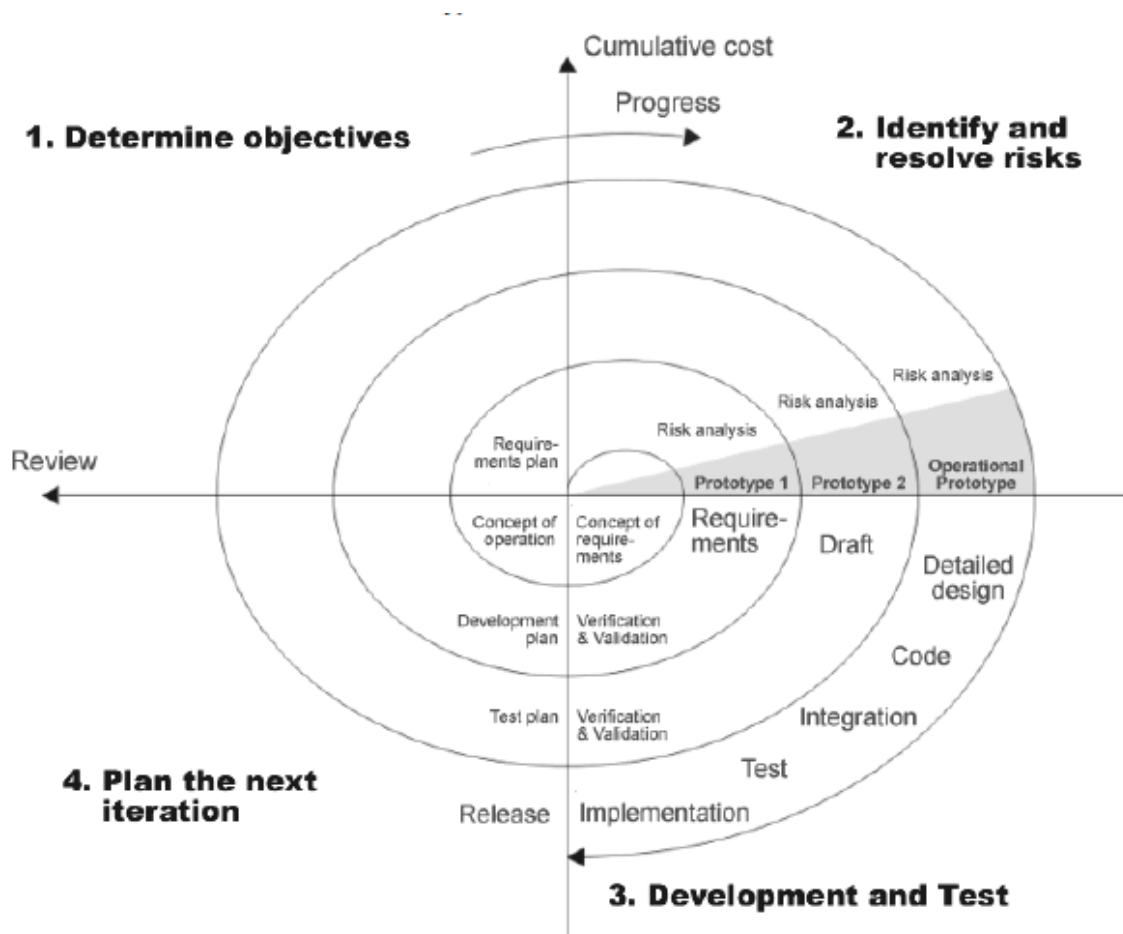
**Figure 3: This figure shows the V-Model. (Ruparelia, Navan B, 2010)**

The left leg of the V shape represents the evolution of user requirements into ever smaller components through the process of decomposition and definition; the right leg represents the integration and verification of the system components into successive levels of implementation and assembly. The vertical axis depicts the level of decomposition from the system level, at the top, to the lowest level of detail at component level at the bottom. (Ruparelia, Navan B, 2010)

Having reviewed the V-model, the next traditional methodology to consider is the spiral model.

### 2.3.3 The Spiral Model

The major issue with the waterfall projects is that “document-driven standards have pushed many projects to write elaborate specifications of poorly understood user interfaces and decision support functions followed, by the design and development of large quantities of unusable code” (Boehm, B,1988). Based on this Boehm defined the spiral model to put risk analysis at the heart of the development process.



**Figure 4: The Spiral Model: The development moves from the centre out and produces a prototype at the end of each cycle. (Boehm, B, 1988)**

As the project progresses, the prototypes evolve into the completed implementation. Risk management is used to determine the amount of time and effort to be expended for all activities during the cycle, such as planning, project management, configuration management, quality assurance, formal verification and testing. Hence, risk management is used as a tool to control the costs of each cycle.

The main advantage of the Spiral model is that it reduces the risk in software development by producing prototypes at intermediate stages of the project's lifecycle. If the project is simple enough then the spiral model cycle can be the same as a waterfall based project. The Spiral Model is the last traditional model to be reviewed. The next section summarizes the traditional models

### 2.3.4 Summary

This section provides a quick comparison between the three models outlined above.

Model	Advantages	Disadvantages
Waterfall	<ul style="list-style-type: none"> <li>• Easy to understand;</li> <li>• It is widely used;</li> <li>• Reinforces good habits.</li> </ul>	<ul style="list-style-type: none"> <li>• Does not match reality;</li> <li>• Software is delivered late in project;</li> <li>• There is significant administrative overhead;</li> <li>• Difficult and expensive to make changes to documents.</li> </ul>
V-Model	<ul style="list-style-type: none"> <li>• Easy to use with clear set of deliverables;</li> <li>• Test plans are developed earlier than the waterfall method, which improves the chance of success;</li> <li>• Works well when requirements are well understood.</li> </ul>	<ul style="list-style-type: none"> <li>• Very rigid, like the waterfall method, so it is difficult to adjust the scope of a project;</li> <li>• No early prototypes and there is no clear path for how to handle issues found in the testing phase.</li> </ul>
Spiral Model	<ul style="list-style-type: none"> <li>• High focus on the project risks;</li> <li>• Software is produced earlier in the project.</li> </ul>	<ul style="list-style-type: none"> <li>• Very dependent on the risk analysis, and the risk expert;</li> <li>• Can be costly to implement;</li> </ul>

		<ul style="list-style-type: none"> <li>• Does not work well for small projects.</li> </ul>
--	--	--

**Table 1: The advantages and disadvantages of the individual traditional software methodologies.**

In this section, we have reviewed three of the major methodologies typically used in software development. The next section examines the issues in project manager.

### ***2.4 Difficulties in Software Project Management***

The Standish Chaos report, which first appeared in 1994, stated that “70% of software projects end in failure”. This may be an overstatement, as if this were true the field of software development would be in crisis. However, software applications are prevalent in every element of modern life. This would suggest that a significant body of software is being developed successfully. (Glass, R, 2006) Despite this, Software Project Management is seen as difficult. Many projects fail to meet the success criteria of “on time and within budget”. These issues are more prevalent in software projects than in traditional projects. There are a number of characteristics of software development that make them more difficult to manage:

- Software projects are nondeterministic: When building a bridge or a home, we can create the plans and a detailed blueprint. We then use these to complete the construction. When building a software project, the exact configuration of that project is not known until the project is underway, and often only when it is near completion. Managing and scheduling a nondeterministic project is more difficult than a deterministic project. (Hall, N.G., 2012);
- Determining progress: Again using the example of a construction project, it is easy to see the state of the project. There are visible cues, for example, the foundation is laid, the roof is on, the outer structure is complete. Software projects do not have these cues. The project can sometimes not be available until all the parts are available. Also, many parts of the program have no visible cue. It is therefore more difficult to determine if the project is on track or if it has hit a problem. (Hall, N.G., 2012);



- Time pressure: Software projects are not as large as traditional projects. If the project overruns then the software becomes redundant. The organization will fall behind their competitors. They are also more subject to change during their lifecycle, as customers are uncertain of their requirements. (Hall, N.G., 2012);
- Experience: Software development is a practice that has been around for less than a century. Construction practices have been around for many millennia. The processes used and understanding of them have evolved as the systems have become more complex. In software, the rate of change is significant and the process may not have time to mature fully.

Having looked at the key differences between software and traditional projects, the next step is to look at reasons for project failure. The following table represents the main reasons which have been identified.

Reason	Description
Senior Management Not Involved	During a successful project, senior managers will contribute to the success by showing interest and promoting the project. They will also free up the necessary resources in a timely manner.
Too many requirements and scope changes	As the project develops, the project delivery requirements keep changing. This can have a poor impact on team moral.
Lack of necessary management skills	The management of software projects is difficult. The skills necessary are not present in the team, so the complexity of the project leads to problems which are not managed correctly.
Over-budget	The project goes too far over-budget and is cancelled.
Lack of necessary technical skills	The project team members are not skilled in the technologies that are required in the project. The technologies may prove

	harder to master than the team anticipated or the team does not use the technologies correctly resulting in problems for the project.
No more need for the system to be developed	The project is cancelled because it is no longer needed. This may be a change in business requirements or alternatively a symptom of the length of time the project has taken.
Over-schedule	The project is cancelled because it has taken too long.
Technology too new; did not work as expected	The problem is with a new technology which has either been oversold or misunderstood by the technical team.
Insufficient staff	There are not enough people available to execute the project.
Critical quality problems with software	The software produced does not meet the requirements, in that the software is not reliable, produces incorrect results or is not performant.
End users not sufficiently involved	The end users are not involved enough with the project. As a result, when the results are presented to them, they are unhappy with what they see. This can also lead to issues with business sponsorship. As the users are not involved the project loses business sponsorship.

**Table 2: Reasons for project cancellation (El Emam, K. 2008)**

The reasons for project cancellation are varied, though there are key issues which point to misunderstanding of the initial project requirements. Having completed a review of the difficulties of software project management, the next section provides a conclusion on project management.

## ***2.5 Conclusion***

This chapter has focused on project management. The chapter gives a brief history of project management, stating that project management has been in existence from ancient times and has evolved to its current state in the last century as businesses have realised the advantages of planning over ad-hoc delivery. It then defines projects and project management and discusses the trade-offs necessary to make a project a success. These trade-offs focus on accepting change in either cost, schedule or scope. The section describes three of the traditional software methodologies used in software project management. The focus of the chapter then turns to how costs and schedules are created, basing them on estimates. The chapter then examines the issues in software project management, specifically with reference to traditional non-software projects. The key difference between traditional projects and software projects is that software projects are nondeterministic and not transparent. This means that the components of software projects are difficult to determine at the outset and it is more difficult to see progress throughout the project. Finally, the reasons for cancelled projects are listed.

In summary, software project management is difficult. Success in software project management means accepting that change will happen. How to handle change is one of the reasons for Agile methodology. Also, trying to manage a project that evolves constantly and in which progress is not transparent is difficult. In this project, we will examine project tracking mechanism using gamification. The next section gives an overview of Agile.

## 3 AGILE

### 3.1 Introduction

This section of the document introduces Agile Software Development methodologies. The section starts by introducing the agile manifesto and some of the methodologies in use. The history of the agile movement is then discussed. The document then compares agile software methodologies against traditional software methodologies.

### 3.2 Agile Overview

This section of the document provides an overview of the Agile family of methodologies. It first looks at the Agile manifesto, then the guiding principles behind the manifesto.

The Agile manifesto was created in 2001. It represents the outcome of a meeting between leading advocates of Iterative and Incremental development (IID). As an outcome of this meeting an Agile manifesto was produced and some guiding principles for the project team. The Agile manifesto is as follows:

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

- 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.” (Fowler, M *et al.*, 2001)

The authors of the manifesto consider that processes and tools are important, but that emphasis should be on individuals and interactions. Tools and process can provide a means to track a project, but the manifesto advocates that direct contact between people is better. Similarly, spending time developing working software is more important than comprehensive documentation. Documentation should be kept to a minimum and should be where it is most convenient for the development and

maintenance team. Rather than spending time negotiating requirements between the customer and the development team, the effort should be spent on collaborating during the development. Wherever possible, customers should be co-located with the development team. This benefits the team, as issues can be resolved quickly, as all the people required to solve the problem are available. Finally, embracing change and being able to respond to it is more important than following a rigorous plan. Requirements change, particularly in large projects, so having to change a plan and all the documentation associated is time-consuming. It is better to have a process and a team that can respond well to change.

In addition to the manifesto, the agile movement founders defined 12 principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software: The team's aim is to deliver working software that provides some benefit to the customer;
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. When requirements change, this is part of the customer's need to get the product working in the best and most appropriate manner. This aim is aligned with the development team's objective and so should be welcomed;
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. The aim of the team should be to get this software to the end user as quickly as possible, with the improvements coming in small, but frequent intervals;
4. Business people and developers must work together daily throughout the project. Ideally the customer and development team would be co-located, however in the absence of this, the customer and the developers should strive to work together throughout the project. This level of interaction is key to the success of projects;
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. Agile works only when the team is motivated to succeed. It is necessary for the team to hold itself responsible, and without the motivation the team will not do this. Given the motivation, it is necessary to ensure that the environment and tools are available. Once that is in place, the team should be trusted to deliver.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. Documents are prone to mistakes, and without the conversation there is no opportunity to correct these misunderstandings. A conversation where everyone is comfortable asking questions is more effective and also more efficient.
7. Working software is the primary measure of progress. Other metrics can give indication of success, however, the amount of working software delivered is the key metric to judge a project by;
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. While overtime is permitted, it is not advisable for the team to work long hours on a constant basis. The team should work at a pace that they are comfortable with.
9. Continuous attention to technical excellence and good design enhances agility. If the team focuses on producing good design and develops an environment which supports technical excellence then the team will be better able to respond to change. Refactoring code to improve its design will ultimately result in a team that is better able to respond to change.
10. Simplicity – the art of maximizing the amount of work not done – is essential. No features or code fragments that are not absolutely required should be included in the development effort. In addition, trying to future-proof code and design is not recommended. The team should focus on delivering what is required and only that;
11. The best architectures, requirements, and designs emerge from self-organizing teams. In a self-organizing team, the team members will take on work where it is required. This allows a team member to apply their expertise rather than having one expert lead the project. Over time, with the best people working on the key areas that they are most suited to, the best architecture and design will emerge;
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly. The team is focused on delivering working software. However, it needs to have time allotted to review how it is doing as a team. This retrospective review allows the team to identify what went well, what could be improved and what went badly. This will allow the

team to make adjustments to their processes and find ways to improve their delivery. (Fowler, M *et al.*, 2001)

The principles can be applied to any Agile project. They can be adapted to varying degrees, but are present in the different Agile methodologies. The principles are designed to help the development team, guiding them in how to work in an Agile manner. The principles represent a breakdown of the elements of the manifesto and are designed to guide teams applying an Agile manifesto.

This section has given an introduction to Agile methodologies. Specifically it focuses on explaining the Agile manifesto and the principles which underline the manifesto. The next section of the document describes the history of Agile methodologies.

### ***3.3 History of Agile***

Despite the fanfare surrounding the Agile manifesto, Agile is not new. Iterative development has existed and been used in early projects in the 1960s and 1970s. Even the foundation paper for the Waterfall methodology, noted that only in the best case would an issue captured in one phase only impact the previous phase. It was more likely that all previous phases of the project would be impacted. In the 1970s while the waterfall methodology was growing in popularity, other work was been done to describe IID. Basili, VR and Turner, AJ (1975) describe IID:

“The basic idea behind iterative enhancement is to develop a software system incrementally, allowing the developer to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system.”

In his book, “Software Metrics” (Glib T, 1976) included discussions on evolutionary project management. This book contained some of the earliest material “with a clear flavour of Agile, light, and adaptive iteration with quick results, similar to that of newer IID methods”. Over the next two decades this iterative approach continued to gain traction with software engineers and academics, but its adoption was hampered by the US department of defence (DoD) adoption of Waterfall as a standard. Many papers in the 1980s and 1990s suggested new iterative methodologies or criticised the Waterfall methodology. In the mid-1990s the DoD relaxed its standards and this paved the way for adoption of IID methodologies. Included amongst these methodologies

were the family of methodologies that are now referred to as Agile methodologies. (Larman C and Basili VR, 2003)

Since the creation of the manifesto Agile has become common place in the development of the software. 94% of organizations who took part in the recent State of Agile survey (Version One, 2015) indicated that Agile was used in the organization. There have been a number of other developments in the past decade:

- **Lean Movement:** The development and / or popularity of the Kanban and Scrumban methodologies are tying Agile practices to Lean methodologies. These practices aim to eliminate waste in the development of software;
- **Agile in a global environment:** Many organizations are now global in their nature. The development team will often be located in a different global location to the customers. Indeed the development team may even be globally dispersed. This adds problems for time overlaps but also cultural differences. For Agile practices, which encourage face to face communication over documented requirements, this represents a difficulty. Research has begun into how this can be overcome;
- **Scaled Agile:** This is making the whole organization Agile. People have used Agile thinking to solve problems in different disciplines, such as Architecting, Design, Marketing, Portfolio Management and Program Management. (Laanti, M, 2014)

These examples are only some of the changes that have taken place since the Agile Manifesto was first introduced. Agile has continued to evolve, partly because “new innovations and new technologies come to markets with increased speed”, (Laanti, M, 2014) so organizations are under increasing pressure to be innovative.

This section has given a brief introduction to the evolution of Agile. The next section describes the key Agile methodologies in more detail.

### ***3.4 Key methodologies***

This section describes the key points of the survey used to establish which Agile methodologies are most actively used. It then describes the top five of these in detail.



### 3.4.1 Version One survey

The approach taken was to use a standard industry report on the use of Agile methodologies. The report selected was the “9<sup>th</sup> Annual State of Agile Survey” available from Version One. (Version One, 2015). 94% of organizations practice Agile. The level of use in organization varies.

Agile in the organization	Percentage
All of our teams are Agile	9%
More than half our teams are Agile	36%
Less than half of our teams are Agile	50%
None of our teams are Agile	5%

**Table 3: Shows the use of Agile within the respondent’s organization. While Agile is being adopted across organizations, the majority of project teams are not Agile.**

“87% of respondents said implementing Agile improved their ability to manage changing priorities”, while ”53% said that the majority, if not all, of their Agile projects have been successful”. The top three benefits of adopting Agile are:

- Manage changing priorities;
- Team productivity;
- Project visibility (82%).

The Agile methodology used was also surveyed

Methodology	Percentage
Scrum	56%
Scrum / XP hybrid	10%
Custom Hybrid (multiple methodologies)	8%
Scrumban	6%
Kanban	5%

**Table 4: Shows the top 5 methodologies used by the respondents. The others were used by 4% or less of the respondents The most popular methodology is Scrum with hybrids of Scrum popular too.**

The survey shows that Agile methodologies are widely used in industry and there is a belief that the methodologies have improved project delivery. The next section describes the most popular Agile methodology.

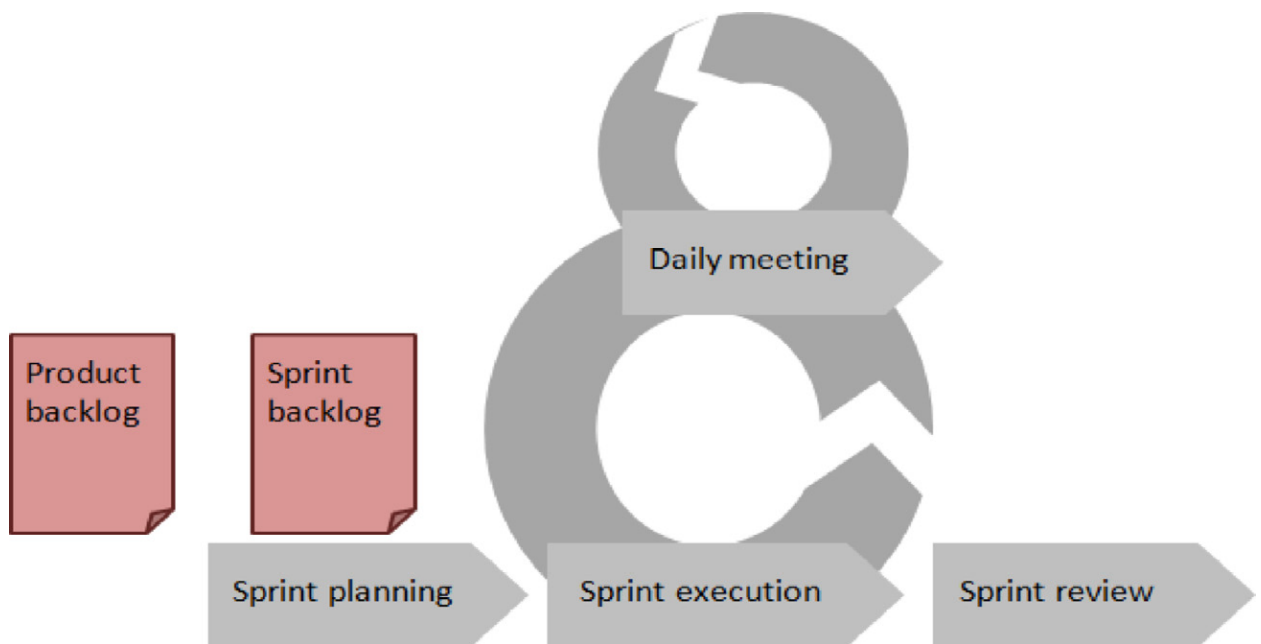
### 3.4.2 Scrum

This section of the document describes the components of a Scrum methodology.

#### 3.4.2.1 Definition

“Scrum is a method that aims to help teams to focus on their objectives. It tries to minimize the amount of work people have to spend tackling with less important concerns. Scrum is a response to keep things simple in the highly complicated and intellectually challenging software business environment”. (Schwaber K, 2000) Scrum does not include any specific development techniques but a method of managing a workload. The name is taken from a rugby Scrum where the team all pushes together in the same direction.

#### 3.4.2.2 Components



**Figure 5: Shows the main elements of scrum. The sprint execution is a time boxed period in which the team meets daily to discuss the progress of the work taken on in that period. The output of a sprint is working software components (Schwaber, K, 1997)**

Component	Description
Product Backlog	This is a prioritized list of customer requirements. The priority is set by the customer.
Sprint Backlog	This is the list of components or tasks being tackled in the

	current Sprint. The Sprint backlog is prioritized by the development team. This prioritization is completed during Sprint planning.
Sprint planning	During Sprint planning the team will examine the product backlog and take on work they feel is achievable in the Sprint. The amount of work taken on will depend on the team's ability to deliver, availability during the Sprint and understanding of the requirements. The team may also take on a requirement in a manner which matches a more ordered development path.
Sprint Execution	This is when the team develops and tests the software. The Sprint last for a number of days, typically boxed into two or three week periods.
Daily Meeting	This is a meeting where the team gathers to discuss the progress made in the Sprint. Typically, the team will consist of the development team, together with a Scrum Master and a representative of the customer. The team members will provide an update on their progress, focusing on what they did yesterday, what they plan to do today and any issues or blockages that will prevent them from completing their tasks. The Scrum Master is responsible for removing any blockages. The Scrum meeting is not intended to be a long meeting, but it is the main focal point of Scrum where issues should be raised.
Sprint Review	At the end of each Sprint the team meet to review the process. They will focus on what has worked well, what could be improved and what practises should be stopped.

**Table 5: The chief components of the Scrum methodology.**

#### 3.4.2.3 Benefits

The main benefits of Scrum are as follows:

- It is flexible throughout the project, it “provides control mechanisms for planning a product release and then managing variables as the project

progresses. This enables organizations to change the project and deliverables at any point in time, delivering the most appropriate release”. (Schwaber K, 2000);

- Allows the developer to produce the best solution as they learn as the project develops and the environment changes;
- “Small, collaborative teams of developers are able to share tacit knowledge about development processes. An excellent training environment for all parties is provided.” (Schwaber K, 1997)

Having examined the most popular methodology, Scrum, the next section looks at the Extreme Programming XP.

### 3.4.3 Extreme Programming

This section provides an overview of Extreme Programming (XP). In the survey of the Agile projects, XP on its own was not very well used. However, the use of XP and Scrum combined in a hybrid is the second most popular methodology.

#### 3.4.3.1 Definition

“XP turns the conventional software process sideways. Rather than planning, analyzing, and designing for the far-flung future, XP exploits the reduction in the cost of changing software to do all of these activities a little at a time, throughout software development”. (Beck K, 1999)

”Extreme Programming is a discipline of software development with values of simplicity, communication, feedback and courage. We focus on the roles of customer, manager, and programmer and accord key rights and responsibilities to those in those roles.” (Jeffries R *et al.*, 2001)

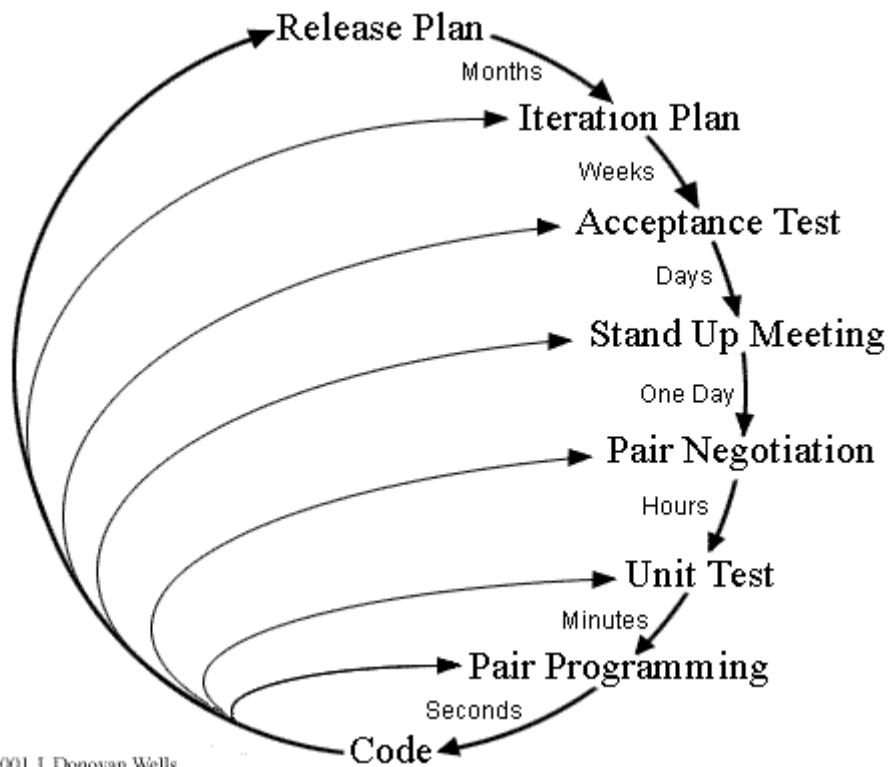
Practice	Description
Planning game	Customers decide the scope and timing of releases based on estimates provided by programmers. Programmers implement only the functionality demanded by the

	stories in this iteration.
Small releases	The system is put into production in a few months, before solving the whole problem. New releases are made often—anywhere from daily to monthly.
Metaphor	The shape of the system is defined by a metaphor or set of metaphors shared between the customer and programmers.
Simple design.	At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no duplicate code, and has the fewest possible classes and methods. This rule can be summarized as, “Say everything once and only once.”
Tests.	Programmers write unit tests minute by minute. These tests are collected and they must all run correctly. Customers write functional tests for the stories in a iteration. These tests should also all run, although practically speaking, sometimes a business decision must be made comparing the cost of shipping a known defect and the cost of delay.
Refactoring	The design of the system is evolved through transformations of the existing design that keep all the tests running.
Pair programming	All production code is written by two people at one screen/keyboard/mouse.
Continuous integration	New code is integrated with the current system after no more than a few hours. When integrating, the system is built from scratch and all tests must pass or the

	changes are discarded.
Collective ownership	Every programmer improves any code anywhere in the system at any time if they see the opportunity.
On-site customer	A customer sits with the team full-time.
40-hour weeks	No one can work a second consecutive week of overtime. Even isolated overtime used too frequently is a sign of deeper problems that must be addressed.
Open workspace	The team works in a large room with small cubicles around the periphery. Pair programmers work on computers set up in the center.
Just rules	By being part of an Extreme team, you sign up to follow the rules. But they're just the rules. The team can change the rules at any time as long as they agree on how they will assess the effects of the change.

**Table 6: This table shows the XP practices. (Beck K, 1999)**

### 3.4.3.2 Planning Loop



Copyright 2001 J. Donovan Wells

**Figure 6: This shows the planning loop for XP projects. (Beck K, 1999)**

In this diagram the release plan feeds into the iteration plan over the period of months, while the iteration plan feeds into the acceptance tests over a period of weeks. The code will constantly feed into the pair programming process.

### 3.4.3.3 Benefits

Practice	Benefit
Pair Programming	This results in continuous code review, which results in defects being caught in development and the number of defects being statistically lower.
Pair Negotiation	The designs are better and code length shorter and the team solves problems faster. This is due to on-going brainstorming and discussion.
Pair Programming	People learn more about the system and

	software development as the pairs share knowledge. At the end of the project release more people have a good understanding of the project.
Pair Programming, Iteration planning	People learn to work together and talk more often together, giving better information flow and team dynamics.
Small releases, continuous integration	The complexity of the release is reduced. The time spent on planning the release is reduced and the likelihood of error is reduced.
Test driven development	The tests are determined first. This allows the developer to see what is required by running the test. The requirements are mapped to tests.

**Table 7: This table outlines some of the practices and the advantages that they have. The focus on this is provided by the benefits of pair programming. (Cockburn A and Williams L, 2000)**

Having reviewed Extreme Programming in this section, the next methodology to be reviewed is Kanban. Although, Scrumban scored higher in the survey, it is based on Kanban, so it would be easier to discuss that first.

### **3.4.4 Kanban**

This section examines the Kanban methodology.

#### **3.4.4.1 Definition**

“Kanban is a Japanese word meaning a signboard, and it is used in manufacturing as a scheduling system. It is a flow control mechanism for pull-driven Just-In-Time production, in which the upstream processing activities are triggered by the downstream process demand signals”. (Ahmad, MO, 2013)



The Kanban methodology for software development was developed by David J Anderson in 2004. Its aim was to have the team focus on the workflow and try to limit the amount of work in progress at any one time. Kanban use a board to show the status of the work item, allowing the team to easily visualize how the process is going. Rather than organizing into iterations, the focus is on the flow of the stories, with more work being taken on when the team are able to tackle it. (Ahmad, MO, 2013)

#### 3.4.4.2 Principles of Kanban

Visualise the workflow
Limit work in progress (WIP)
Measure and manage flow
Make process policies explicit
Improve collaboratively (using models and the scientific method)

**Table 8: This outlines the principles of Kanban**

The main advantage of Kanban-driven operations is that WIP is reduced and the overall production flow can be balanced easier.

Having discussed Kanban, the next section discusses Scrumban

#### 3.4.5 Scrumban

This section looks at the Scrumban Agile methodology.

##### 3.4.5.1 Definition

Scrumban is a combination of Scrum methodology with Kanban methodology. The process is to start with what you have in Scrum and agree to evolve the process. The team introduce new artefacts and drop existing ones when the team agree they make sense. (Yeret, Y, 2015)

The aim of Scrumban is to make Scrum leaner. It utilizes elements from Kanban, but maintains structure and activities of Scrum. The team uses Kanban's visual workflow board and focuses on limiting WIP at every development stage. (Khan Z, 2014)

### 3.4.5.2 Principles

Principle	Description
Visualize the workflow	This is taken from Kanban. Visualizing the workflow helps the team to identify the bottlenecks in the project.
Pull the work	The work is pulled as and when needed, while in Scrum the work is all lined up at the start of the iteration. The tasks are not bound to individuals until they are pulled.
Limit Work in Progress Items	This is done at every stage of development based on team capacity.
Make the team rules explicit	The team rules are explicit and clear to everyone. This is to overcome the changing rules of a self-organizing team.
Shorter planning meetings	“The planning can still happen at regular intervals, synchronized with review and retrospective, but the goal of planning is to fill the slots available, not fill all of the slots, and certainly not determine the number of slots. This greatly reduces the overhead and ceremony of iteration planning”. (Ladas, 2008)
Review, Retrospectives and Daily Stand-up meetings	These meetings are maintained from Scrum.
Metrics and optional estimations in Scrumban	Scrumban prefers metrics like cycle time and lead time over velocity calculation.

**Table 9: This table outlines the principles of Scrumban (Yeret, Y, 2015)**

The following are the key benefits of Scrumban:

- The focus is on improved development times, rather than improving estimates;
- Provides more structure for the team than Kanban, by maintaining retrospectives and daily stand-up meetings from Scrum;
- Like Kanban, it removes the need for unnecessary elements from Scrum, such as lengthy planning meetings.

Scrumban completes the review of different Agile methodologies. The next section describes the key differentiators between Agile and traditional software methodologies.

### ***3.5 Agile versus other software development methodologies***

#### **3.5.1 Comparison**

In order to compare Agile methodologies with the traditional methodologies the paper first summarized the characteristics of each:

Area	Agile	Traditional
Approach	Adaptive, requirements, estimates and costs are adjusted as the project progresses.	Predictive, the requirements are identified at the start of the project. Estimates and costs are predicted.
Documentation	Documentation is not as important in Agile. The main aim is working software. Documentation provided should be the minimum to ensure that the software is understood.	Requirements documentation is viewed as the key piece of project. A main element in heavyweight methodologies is the big design upfront.
Process	Agile process adapt to the actual, rather than following a prescribed process.	Design a process that will work in the same manner no matter who is using that process.
Tools	Communication is preferred in a face to face manner, rather than through tools. The tools	Project management tools, Code editors, compilers, etc. must be in use for completion and delivery of

	can support, but face to face is considered better.	each task.
Collaboration	Agile tries to involve the customer as much as possible.	In traditional models, the customer is involved at the start, during requirements gathering and at the end of the project, during User Acceptance Testing.
Simplicity	Agile teams will develop software to be as simple in design as possible, covering only the functionality which is absolutely necessary.	The larger nature of traditional software project, with fewer releases, encourages the developers to try to future-proof deliveries. This can mean adding extra requirements and making design more complex than it needs to be.

**Table 10: A comparison of Agile and Traditional Software Methodologies (Awad, MA, 2005)**

There are significant differences between Agile and Traditional Software Methodologies. Agile has focused on trying to reflect the reality of software development.

### 3.5.2 Issues

Having compared Agile with traditional methodologies, the next step is to examine whether or not Agile resolves the issues found with them.

Reason	Description
Senior Management Not Involved	Agile development does not address this issue. Increased visibility, as a result of customer involvement, raises issues to management more frequently.

Too many requirements and scope changes	Agile methodologies are designed to welcome changes in requirements. The development work is iterative, so the change is less disruptive and moral is less likely to be impacted.
Lack of necessary management skills	The Agile team is responsible for itself. This would suggest less management skill is required. However, migrating from traditional to Agile methodologies is difficult and may require significant change to existing habits.
Over-budget	Budgeting for an entire project is not part of Agile projects. However, if the Agile project is costing too much it may still be cancelled.
Lack of necessary technical skills	Agile allows for teams to refactor designs as the team becomes more familiar with the technical skills. In addition, the self-organizing nature of the teams allows those who understand the new technology to take on the leadership.
No more need for the system to be developed	This is not impacted by Agile
Over-schedule	Agile will meet the requirements shortly after they have been defined. This mitigates against scheduling issues, as Agile projects produce some working software earlier.
Technology too new; did not work as expected	Agile mitigates this by meeting the issue earlier, before the team has invested heavily in the technology.
Insufficient staff	Agile mitigates this, as the velocity of the team would be identified and the

	likelihood of completing the project with the staffing levels will be clear
Critical quality problems with software	Software is tested as it is produced, namely in small iterations. Issues with quality should be identified early.
End users not sufficiently involved	End user involvement is a principle of Agile. If this is not case, then the project is not Agile.

**Table 11: Agile resolves the issues of traditional models (El Emam, K. 2008)**

Based on this table, it can be said that Agile methodologies have a positive impact on many of the issues of traditional software development.

This section compared against Agile and traditional methodologies. It then reviewed whether Agile resolved the issues in traditional methodologies, with clear indications that it does resolve them. The next section looks at the studies of Agile in academia.

### ***3.6 Studies in Agile***

Agile methodologies have provided a significant amount of studies in academia. Searching for the term in Google Scholar reveals over 7,000 responses. Filtering to the last 4 years reduces this to over 3,200 papers. To filter this down further, the thesis focuses on three terms which are most relevant to the thesis:

#### **3.6.1 Migrating to an Agile methodology**

This section looks at the key difficulties of migrating from traditional development to Agile software development. When migrating from a traditional to an Agile methodology, there are three main categories of issues which are typically encountered:

- **Development issues:** If you migrate to lightweight agile processes you either maintain the key processes in traditional processes and therefore lose the agility, or you remove the traditional processes and risk losing the safeguards that they provide. Using a small pilot project will result in variability between the Agile project and the existing projects. Teams have to adjust to the new

shorter lifecycle, though test driven development may assist in this as regression tests are built. Requirements are also different, with less focus on formality and non-functional requirements in Agile. It is reasonable to adjust Agile to include some of the requirement normally captured during traditional projects;

- Business Process conflicts: contracts and job roles can often be defined in relation to traditional projects. For example, the project manager role changes from command and control to facilitator. This has impacts for the employee, but also for their managers and HR representative. Their goals in a traditional project will be different from their goals in an Agile project;
- Team conflicts: Agile requires that the team be built around motivated software developers. When moving from traditional to Agile methodologies, the team may be motivated or demotivated. Another people consideration is the co-location of the team. This may result in the movement of staff from one area to another, which can have implications for managers and HR. (Boehm, B. and Turner, R., 2005)

### **3.6.2 Estimation in Agile**

In Agile projects, the most used techniques are subjective, for example, expert judgement or planning poker. Formal estimation methods, such as Case Based Reasoning and Wideband Delphi are not used. The most important factors in estimation are generally considered the skill of the team, the size of the task and the relative prior experience. The main type of size estimation in use is story point or use case estimation. When calculating the effectiveness of estimates, teams have used the Mean Magnitude of Relative Error (MMRE). This is the average of the Magnitude of Relative Error (MRE). It is calculated as:

$$(\text{Actual effort} - \text{Estimated Effort})/\text{Actual Effort}$$

This measure can be distorted by a bad estimate and is not necessarily an indication of a team that is poor at estimating. (Usman, M et al., 2014)

### **3.6.3 Agile and global software development**

Global software development has become more prevalent in recent times. Larger companies are setting up offshore sites to work on development projects. Other companies are using dedicated outsourcing companies to implement projects. Project teams can be split across country and timeline borders. Given the adoption of Agile it is natural that some of these organizations would try to adopt Agile practises. However, the global nature poses some specific challenges to Agile implementation.

- Lack of overlap for communication: Agile relies on communication, preferably face to face. This can be achieved through video conferencing. However, the time zones can still cause a problem. Team have overcome this by working later hours, implementing local Scrum teams and posting updates in advance of meetings;
- Collaboration difficulties: Aside from time issues, teams from different cultures and with different first languages can have difficulty collaborating. Teams may not understand each other's cultural habits, including how they respond to questions and challenges. This can be overcome by visiting sites and establishing sites;
- Communication bandwidth: Teams require a selection of communications methods to support global software development. This will include video conferencing, phone, instant messaging and SMS;
- Tool support: Without the necessary supporting tools, teams cannot successfully implement Agile global software development. (Hossain, E *et al.*, 2009)

If these issues are overcome, it is possible to successfully implement Agile in a global software development project.

### **3.7 Conclusion**

This chapter has provided an overview of Agile methodologies. It firstly describes the Agile manifesto and the principles behind the manifesto. For this thesis, the key principles include:

- Building projects around motivated individuals. The project is looking at how motivation can be maintained despite the necessary use of tracking tools;



- Agile processes promote sustainable development. It is hoped that gamification will help improve the tracking in the project. This is necessary to help communicate clearly the team's effort in delivering each iteration;
- At regular intervals the team reflects on how to become more effective: At the end of the project it is hoped the team has more accurate information to use when reflecting on progress. This accurate information should also be used as feedback to future estimation.

This section examined the history of Agile, highlighting that it has its roots in IID and briefly discussing the path of that evolution. Agile is now widely adopted in organizations throughout the world. The next section outlined some of the more popular Agile methodologies. A comparison between Agile and the traditional methodologies was then completed. This highlighted that Agile had been designed to solve many of the issues with the traditional approach. Finally, the section examined the major issues being studied in relation to Agile. These suggest that Agile is more difficult in a global environment and that it is not a trivial task to migrate a team from traditional methods to Agile approaches.

Having examined project management and then specifically Agile methodologies, the next chapter focuses on the second aspect of the dissertation, the motivation of software engineers.

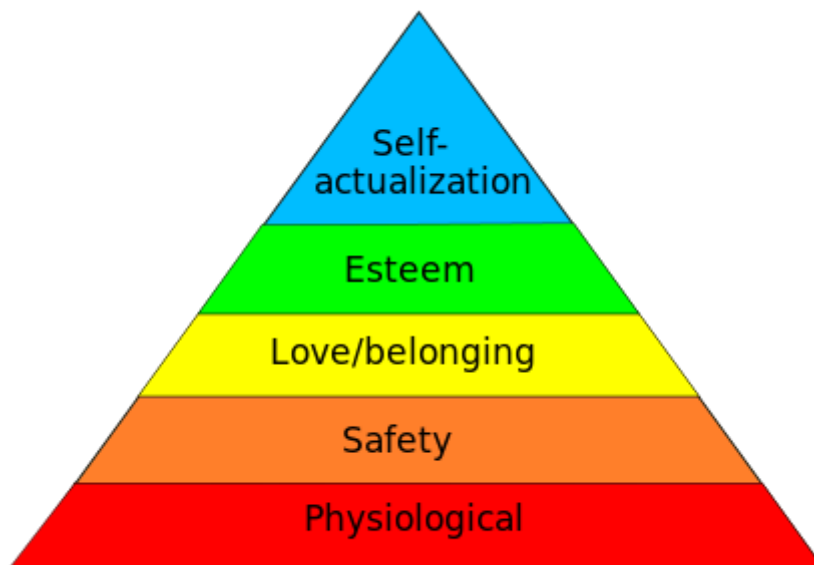
## 4 MOTIVATION OF SOFTWARE ENGINEERS

### 4.1 Introduction

This chapter will discuss motivation of software engineers. The chapter starts with a review of motivation theory combined with a brief discussion on how work has changed in the past century. The next section looks at studies into how to persuade individuals to do something they might not otherwise do. The next two sections focus on specific theories used in the project. Finally, the motivation of software engineers is examined.

### 4.2 Overview of Motivation

There are a number of papers in the area of motivation which are considered classics. Maslow's 1954 paper on the hierarchy of needs is the first of these.



**Figure 7: Maslow's Hierarchy Reproduced from simplypsychology, (2015)**

In this hierarchy the basic needs of human life: air; food; water and sleep, are represented at the base of the hierarchy. If an individual satisfies these needs, they move up to the next level of the hierarchy; safety. The need for safety represents the

need for personal security and in modern times the need for employment. The next level represents the need for a sense of belonging, with the need for family and friendship as part of this level. The next level is esteem, which represents confidence and self-esteem. The final level is self-actualization, this includes needs such as morality and creativity. So once people were satisfied at one level, they then looked at the next level to provide their satisfaction.

A second of these papers was introduced by Frederick Herzberg, (1966). He introduced the concept of hygiene and motivators. He found that “the things that make people satisfied and motivated on the job are different in kind from the things that make them dissatisfied”. This is contrary to understanding where we assume that satisfaction is the opposite of dissatisfaction. Herzberg argued that in relation to work the opposite of satisfaction is no satisfaction, and the opposite to dissatisfaction is no dissatisfaction. Motivation factors are intrinsic to the job, they include achievement, recognition, the work itself and responsibility; hygiene factors are extrinsic motivators, they include working conditions; salary, security.

Porter and Lawler, (1968) introduced a model of intrinsic and extrinsic work motivation. Intrinsic work is the work people do because they find it interesting while extrinsic work comes from the outside work and is motivation provided by the consequence of the work. An example is a reward you might receive for completing a task early. The model proposed that you could make work more interesting and provide more rewards to make employees more motivated. However, experiments find that some extrinsic rewards were demotivating. Deci, (1971) proposed Cognitive Evaluation Theory to explain that some extrinsic rewards, such as tangible rewards had a negative impact on intrinsic motivation.

Over the last quarter of a century a number of models have been developed. Locke and Latham, (1990), developed goal-setting theory which stated that to maximize peoples motivation they must have goals that are difficult and intrinsically rewarding to them, but also that their understanding of the goal is such that they know what they must do to meet the goal and they feel they can meet these goals. Building on previous work Frese, (2001) discusses the concept of personal initiative. This is where the employee “uses an active approach that is characterized by its self-starting and proactive nature and by overcoming difficulties that arise in the pursuit of a goal”. This is based on action regulation theory, which states that giving an employee greater control, or “decision latitude”, will result in increased motivation. Task specific motivation,

introduced by Kanfer, (1987), combines an individual's ability with their motivation in determining the success of the task. Motivation is made up of two parts; distal factors which are concerned with the task itself, and proximal factors which are concerned with the effort to keep at a complex task. Hackman and Oldman, (1980), argued "that the most effective means of motivating individuals is through the optimal design of jobs". They recommended jobs be redesigned to provide variety; afford considerable freedom; and provide meaningful performance feedback.

Cougar and Zawacki, (1980) introduced the job description survey for data processing JDS/DP. In this survey, data was collected on forty five variables to determine which were the most important and influential in employee motivation. This was collated for more than 1,000 analysts and programmers. This survey has become influential in motivation papers relating to software engineers.

In his later work Herzberg, (2003) highlights the impact of a job enrichment experiment. He applied seven principals of vertical job loading as part of this experiment. The principals are:

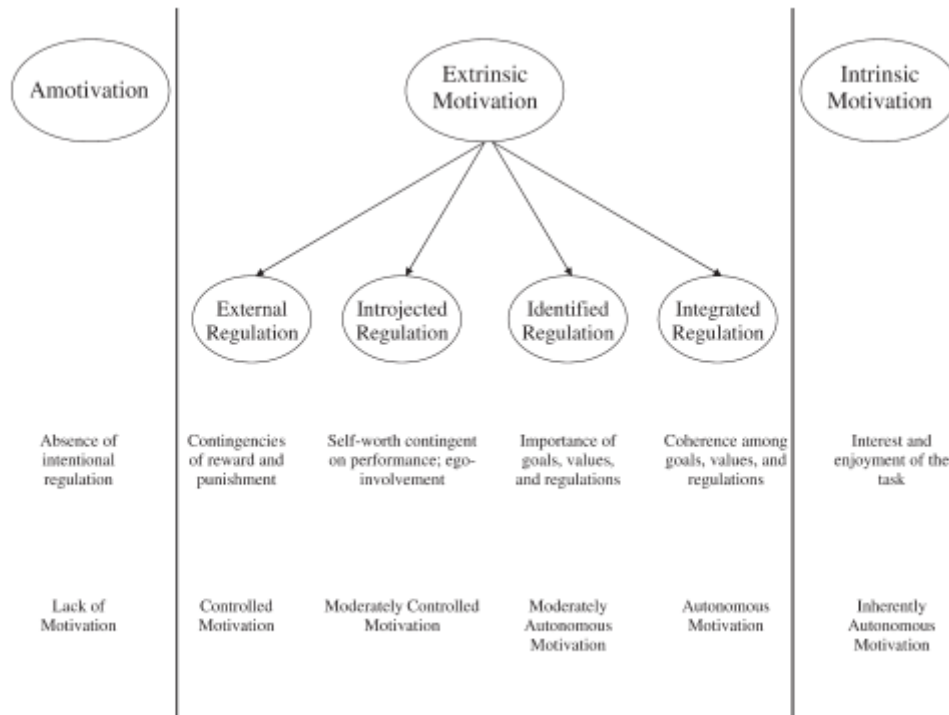
- Removing some controls while retaining accountability;
- Increasing the accountability of individuals for own work;
- Giving a person a complete natural unit of work;
- Granting a person a complete natural unit of work;
- Making periodic reports directly available to the workers rather than to supervisors;
- Introducing new and more difficult tasks not previously handled;
- Assigning individuals specific or specialized tasks, enabling them to become experts (Herzberg, 2003)

Gagné, M. & Deci, E.L., (2005), defined self-determination theory. This theory builds on a number of existing theories including earlier work by the authors (CET). The addition to the theory was to introduce amotivation, automotivation, and control motivation to differentiate between external positive and motivating factors.

Having discussed the history and development of motivational theory, the chapter now focuses in on self–development theory.

### 4.3 Self-determination theory

Gagné, M. & Deci, E.L., (2005) introduced self-determination theory, as a means to explain the difference between positive and negative extrinsic motivational behaviour.



**Figure 8: Self-determination theory reproduced from Gagné, M. & Deci, E.L. (2005)**

The theory, shown in figure 8, provides two different categorizations of motivation. Across the bottom of the diagram are various levels of two categories, control motivation and autonomous motivation. Controlled motivation is where the motivation is outside the control of the individual. Autonomous motivation is where the motivation relates to items the person can control. In addition to this, motivation is categorized into three high level categories:

- Amotivation, which is the absence of motivation, is added to the discussion. This is where a person does not act;
- Extrinsic motivation: This is external motivation and is decomposed into four separate sub-categories:

- External Regulation: This is the use of rewards and punishments for the completion of tasks. These are considered controlled motivations and these can have a negative impact on intrinsic motivation;
- Introjected Regulation: This relates to self-worth and ego. The success or failure of the tasks is reflected in the employees self-worth. It is controlled motivation, but not to the same extent as external regulation;
- Identified Regulation: This is the area of goals and values. It relates to the expected norm. This is moderately autonomous motivation, because it is the individual's decision to go with the norm or not;
- Integrated Resolution: This is the alignment of goals with the goals of the individual. If the goals are aligned then the individual will be motivated in a manner that is similar to their own intrinsic motivation. Often behaviour becomes part of the person's intrinsic motivation;
- Intrinsic motivation remains the same as in other models. Basically, a person has autonomy in their job and is working on something that they like to do.

Having examined SDT motivation theory, the next section examines persuasion model.

#### ***4.4 Persuasions Models***

Work on persuasive motivation has identified that there are multiple routes to persuasion. Petty and Cacioppo, (1984) described an Elaboration Likelihood Model, which included two approaches to persuasion: central and peripheral routes. A central route means that the elaboration likelihood is high; the subject is engaged by the arguments for recommendation. The subject will have examined the arguments, reviewed their own experience and made associations and drawn inferences with the proposal. In this manner it is more likely that the persuasion will be effective in the long term, or be internalized. Peripheral route is the opposite, in that the subjects will not have considered the arguments and while there may be an initial uptake on the persuasive idea, it is unlikely to be internalized. Although the model focuses on the two extremes of central and peripheral routes, the persuasive argument can in fact be situated anywhere between the two extremes.

Having discussed the theory of persuasive models, the next section examines what factors motivate software engineers.

#### 4.5 *Motivating Software Engineers*

This section examines what motivates software engineers. Sharp, H *et al.*, (2009) conducted a thorough review of the literature on motivation of software engineers. As part of this they reviewed the existing papers to determine whether software engineers were different from other groups of workers. The results were that 54% of papers concluded that software engineers were different, 24% concluded that software engineers were not different, while the remaining 22% concluded that the context was important to determining whether software engineers were motivated differently from other groups. In their review they attempted to review a number of research questions. The first review question was “what are the characteristics of software engineers?” The main characteristics found were the need for “growth and independence”. The need for growth may be related to the fast changing nature of technology and the tendency for IT to evolve new languages and techniques. A software engineer who continues to do the same job in the same manner will not be very marketable. Independence relates to autonomy, and may be due to the fact that the work is something that can be done as a creative task not subject to “overbearing management”.

The next question “What motivates and demotivates software engineers?” Sharp, H *et al.*, (2009). The most cited aspect is “the need to identify with the task”. Demotivation factors relate to Herzberg hygiene factors. They also found that some factors could be motivational or de-motivational depending on the context. The following table examines the motivators and aligns them with SDT.

Motivators of Software Engineers	Self Determination Theory
Identify with the task (clear goals, personal interest, know purpose of task, how it fits in with whole, job satisfaction; producing identifiable piece of quality work).	Intrinsic
Employee participation/involvement/working with others.	Integrated Regulation

Good management (senior management support, teambuilding, good communication).	Identified Regulation
Career Path (opportunity for advancement, promotion prospect, career planning).	Integrated Regulation
Variety of Work (e.g. making good use of skills, being stretched).	Intrinsic
Sense of belonging/supportive relationships.	Intrinsic
Rewards and incentives (e.g. scope for increased pay and benefits linked to performance).	External Regulation
Recognition (for a high quality, good job done based on objective criteria).	Introjected Regulation
Development needs addressed (e.g. training opportunities to widen skills; opportunity to specialise).	Integrated Regulation
Technically challenging work.	Intrinsic
Job security/stable environment.	External Regulation
Feedback.	Integrated Regulation
Autonomy Work/life balance (flexibility in work times, caring manager/employer, work location).	Intrinsic
Making a contribution/task significance (degree to which the job has a substantial impact on the lives or work of other people).	Intrinsic
Empowerment/responsibility.	Intrinsic
Appropriate working conditions/environment/good equipment/tools/physical space/quiet.	Integrated Regulation
Trust/respect.	Intrinsic
Equity.	Intrinsic
Working in company that is successful (e.g. financially stable).	External Regulation

**Table 12: Shows the motivation factors associated with software engineers. Sharp, H *et al.*, (2009)**

De-motivator	Self-Determination Theory
--------------	---------------------------



Risk.	External Regulation
Stress.	Introjected Regulation
Inequity (e.g. recognition based on management intuition or personal preference).	External Regulation
Interesting work going to other parties (e.g. outsourcing).	External Regulation
Unfair reward system (e.g. Management rewarded for organisational performance; company benefits based on company rank not merit).	External Regulation
Lack of promotion opportunities/stagnation/career plateau/boring work/poor job fit.	Introjected Regulation
Poor communication (Feedback deficiency/loss of direct contact with all levels of management).	Introjected Regulation
Uncompetitive pay/poor pay/unpaid overtime.	External Regulation
Unrealistic goals/ phoney deadlines.	External Regulation
Bad relationship with users and colleagues.	External Regulation
Poor working environment (e.g., wrong staffing levels/unstable/insecure/lacking in investment and resources; being physically separated from team).	External Regulation
Poor management (e.g. poorly conducted meetings that are a waste of time).	External Regulation
Producing poor quality software (no sense of accomplishment).	Introjected Regulation
Poor cultural fit/stereotyping/role ambiguity.	Introjected Regulation
Lack of influence/not involved in decision making/no voice.	Introjected Regulation

**Table 13: De-motivators associated with software engineers. Sharp, H *et al.*, (2009)**

This section has focused on what motivates software engineers and whether they are different from other work groups. The next section concludes the chapter.

## ***4.6 Conclusion***

This chapter has reviewed the motivation in general and then software motivation specifically. A brief summary was given of some of the key models and developments in motivation literature, with more detail provided on SDT (Gagné, M. & Deci, E.L., 2005). This theory highlights that motivation is complex, with some factors such as rewards being de-motivating if not managed correctly. It is important to review the experiment against SDT as this will give an indication of the long term acceptance of the behavioural change. The next section of the chapter describes the elaboration likelihood model (Petty and Cacioppo, 1984). The persuasion route used in the experiment will be described as part of the experiment. The main motivation and de-motivation factors for software engineers are then compiled as part of a review of motivation and software engineers. These are then presented with the different categories of motivation to show whether they can be expected to have a long term motivational affect. These factors will be used in designing the gamification experiment. Having reviewed motivation in this chapter, the next chapter examines gamification.

## **5 GAMIFICATION**

### ***5.1 Introduction***

This section of the document introduces the topic of gamification. The first sub-section introduces the methodology used to complete this literature review. The next section provides an overview and some definitions of gamification. The next section describes elements of the games. The next section looks at projects which have been completed which included gamification. The final section looks at gamification and Agile.

### ***5.2 Approach***

This section of the document examines the methodology used to complete this literature review. The section describes the process used to retrieve the papers, how they were rated and how they were selected for inclusion in the review.

#### **5.2.1 Approach**

This section of the document outlines the approach taken to the literature review of gamification.

The approach taken was to first search using Google Scholar for articles relating to an overview of gamification. Terms were identified and the search completed. The volume of papers, and the recent nature of research in the field resulted in a filtering to those in the past 4 years. Papers not in English were also filtered, not based on their worth, but based on the authors inability to translate them. Only papers which had been cited were included.

Having established a list of papers as a basis, the next step was to categorize papers into subject area. The subject area was chosen based on the title and the journal that the paper existed in. The key papers of interest for this research were:

Overview of gamification: For the overview of gamification, the approach taken was to review the abstract of the papers found. The paper was then included for full review if it was genuinely an overview of gamification paper, or provided a discussion point on gamification, not included in other papers. A review of the references in each of the

selected papers was included, to see if any key papers were missed by the initial selection.

Gaming Elements: Only papers which described elements of gamification were considered. A review of the references in each of the selected papers was included, to see if any key papers were missed by the initial selection.

Papers relating to software development: This focused on papers that contained gamification as some part of software development, for instance requirements gathering or version control. A review of the references in each of the selected papers was included, to see if any key papers were missed by the initial selection.

Literature reviews for other subject areas: Other subject areas are only briefly covered in this paper to provide a context for gamification. For these papers it will be sufficient to review existing literature reviews where possible.

Having outlined the approach to the literature review, the next section describes the results.

### 5.2.2 Results

This section of the document describes the results of the literature review

Term	Total Papers	Since 2011	English	Cited
Gamification Overview	11	8	8	6
Gamification Review	5	5	5	1
Defining Gamification	1030	809	667	263

**Table 14: This table shows a breakdown of the papers based on the initial search terms.**

Initially, all papers were included. This was then filtered to those papers since 2011. Papers not in the English language were then excluded and finally to filter further, only papers which had been cited were included for further analysis.

Subject	Percentage
Education	26%
Overview	19%
IT/Data	11%
HCI	9%
Social Networks	8%

Games	7%
Business	4%
Crowdsourcing	3%
Health	3%
Other	3%
Energy	1%
Legal/Crime	1%
Mobile	1%
Media	1%
Robotics	<1%
Military	<1%

**Table 15: This table shows a breakdown of the papers by subject area. The subject area was chosen based primarily on the title and the journal that the paper appeared in.**

Subject	Percentage
Overview	38%
Other	15%
Education	8%
Experiment	8%
Game Elements	8%
Health	8%
Motivation	6%
Games	4%
Software	4%
IT	2%

**Table 16: This table shows a breakdown of papers which were considered overview following the review of abstract.**

### ***5.3 Overview***

This section of the document describes gamification. It starts with the definition of gamification and then provides a context for gamification. The section describes the perceived benefits and also the challenges to those benefits.

### 5.3.1 Definitions of Gamification

This section of the document looks at the definitions of the gamification. The first definition is provided by Deterding S, (2011) and is the most widely cited. Deterding defines gamification as “use of game design elements in non-game contexts” (Deterding S, 2011) . In order to understand this definition we need to examine the components of the definition:

- **Game:** As part of this definition games are distinguished from play. Play is free form while games are “playing structured by rules and competitive strife toward goals”. (Deterding S, 2011)
- **Design:** In this definition, design refers to game based design, not game devices. Gamified applications contain elements of design from games, but are not proper games. It is possible the user may choose to play the game or not play the game while still completing the function.
- **Elements:** Refers to the distinction between fully fledged games and parts of games used in another application. “The characteristic of “gamified” applications might be that compared to games, they afford a more fragile, unstable ‘flicker’ of experiences and enactments between playful, gameful, and other, more instrumental-functionalist modes.” (Deterding S, 2011) Deterding looks to “Ten Ingredients of great games” (Reeves B and Read JL, 2013) to define what elements are normally found in a game. While the elements can appear in non-gaming contexts, they are normally found in games. In actual games, there is likely to be more than one of these elements.
- **Non-gaming context:** Gamification uses elements of games for purposes other than their normal expected use as part of an entertainment game. The gamification is not limited to any single context and no context is excluded, excepting the use of game elements in game environments. This has been dropped primarily because it is circular in nature.

Having reviewed Deterding’s definition, there is a need to look at other definitions of gamification. The other definition quoted most often is from Huotari and Hamari, (2012). They define gamification as “a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation”. (Huotari and Hamari, 2012)

To gain a better understanding of this we will look at the parts of the definition:

- Service: Huotari and Hamari, (2012) were focused on defining gamification with respect to the service marketplace. They use the following definition of a service. “The application of specialized competences (knowledge and skills), through deeds, processes, and performances for the benefit of another entity or the entity itself”. (Vargo and Lusch, 2004). This definition ensures that gamification can be applied anywhere that an act that assists another entity can be applied;
- Affordance: This is a relation quality between an object and a subject. This implies the definition of gamification has to affect not just the application that you are changing but also that different subjects will react differently;
- Gameful Experience: Here the focus is on the user’s experience rather than the game elements or mechanics;
- Overall value creation: The gamification should support the user in meeting their needs.

This definition is more aware of the subjective nature of gamification. It focuses less on the game mechanics and more on the user’s experience.

Zichermann *et al.*, (2011) define gamification as “The process of game-thinking and game mechanics to engage users and solve problems”. This definition is at a very high level, and can be applied to any use of motivation. This definition is difficult to critique expecting that the vague nature makes it difficult to assign gamification to specific situations.

Having reviewed the main definitions of gamification within academia, we will now look at some of the definitions that are used by the industry. As gamification is a relatively new term, it is important to review it from the perspective of industry, particularly given its current popularity. From the industry there are a number of different definitions.

- The verb 'to Gamify' means to apply game mechanics in everyday applications and situations to boost engagement, fun and good behaviors. [Gamify, Inc, 2015)

- Gamification is the process of integrating game mechanics and dynamics into a website, business service, online community, content portal, marketing campaign or even internal business processes, in order to drive participation and engagement”. (Bunchball, Inc., 2015)

These industry definitions are very much sales based, providing promises of improvements in behaviour using the gamified applications. (Llagostera, E, 2012)

Deterding, (2013) has reviewed his definition and extended it to provide a view which is not only focused on the technical aspects. Gamification has become an industry focused on “driving any desired activity by tracking it and adding a feedback layer of points, badges, leader boards, and incentives on top”. The industry has ignored research on motivation and the ethics of influencing behaviour. Deterding has addressed this issue with six enhancements to his original gamification definition.

Rethink the scope of gamification: The scope of gamification needs to be extended beyond the game itself. The context in which the game is used in has an impact on the gamification.

- Autonomy: Having to play games as part of one’s profession is generally described as less enjoyable and less engaging by practitioners, and comes with more frequent unpleasant experiences of being controlled. (Deterding S, 2013) Games satisfy the basic psychological needs of autonomy, which leads to enjoyment (Deci and Ryan, 2012);
- Gaming the system: In games a degree of gaming the system is acceptable, even laudable. There is a limit to this, as rules are intended to be bent not broken. However in work, the gamification application does not have the same “bracketing” to ensure what is allowable in gaming the system and what is considered beyond the norms (Deterding S, 2013);
- Acting out of bounds: “Embarrassment lies at the heart of the social organization of day-to-day conduct”. (Scheff, TJ, 2003) People regulate their behaviour to avoid embarrassment. Expecting people to playfully and / or gamefully engage with them in a non-gaming context is asking them to act out of bounds. Gamification must include setting the comfort level of members in the workplace to play the game;



Rethink the goal of gamification: What we are trying to achieve through gamification has to be rethought. Typically, the goal is to modify user behaviour using elements from game design. However, there is a need to refocus the goal of gamification:

- From elements to experiences. Rather than focus on the mechanics of games, gamification should focus on giving the user a gaming experience. (Deterding S *et al.*, 2011). The focus shift from game element to game experience brings the original Deterding definition closer to that of Houtari and Hamari, (2012);
- Playful design: By focusing on the relationship with games, rather than play, the definition misses on a body of work related to playfulness in work. The definition is too restrictive to cover “motivations like curiosity, or design for exploration, transgression, creativity, or innovation”. (Deterding S, 2013);
- Motivational Experiences: The ultimate aim of gamification is to motivate behaviour change in users. Gamification, is therefore a subset of the motivational design and in particular pervasive design. Gameful and playful designs are tactics for applying motivation. (Deterding S, 2013).

Rethink Gameful Experiences: Currently gamification focuses on making something which is not enjoyable more enjoyable by adding game elements on top of it. This assumes “a game design element produces one (and only one) kind of motivational experience across users and contexts”. (Deterding S, 2013) However, enjoyment is a relationship, or an affordance in Human Computer Interface (HCI) terms. “An affordance is not an objective feature of a design element, but a relational quality of both object and subject”. (Deterding S, 2013) Under the new definition gamification should be part of the whole experience.

Rethink Gamification Design: Rather than using game design patterns, which are restricted in their domain context, Deterding, (2013) has determined that a “re-envisioned gamification design method would entail formalising desired motivational experiences in the form of design lenses, using these lenses to analyse target activities, and then engage in iterative experiential prototyping until the total prototyped socio-technical system affords the targeted motivational experiences”. In summary build the application with the motivational experience as part of the goal, and ensure through prototyping that it is achieved.

Rethink Gamification Ethics: The aim of gamification, as defined, is to modify the behaviour of users, The ethics have been challenged, as the desired behaviour is dictated by those designing the gamification. Little thought was given to this in the original definition. However, ethical gamification (as with any other design practice) would include:

- support the users well-being by being a tool for “positive design”. (Desmet E and Pohlmeier, D, 2013);
- a practice performed virtuously, excellently in itself;
- “something that realises, furthers, or is at least congruent with living a good life with others” (Deterding S, 2013).

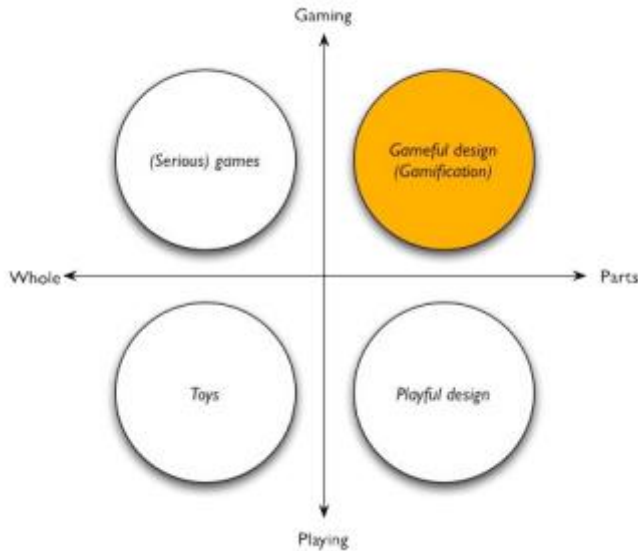
Rethink Gamification’s Purpose: Currently gamification focus is on modifying a behaviour to some perceived better behaviour as defined by the organization. The process deals with the symptom but not with the underlying reasons for the behaviour. The purpose of gamification needs to refocus to the real problem and therefore improve the user’s wellbeing.

In summary, gamification needs to take a more holistic view of the problem domain, understanding the underlying causes of the behaviour and designing the gaming experience as part of the whole solution, rather than bolting it onto an existing application. The field of gamification should form part of the motivational studies, specifically persuasive motivation. Deterding advocates extending the definition to include playfulness, however, this is not something the author agrees with, as there is sufficient difference between the two to maintain them separately. Finally, as with any behavioural modification, there are ethics that need to be considered.

For the purpose of this experiment, gamification will focus on the mechanical definition. The experiment will attempt to apply game elements to an existing process. This is a bolt-on to an existing application, which does not concur with the later definitions of gamification.

### 5.3.2 Situating Gamification

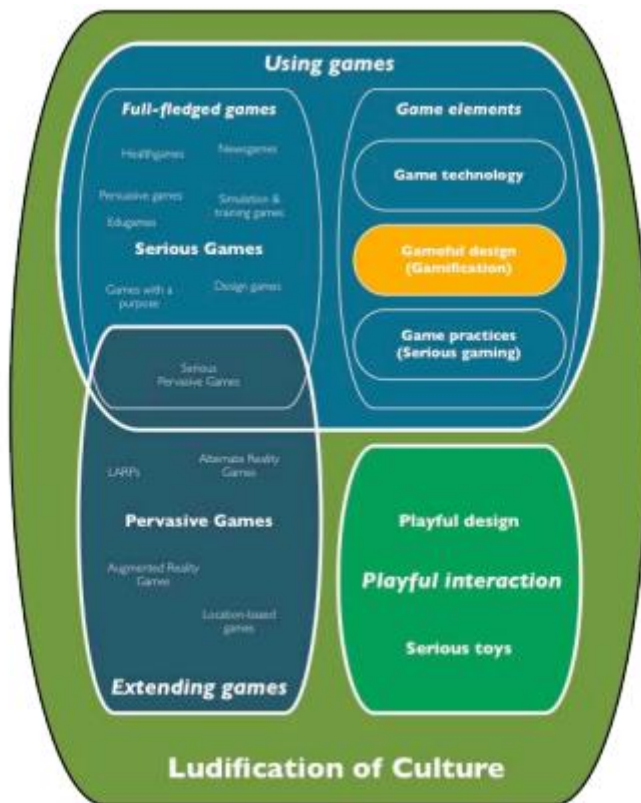
This section attempts to describe gamification in its relationship to other subject areas, specifically play, serious games and toys.



**Figure 9: Shows where gamification is placed in relation to other subject areas (Deterding S, 2013).**

It shows that, while serious games are in principle games, gamification is only using parts of games and are not in themselves a game. Gamification is also separate from playful design, in that it is more closely related to games than play, given that it will have structure and rules.

While this first graphic shows gamification in relation to serious games, “games for serious purposes”, and to playful design, the next diagram shows gamification in relation to the overall ludification of culture.



**Figure 10: Ludification of Culture (Deterding S, 2013)**

Within the socio-cultural trend of ludification, there are at least three trajectories relating to video games and HCI: the extension of games (pervasive games), the use of games in non-game contexts, and playful interaction. The use of games in non-game contexts falls into full-fledged games (serious games) and game elements, which can be further differentiated into game technology, game practices, and game design. The latter refers to “gamification”.

So in line with Deterding’s, (2013), gamification definition, gamification is part of the use of games but is distinct from fully fledged games but relies on game elements. Using the Huotari and Hamari, (2012) definition, gamification can sit in two of the three trajectories, using games and pervasive games. Gamification also applies to game elements in this definition. Although playful design and interaction are an interaction and so could be included by the scope of this definition, the necessity of a gaming experience has excluded it.

Having discussed the situation of gamification, the next step is to discuss the issues with gamification.

### 5.3.3 Issues with gamification

This section highlights some of the issues that have been raised in respect to gamification. Before doing this the paper includes a subset of a survey completed by Shahri *et al.*, This survey focused on the ethical aspects of gamification. The following are the issues that relate to the experiment:

- Gamification can lead to tension amongst colleagues, when applying a leader board, or on the individual, when used as a monitoring system;
- Gamification captures a lot of personal data. This can cause privacy issues and may lead to freedom of information issues. It also makes member's vulnerable, as they may overlook the data gathering;
- Gamification could push people beyond the requirements of their job. To get to the top of a leader board, people might work overtime constantly.

Some of the other issues raised are highlighted in this table below.

Issue	Description	Rebuttal
Gamification is presented as a "relevant topic of discussion and as a desired buzzword for businesses". (Llagostera E, 2012)	As a result, gamification "keeps the term 'game' and puts it right up in front, drawing attention to the form's mysterious power. But the kicker comes at the end: 'the ify' suffix it makes applying that medium to any given purpose seem facile and automatic" (Bogost,2015). However, their efforts are the same as common marketing practices of selling generic solutions that can be adopted by	All terms are subject to similar hype cycle. This does not impact on the validity of the term, but is more a reflection on the IT industry.

	several brands (Bogost,2015)	
Advocates of gamification have profited from the term	Advocates are active in the definitional debate around the term, and also work to produce more and more discussion and a public presence for it.	Thought leaders will naturally be in the public extoling their ideas. Again this is not specific to gamification.
Gamification manipulates people's emotions and motivations	Gamification's appropriation of video games is not focused in their learning potential, but on their capacity to generate effective, informational and economic value through the shaping of individual's emotions. (Llagostera E, 2012)	Gamification can be misused but the actions may also be to get people to perform tasks which they are paid to complete.

**Table 17: Shows some of the main points against gamification. For completeness I have added a rebuttal, which is the author's own opinion.**

Having completed the overview of gamification, the next section of the document describes Game elements.

#### ***5.4 Game Elements***

This section of the document describes game elements. It provides a framework of games; then the different levels of game elements and finally provides a brief description of some game elements.

Framework Element	Description
Purpose of the game	This is the aim of the game, for instance to checkmate one's opponent in chess.
Procedure for action	This is the method of play, what the

	players have to do to play.
Rules governing action	This may be straightforward, for example start when the starting pistol sounds and run for 100 meters, or a complex combination of rules, for example, football which has rules governing length of play, offside, valid tackles, use of hands.
Number of required participants	This will be a minimum and a maximum number of players.
Roles of participants	All players may have the same role, or in some games players have different roles. For instance football has goalkeepers and outfield players.
Results or pay-off	Value assigned to the outcome of the action. This can be a medal, money or some other prize.
Abilities and skills required for action	This can be very simple set of skills or complex.

**Table 18: This table shows the framework for games. It will be used when defining the gamification experiment (Avedon EM, 1981)**

In gamification, game elements can be used at different levels of abstraction.

Level	Description	Example
Game interface design patterns	Common, successful interaction design components and design solutions for a known problem in a context, including prototypical implementations	Badge, leaderboard, level.
Game design patterns and mechanics	Commonly reoccurring parts of the design of a game that concern	Time constraint, limited resources, turns.

	gameplay	
Game design principles and heuristics	Evaluative guidelines to approach a design problem or analyse a given design solution	Enduring play, clear goals, variety of game styles.
Game models	Conceptual models of the components of games or game experience	Challenge, fantasy, curiosity; game design atoms.
Game design methods	Game design-specific practices and processes	Playtesting, playcentric design, value conscious game design.

**Table 19: Levels of abstraction in the use of game elements in gamification. This is used in the describing the experiment (Deterding, S, 2011).**

The following table represents a selection of game elements. The purpose is to introduce the reader to the elements, as some of these will be selected for the experiment. The list is not intended to be complete, however the major elements of games are included.

Element	Description
Appointment Dynamics	Where the user has to arrive at a place by a certain time to gain reward or status.
Avatars	This is a representation of the self in the game. In the gaming world people can build characters to represent themselves or can play as characters.
Competition under rules that are explicit and enforced	Competition is an element of games. The rules of the game are clearly stated and enforced.
Discovery	This is where the players discover or learn something as they play. This is also referred to as exploration.
Feedback	It is possible to receive feedback on how



	you are progressing. This is often in the form of graphical indication of health.
Levels	In games it is possible for the game to become more challenging as you proceed through the game environment. At different levels new challenges can be added, but new rewards can also be available.
Loss Aversion	Rather than reward for achievements this game mechanism takes away rights.
Lottery	In this case the game winner is based solely on chance.
Marketplaces and economies	Within the game it is possible to buy or upskill based on money or trades.
Narrative context	This is a context in which the game is played. This takes the form of a back story in which the game is placed.
Parallel communication systems that can be easily configured	Games will support communication between the players, directly through the game. The systems themselves must be reliable and easily configurable otherwise they will detract from the game.
Reputations	In gaming a user will be able to build a reputation based on their gameplay. The reputation can include collection of powers or be based on level of skill associated.
Rewards	When completing a task a player is given a reward. This may be a badge indicating a higher level of skill or a point's reward. It may even be an out-of-game reward, where the player is able to exchange points earned in the game for a material

	reward in the real world.
Teams	Game players can combine into teams to compete against other teams or alternatively to achieve a task by working in unison.
Three-dimensional environments	The game environment should be attractive for the game to be successful. Current games use three-dimensional graphics to immerse players in increasing realistic worlds.
Time pressure	Limited time to complete the game.

**Table 20: A selection of game elements or mechanisms. Some are taken from Reeves and Read’s “Ten Ingredients of Great Games” (Reeves D and Read TJ, 2013) while others are taken from gamification.org website.**

Having outlined the game elements, the next step is to look at the existing use of gamification in the academic world.

### ***5.5 Existing Papers***

This section looks at the existing papers which use gamification. There is a focus on papers which use gamification together with software development.

#### **5.3.4 Education**

As can be seen in table 14 Education is represented by 26% of the papers relating to Gamification. This is the largest subject area in which gamification has been discussed. This section of the document gives a brief overview of the use of gamification within education.

Gamification appears a good match for education. There are a number of reasons for this.

Games are built on sound learning principles. Play is an important element of healthy child development (Ginsburg, 2007) Children learn through play. Because digital

games can provide an opportunity for play through simulated environments, these games are not necessarily a distraction from learning, but rather can be an integral part of learning and intellectual development (Ke, F, 2009). Games provide an environment where failure can happen without consequence, allowing learning to happen. However, while it is clear that learning can transfer from one game environment to another, it is not clear that learning from within the game environments will translate to skills outside the environment.

Games provide personalized learning opportunities. As games support the use of levels games can provide students the ability to learn at their own pace and at a level that suits where they are. Games, through the use of levels and permissions, can force students to go through appropriate learning progression, whereas classrooms can result in students missing steps on which future lessons are built.

Games provide more engagement for the learner. Traditional schooling has been often been labeled as boring for many students. In fact, nearly half of high school dropouts said a major reason for dropping out was that the classes weren't interesting, and 70% said they were not motivated or inspired to work hard (Bridgeland, Bilulio, & Morison, 2006). Games contain the pieces necessary to engage students and help them enter a state of flow (Csikszentmihalyi, 1991) where they are fully immersed in their learning environment and energized and focused on the activity they are involved in. When complete attention is devoted to the game, a player may lose track of time and not notice other distractions. Games support many of the components of flow such as clear goals, direct and immediate feedback, balance between ability level and challenge, and sense of control. Naceur and Schiefele, (2005) have shown that student interest was a better predictor than student ability in challenging reading comprehension tasks, and that interest was also related to persistence in reading difficult texts and in long-term retention of reading material.

Games teach 21<sup>st</sup> century skills. Teaching and assessing 21st century skills “frequently requires exposing learners to well-designed complex tasks, affording them the ability to interact with other learners and trained professionals, and providing them with appropriate diagnostic feedback that is seamlessly integrated into the learning experience.” (Rupp M *et al.*, 2010) This is what well-designed games do. Games foster

collaboration, problem-solving, and procedural thinking (Johnson *et al.*,2015) which are important 21st century skills. Current classroom teaching can be focused on teaching skills that are directly testable.

Games provide an Environment for Authentic and Relevant Assessment: Games and traditional assessments share underlying characteristics that provide a means for quantifying knowledge and abilities. The two environments use complimentary technologies that can combine to create more accurate models of student knowledge, skills, and behaviours. In games, the assessment process occurs as the game engine evaluates players' actions and provides immediate feedback. While methodologies have been created for designing games for assessment, there is still a need to provide analytical tools and update the competency models.

In general, the research supports that digital games can facilitate learning, but it is difficult to draw stronger conclusions about the educational impact of digital games at this point because relatively few games have been tested against other teaching and learning approaches (Egenfeldt-Nielsen, 2006). Research should prioritize how games can best be used for learning. (McClarity, K.L, 2012)

In summary, education and gamification appear a good fit. Games and play are recognised as an integral part of learning in children. Games provides a means for assessment and counter negative elements of classroom teaching. The next section looks at gamification and IT.

### **5.3.5 IT**

Gamification and IT are generally combined in two manners. The first is how gamification can improve IT process, while the second is around standardizing the development of the gamification.

Improving the IT process has focused on the fact that many of the processes that are part of software engineering are not appealing to software engineers. Software engineers resist software methodologies designed to improve the overall success of software projects. (Reimenschneider, 2002) Initial works in this space focused on

socializing software development. (Treude, C & Storey, M., 2010) describe using dashboards and feeds to summarize development data which is extracted from the integrated development environment (IDE). In addition to this, tools will provide updates to the developer, therefore socializing the development progress. Leif and Schneider, (2012) extended this to look at a methodology for building in socializing into software engineering. Here they considered gamification as a means to motivate software engineers and looked at the possibility of using it to “augment software engineering methods”. They extended this work in Leif and Schneider, (2012b) where they examined socializing version control. In this experiment, using a leader board and newsfeed, they attempted to improve the use of version control. The aim of the game was to increase the number of commits made by the software teams. Other applications of gamification include: leaderboards for punctuality, Costa, Joao, (2013) and user requirements elicitation, Fernandes, Joao, (2012) There are products in the industry relating to gamification. This includes Bug Fixing and Access Control (Enterprise Gamification, 2015).

Gamification Modeling Language (GaML) is a model language which attempts to provide a language that can be compiled by gamification platforms without the need for software developers. This language provides a mechanism for defining gamification mechanics. (Herzig P *et al.*, 2013)

Having completed the review of the gamification projects the next section focuses on gamification and Agile.

## ***5.6 Gamification and Agile***

This section of the document looks at the use of game elements in Agile methodologies.

In Agile, gamification is often used as part of release planning. This gamification is in the form of “Planning Poker”. Planning Poker was defined by James Greening, (2002), in an effort to resolve the problem of “analysis paralysis” in release planning.

The issue is that release planning requires a high level estimate to give the business an indication of what stories to include in a release and how to prioritise them. However, when faced with the need to produce an estimate, the team spend a significant amount of time discussing the low level detail, and how the story will be implemented, rather than giving an estimate. These discussions are often only of interest to the development team, or sometimes specific members of the development team, with others in the meeting quickly losing interest. Retrieving an estimate from these discussions is difficult as the protagonists are not willing to commit until all the details are known. The release planning session is used to estimate and prioritise many stories. With the slow low-level discussion the team do not get through enough stories to satisfy the customer. “The release-planning objective is to get a ballpark estimate of the effort to build the product, and to split the product into interesting release. Precision of individual estimates is not the goal. Determining the project scope is”. (Greening J, 2002)

Planning Poker is an approach to release planning which attempts to resolve this issue. The steps are as follows:

- The customer reads the story;
- There is an optional discussion clarifying the story;
- Team members consider their estimate and write the value down. They do not discuss their estimate at this stage;
- Players reveal their estimate;
- There is a discussion on the outlier values, whether they are high or low;
- A consensus is reached following the discussion, or the story may be deferred or it can be pushed back to the customer to provide more detail in the next session.

The aim of this approach is to speed up the estimation for release planning and to get the entire team involved as early in the process as possible. In comparing with other estimation methods, Haugen and Ostvald, (2007), found that Planning Poker may be more accurate than unstructured estimates in a group and may restrict developer over optimism when estimating as an individual.

The name “Planning Poker” suggests that it is a gamification of the process. However, there is a need to look at what elements of the process are gaming elements. First, we look at in the framework of a game, and then we will look at game elements which may be considered part of “Planning Poker”

Framework Element	Description
Purpose of the game	The purpose of the game is to reach a shared estimate on a story.
Procedure for action	The players in Planning Poker are the team members. They must provide an estimate.
Rules governing action	The rules are clear and are outlined above. The main rule is the lack of discussion prior to the initial estimate.
Number of required participants	There is no fixed size in Planning Poker. It is simply all members of the team.
Roles of participants	All participants know their role.
Results or pay-off	There is no prize associated with Planning Poker.
Abilities and skills required for action	Each team member has an opinion, but no abilities required to be part of the game. An ability to persuade others can help, but as there is no winner it is not clear that this is a necessary skill.

Table 21: The framework for games as applied to Planning Poker (Avedon, 1981)

So what game elements may be considered part of Planning Poker?

- Time pressure: Although Planning Poker was defined to improve the performance of release planning, there is no definition of how long each stage should take. As a result I would not consider time pressure part of Planning Poker;

- **Community Collaboration:** This is an element of games where the members work in a team. However, in Planning Poker the members are already in a formed team and it is not the game that brings them together. Greening (2002).

Planning Poker has no other elements that could be described as game elements. Based on this, despite the name, planning poker is not a gamified process. Having completed the review gamification and Agile, the next section concludes the gamification literature review.

### ***5.7 Conclusion***

This section provides an overview of gamification. It firstly defines gamification. The initial definition of gamification focused on the mechanics of game elements and placing gamification as a separate field from play, gaming and serious games. However, industry seized on the “perceived benefits” of gamification and started producing applications which focused on leader boards, rewards and badges. This has resulted in a rethink on the definition of gamification. This definition focuses on the experience of the users, rather than a bolt-on application, and ties gamification to motivational studies. The experiment will use the gamification as a bolt-on to an existing application. As part of motivational studies gamification must consider focus on the ethics of changing a person’s behaviour as well as the category of motivation the game element is trying to provide. A key point is that gamification must include setting the comfort level of members in the workplace to play the game. The chapter then looks at game elements that could be used in the experiment and then examines how gamification has been examined in academia. The highest number of papers relate to education and IT, so the section focuses on these. Finally, gamification and Agile are examined, specifically, “Planning Poker”. The review concludes that it is not a gamified process.



## 6 EXPERIMENT OVERVIEW

This chapter of the document provides an overview of the experiment.

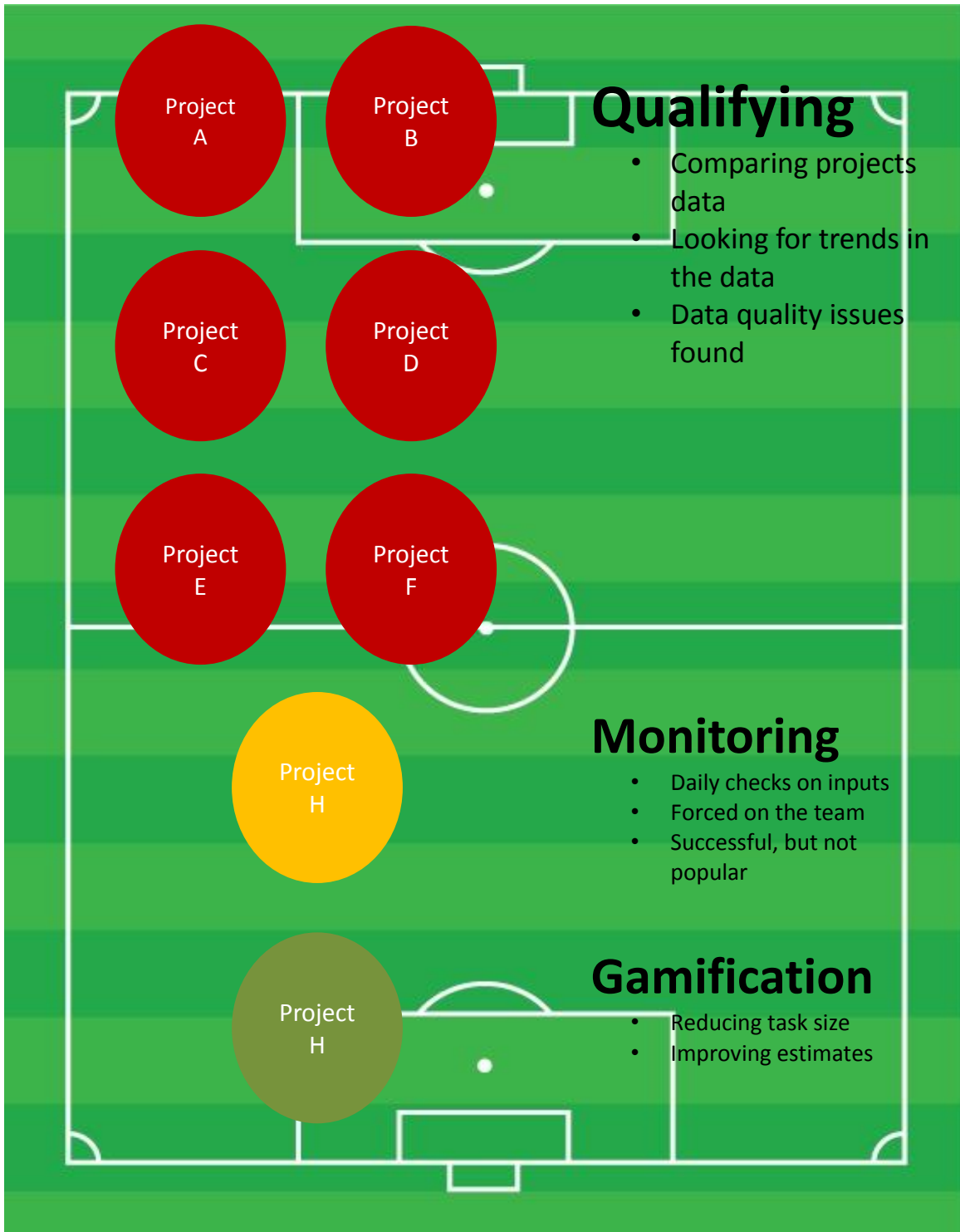


Figure 11: Experiment Infographic

As part of the experiment a number of projects were selected for analysis. The data was extracted and put through a data preparation phase. The intention was then to analyse this data and search for trends in the data. However, following the data preparation, issues with the quality of the data were identified. The result was a decision to introduce monitoring to the experiment.

Monitoring was implemented by capturing the previous days updates and discussing them in the daily scrum. The capture was a manual process, with the capture happening after 5pm every evening during the iteration. This late capture ensured that all team members had ample time to enter their tracking updates. The monitored data gave the team opportunity to discuss the updates, which led to many insights into the progress of the iteration.

The final stage of the experiment was the gamification experiment. This added a lottery element to the existing iteration processes. As part of the process the team were told there would be a reward related to the completed tasks. This would be lottery based, but the chance of receiving the reward increased with the number of tasks completed.

## **7 HISTORICAL ITERATIONS**

### ***7.1 Introduction***

This section of the document outlines the approach taken to establish the baseline for the project. The purpose of the baseline was to produce metrics which could be analysed and used as part of the gamification experiment. This section looks at;

- The methodology used to capture baseline data;
- The results of that data;
- Supplementary data captured following analysis of the results.

### ***7.2 Experimentation***

This section of the document examines the methodology to capture the initial project baseline. The section looks at a number of key areas of the project:

- How the projects used for analysis were identified;
- What metrics were captured;
- The method used to capture each metric;
- The additional factors that could impact on the project.

#### **6.2.1 Identifying Projects**

This section looks at the process to identify candidate projects. There were a number of factors which impacted on project selection:

- **Project Duration:** In order for consideration, the project must have a number of iterations where the team is consistent. All iterations will vary; company and team member holidays; people moving in and out of projects; working hours and business demand on the team. However, for the purpose of this analysis, projects had to be consistent. So a period of six iterations each consisting of a period of 15 working days was identified;
- **Project Data:** All project data must be available in a single ALM tool. This would allow consistent capture of data for each project. If multiple tools were in use, the data available might not be comparable and the project would have

to focus on how to combine the details. Focusing on a single ALM tool allowed the effort of the project to be focused on the measures that the data revealed rather than the effort to combine the data;

- **Team Lead Availability:** As part of the analysis it was necessary to discuss the project and iteration and factors that may have impacted on the project. As a result, in order for the project to be available for selection, the team leadership or Scrum Master must have been available to discuss the project.

This section explains the approach taken in selecting projects. Having identified the projects to capture data, the next step was to identify which metrics needed to be captured for each project. The next section identifies the metrics that were captured.

### **6.2.2 Metrics to capture**

In this section the metrics to capture are identified. The section first looks at the candidate metrics, then the metrics that could be achieved with the existing project data.

#### **6.2.2.1 Candidate metrics**

The candidate metrics were based on the literature review. Javdani T *et al.*, (2013), identified the following as the key metrics for measuring Agile project:

- **Estimate versus Actual:** A comparison between the estimated duration of “stories and defects” and the actuals completed. In Agile, at the beginning of an iteration, the team provides estimates of the work to deliver user stories. This helps the team to decide what to take into the iteration. Therefore, the precision of the estimates is important to the success of the iteration and project. This metric is used to measure the difference between the estimated and actual effort for stories. It can be used by the team to inform future iterations to recognise task types not identified and misunderstood requirements;
- **Velocity:** This is a measure of the team’s activity. The velocity is the number of units of work that the team achieves in an interval of time. The velocity can be the number of days or number of stories or the ideal days. To be an effective measure of the team’s progress in a project the membership must be consistent;
- **Rework:** This is the amount of time spent on defects compared to the amount of time spent on stories. This rework is focused on corrective rework, “Rework to

fix defects discovered in the current version and previous versions during reviews, tests, and demonstrations of the current version”. (Fairley, R.E. & Willshire, M.J. ,2005) The amount of corrective rework can be indicative of the quality of the work being produced.

- **Burndown:** This is used to chart the progress of a project, within an iteration or a release. The burndown of work is determined by comparing the work remaining in the interval period against an ideal burndown. The ideal burndown is calculated as the total work in a time interval divided by the number of intervals. This is then multiplied by the number of intervals that have passed. The result is graphed as a combination chart of columns and line graphs. The chart makes it easy to see progress in the interval and allows the team to react by removing work or supplementing the team. An alternative to the burndown is the burn-up chart. The burn-up chart focuses on work completed rather than work to do;
- **Cumulative Flow:** This is an extension of the iteration burndown and shows the quantity of work in a given state. The diagram is an area chart in which each area is associated with a state of development. The cumulative flow is useful to establish bottlenecks in the project, as it is possible to see the durations of story by state;
- **Earned Business Value:** This metric was devised as an Agile alternative to the Earned Value Analysis (EVA) used in standard project management. EVA is the combination of Budgeted Cost of Work Scheduled (BCWS); Actual Cost of Work performed (ACWP) and Budgeted Cost of Work Performed (BCWP). In Agile we are not in a position to calculate the BCWS and the BCWP, as they require detailed up-front. To replace EVA it is necessary to understand its purpose. Management is looking for information about the value the product is providing and what percentage of the product is complete. EBV is calculated by developing a work breakdown structure of the project, giving a big picture of the project. Then each leaf node, which represents collection of stories or features, is assigned a weighting. Stories within the Work Breakdown Structure leaf node are given individual weighting. The weightings represent the business perception of value of completing the tasks. Percentages are then calculated for each bucket based on the weighting which is compared with the

other buckets in the same grouping. The EBV is calculated by summarizing the total percentage of work done as stories are completed; (Rawsthorne, D)

- Total Effort Estimation: This is taken at the beginning of the project and represents an estimate for the entire project rather than the more accurate estimates on an iteration by iteration. It is based on the number of iterations that are anticipated to be in the project and the size of the team. It becomes difficult to calculate when the team members work in a part-time nature or the project requirements are not clear in advance. (Javdani T *et al.*, 2013)

This section describes the common Agile metrics. The next section evaluates them for inclusion in or exclusion from the project.

#### 6.2.2.2 Metric selection

This section of the document outlines the reasons for inclusion and exclusion of metrics as part of the initial review of the project data. The following metrics were included in the project experiment:

- Estimate Versus Actual: This metric was identified as a key metric in this experiment. One option was to evaluate the use of gamification for improving estimating. The data available includes both actual and estimated data, so this metric was included;
- Velocity: This is a means of measuring the rate of work done. This is of interest to most teams. The data requires a consistent measure of velocity. In order to make this consistent across projects, an ideal day of six hours was identified as the measure of velocity;
- Rework: Reducing the number of defects is a key incentive for all project teams. The time spent on corrective rework is a drag on project resources. To establish this we need to identify defects raised against stories in the iteration. This data was available so was included in the project;
- Burndown: The Iteration Burndown allows the team to identify when projects are slipping. To establish the Iteration Burndown, we need the day over day actuals to determine what was done each day, as well as the total capacity and total estimated work taken on. As these were available or could be calculated this metric was included in the project.

The following metrics were excluded from the project:

- **Cumulative Flow:** This is an extension of the Iteration Burndown, which includes state information. In order to calculate this we need to have the state data together with the timing of when the state changed. This data was not available so this metric was not included in the initial baseline data;
- **Earned Business Value:** This is a business metric which requires weighting values being applied to the project features and stories. These weightings would not be consistent across different projects, so this metric was not included as part of the analysis;
- **Total Effort Estimation:** This metric is applied at the start of projects and is used to make a decision on whether the project should proceed or not. This will vary from project to project, so this metric was not included as part of the analysis.

This section described metrics which were included and excluded in the experiment. The next section will examine how each of the metrics were calculated.

#### 6.2.2.3 Calculation method

This section of the document examines how the metrics were created. The section reviews the data source, the tools used to extract the data; the filtering applied to clean the data and the method used to generate the charts.

##### **Data Extraction**

This section of the document looks at how the data was extracted. The data is stored in Rally a cloud-based platform that is used to run their development lifecycle (Rally, About). In Rally, the teams enter their capacity for an iteration. They also enter estimates and actuals at task level. These tasks and figures are rolled up to the user story level. The data stored in Rally is not available in a local database so it has to be extracted. The extraction method provided by Rally is to use an Add-On to extract data to Excel. (Rally Add-On link)

For each of the projects under review, the following data was extracted:

- **Story level data for iterations.** This is used in calculating the actual versus estimates, velocity and rework metrics;
- **Defect level data for iterations.** This is used in calculating the actual versus estimates, velocity and rework metrics;

- Task level data for iterations. This is used in calculating the actual versus estimates, velocity and rework metrics;
- Capacity: The capacity for the iteration.

The full list of fields extracted is available in Appendix A.

Having extracted the data the next step was to prepare the data for use. The next section outlines the data preparation that is required to make the data usable and comparable.

### **Preparing Data**

In this section the steps required to transform the data into a format that was usable are described. The data preparation was different between story level data and task level data.

#### *Story Level Data*

This data was prepared for story and defect level data. The first step was the combination of data. The data for use in estimates and actuals; velocity and rework required that the data extracted for defects be combined with the data extracted for stories. Combining the data was not difficult as it was in the same format. It was a manual task that had to be done for each project.

Once the data had been combined there was a need to remove data that was not usable. For the metrics based on story and defect level data this included the following issues:

- Data not assigned to an iteration. In this instance the data was assigned to a project, but not associated with an iteration. This data was filtered from the project as the data needed to be assigned to an iteration in order to be included in analysis by iteration;
- No actuals presented: Although the story was marked as complete, the actuals were null or blank. Stories where the actuals were NULL were not considered part of the analysis;
- No estimate presented: In some cases the data did not include estimated data, only actuals. In this instance it was deemed better to exclude this data from the calculation;



Having combined and filtered data, the next step was to normalize the data using calculations. The calculations varied based on the metrics being produced. For estimates versus actuals, velocity and rework the following calculations were used:

- Estimate Story Point: This was calculated on the estimated column, unless it was blank or zero, in which case the actuals were used. The value from this formula was then converted into ideal days by dividing the hourly total by six. A round up to an integer value was then applied;
- Actual Story Point: This was calculated on the actual column. Again it was converted into days by dividing by size. Finally a round-up was used to obtain an integer value;
- Story Hours: this was calculated as any actuals associated with a record which has “Story” in the formatted id;
- Defect Hours: this was calculated as any actuals associated with a record which has “Defect” in the story id;
- Work Efficiency: This is the total of defect hours presented as a percentage of story hours.

#### *Task Level Data*

The task level data was not filtered, so the Iteration Burndown charts were produced using calculations only. For the data in use in generating the Iteration Burndown, there were a number of calculations required.

- Formatted Creation Date: This is the creation date, converted into a format that could be used in future calculations;
- Formatted Last Update Date: This is the task last updated date, converted into a format that could be used in future calculations;
- Calendar Day Completed: This is the number of calendar days that have passed between creation date and the last update date;
- Iteration Day Completed: This is the number of iteration days that have passed between creation date and last update date. It is calculated by excluding weekends from the calendar dates;
- Actuals plus To Do: This is calculated as the actual data plus the To Do data. The purpose is to calculate the total task time. This will be used to calculate the total burndown;

- Iteration Day: This is the number of the iteration day, one to 15 for a three week iteration;
- Actual Burndown: This is the burndown for each day in the iteration. It is calculated by summarizing the Actual plus To Do where the iteration day completed is on or before the iteration day;
- Ideal Burndown: This is the total estimated values less the average daily estimated values by the number of days that have passed.

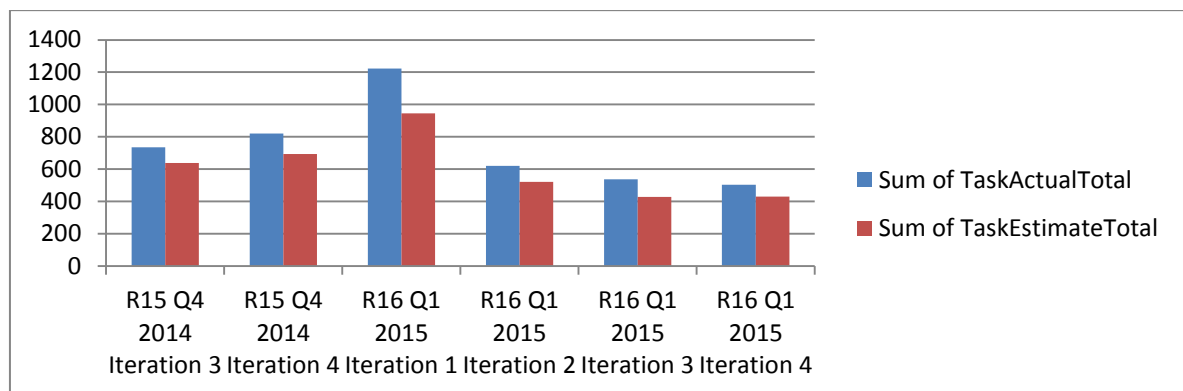
This section has described how the data has been prepared. The next section describes how the charts were produced from the data.

### Visualizing Data

This section looks at how the charts were produced. It describes the type of chart and the variables that were used to produce the chart:

#### *Estimates versus Actual*

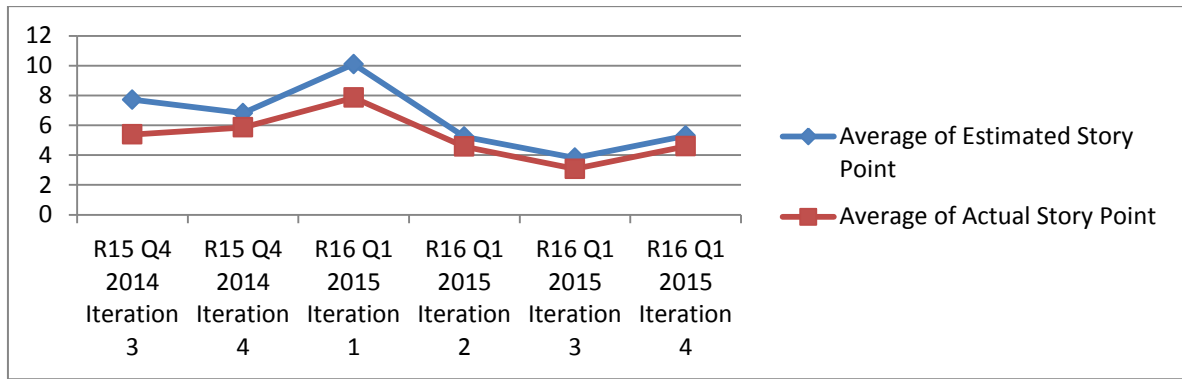
This is a column chart which consists of two series; the estimates and the actuals. The iterations included in the analysis are displayed across the X-Axis. The y-Axis is the number of hours estimated or actual.



**Figure 12: Estimates versus Actuals sample**

#### *Velocity*

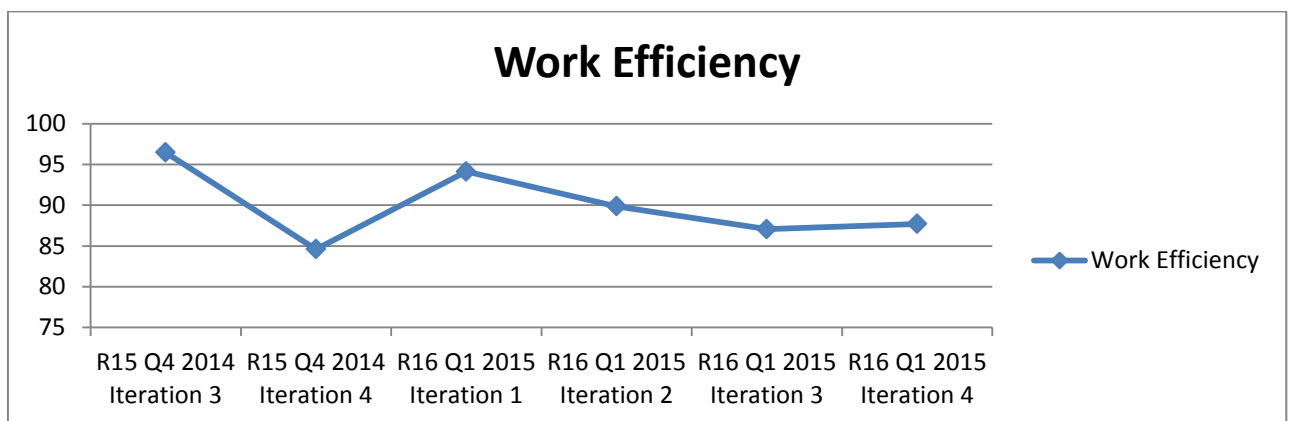
This is a line chart which consists of two series, the estimates and the actuals. The iterations included in the analysis are displayed across the X-Axis. The y-Axis is the number of story points estimated or actual.



**Figure 13: Velocity sample**

*Rework*

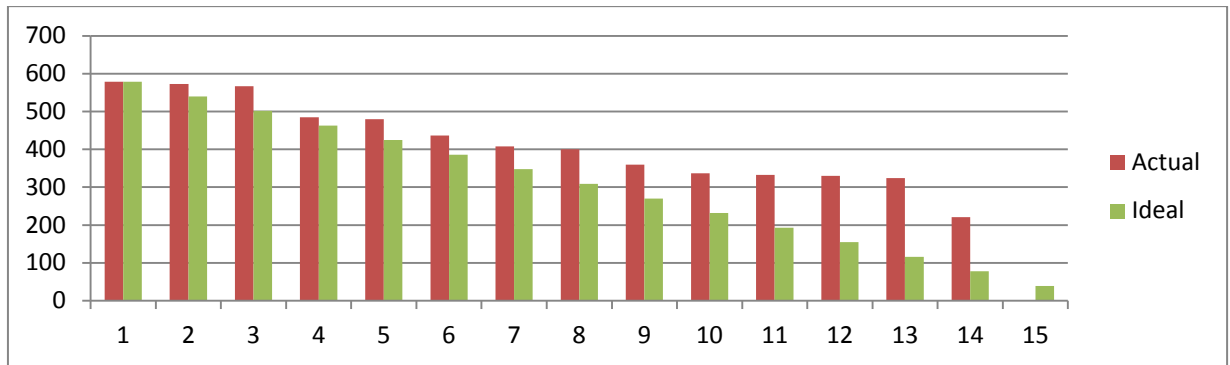
This is a line chart which consists of one series, the rework as a percentage of stories completed. The iterations included in the analysis are displayed across the X-Axis. The y-Axis is the work efficiency percentage.



**Figure 14: Rework sample**

*Iteration Burndown*

The Iteration Burndown is displayed as a column chart, with two series; one for actual and the other for ideal burndown. The X-Axis shows the day of the iteration. The Y-Axis shows the number of hours.



**Figure 15: Iteration Burndown sample**

This section shows the initial charts and describes how they were produced. This completes the section on calculating the initial metrics. The next section looks at how this data was supplemented.

#### 6.2.2.4 Supplemental Data

This section of the document examines the reasons for producing supplementary data in addition to the estimate versus actual, velocity, rework and Iteration Burndown. It then explains how each of the supplementary data was retrieved.

#### Reasons

This subsection outlines the reasons for the addition of supplementary data. They are as follows:

- The Iteration Burndown is not easily comparable across projects. The charts produced are created at an iteration level rather than a project level. However, when looking for trends in the data, the data needed to be at the same level as the other charts. There is a need to combine the iteration data into a single graph;
- The data being compared is across teams and times. There was a need to identify factors which could impact on the iteration. These additional factors could be used in the analysis to explain differences;

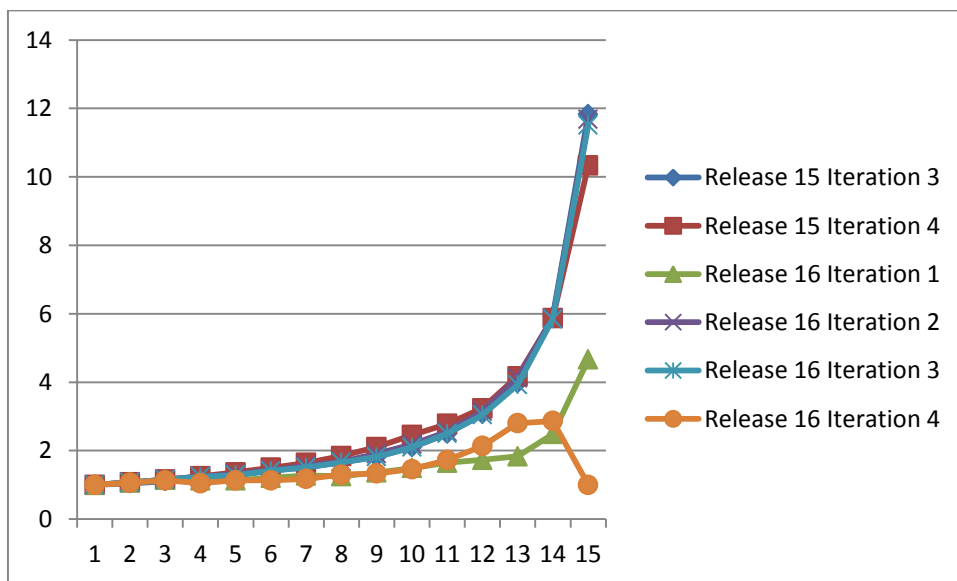
Having reviewed the reasons for additional data, the next section looks at how the first reason, comparable Iteration Burndown, has been actioned.

### Comparable Iteration Burndown

This section examines how the Iteration Burndown was presented at the project level rather than the iteration level. The following are the steps that were required:

Calculate the average difference between the ideal and actual burndown for each iteration. If the actual outstanding is zero, the average is set to one. This was done for each day of the iteration. The averages were then amalgamated into a single set of data. The chart is then produced.

The chart is a line chart which has the iteration day as the X-Axis, the Y-Axis shows the average value. The chart will have a series for every iteration in the project. Ideally, the series should be a straight line with a value of 1, indicating no variance from the ideal.



**Figure 16: Iteration Burndown across projects sample**

This section describes how the project level Iteration Burndown was created. The next section examines the remaining supplementary data and how it has been captured.

## **Other Context Data**

This section examines the data used to provide context on the iterations. It explains why it was captured, and how this data was extracted. The following is the extra data that was captured

- Metrics for holidays;
- Metrics for # of users;
- Metrics for time zones.

### *Metrics for holidays*

Projects run over many months, there will be a number of holidays impacting the project. This will include team member vacation days and corporate holidays. In addition to this, in a global operation there will be global corporate holidays on different dates. For example, in China New Year is a major holiday that falls in February of each year, while in Ireland the equivalent holiday is the period from Christmas to New Year. For this reason, the holidays need to be monitored when examining metrics for iteration. The variety in the results may be explained by the holiday plans of the team member.

To capture the holiday dates, the national holidays for each country were downloaded from the internet.

### *Metrics for # of users*

The number and make up of users in an Agile iteration should be constant. However, in some instances the team makeup changes during the course of a project. People leave teams, moving onto different project, moving out of the company. Project priorities change, resulting in enlarged or reduced team sizes, either temporarily or permanently. This will have an impact on the team performance and the metrics which reflect this. In an iteration, the users are presented in three different ways:

- Capacity: This is amount of time that the user commits to the project during the course of the iteration. If this varies, it indicates that the team size is not consistent. This is captured manually from the Rally tool;
- Estimates: This is the amount of work that the user has estimated. Each user in the project should have estimated work, so the number of users who have capacity should match the number of users who have estimates. If this number

is more than the capacity it could be an indication of team members being added to the project during the iteration. If this number is less than the capacity it could be an indication of a team being taken off the project. This is captured manually from the Rally tool;

- Actuals: this is the amount of work that each user has completed. The number of users with actuals should match those with estimation. Differences are indicative of team members being added or withdrawn from a team. This is captured manually from the Rally tool.

### *Metrics for time zones*

In a global project, team members work in different time zones. If the team are sufficiently separated, the time zone can impact on the amount of time available for communication. In addition to this, not all countries use daylight saving time during summer months. In such situations, the change to and from daylight saving time can impact the overlap between the teams in diverse locations. The limited time may have an impact on the iteration communication. This data was captured manually.

This section has provided focus on the other data used to provide context. The section detailed the type of data and how it was captured. The methodology section has described how the data was captured for the initial analysis. The next step is to describe the results of this analysis.

## **7.3 Evaluation**

This section of the document examines the results from the initial analysis. The following are the key areas of the section;

- The projects being analysed;
- The data filtered from the projects;
- The results for the metrics;
- The context data supporting the analysis.

### 7.3.1 Projects

This section of the document gives a description of the projects in use. The section first introduces the high level projects and then describes the characteristics of the project. The next section describes the outcome of the interview with the Scrum Masters. The different approach to managing the projects is then outlined. The final subsection, subdivides the projects into major releases.

Analysis for the thesis project will be based on two major projects broken into 3 sub-projects. The first major project is a financial cost allocation data warehouse, while the second major project is a financial statistics data warehouse.

Characteristic	Financial Statistics	Cost Allocation
Type of project	Data Warehouse	Data Warehouse
Volume of data	Terabytes	Terabytes
Complexity	Contains standard data warehouse dimensions and facts. Also includes a complex allocation engine for funds.	Contains standard data warehouse dimensions and facts. Also includes a complex allocation engine for costs and revenues.
Team Structure	Business in US; Leadership in Ireland; Development in China.	Business in US; Leadership in Ireland; Development in China.
Team Size	Medium	Medium
Current Status	In production, adding addition features.	In production, adding addition features.
Agile Methodology	Scrum	Scrum

**Table 22: Characteristics of the major projects**

As can be seen by the listed characteristics the two projects are very similar in their structure and content.

#### 7.3.1.1 Financial Statistics Data Warehouse

Having examined the characteristics of the project, there followed an interview with the projects Scrum Master. The aim of this interview was to establish the difficulties



that the team was facing. The full transcript of the interview is available in Appendix B. The following are the areas that the team has struggled with:

- **Business Knowledge:** The development and QA teams are largely separated from the Business. The team interface through the BSA to get an understanding of the business requirements. However, there is a 12 hour time-difference between the locations of these teams. As a result the team tends to rely on interpretation of requirements from the technical lead. The technical lead has overlap with both teams;
- **Delivery:** The number of defects and the time taken to complete delivery remains a concern for the team and the management. In the past the team has focused on the delivery date over quality. The issue has been that with the time zone limiting overlaps, issues can take a lot of time to resolve, even when the fix was relatively straight-forward. An on-going effort has been made to reinforce quality as the key driver;
- **Metrics:** As part of the effort to understand the teams issues, it was determined that metrics should be captured on the iterations. However, the information being entered into the iteration progress tracking tool is not reflective of the team's effort. In particular the actuals completed do not reflect the time put in by the team. The team have often put in overtime but the tool indicates that the team are operating below capacity;
- **Estimation:** The team's estimates are not tying closely with the actuals. This is in part due to the waterfall approach used by the team, where issues found in the higher environments take a long time to resolve as they pass back to development.

The methodology for this group is based on Scrum. The team uses a number of the key components of the Scrum methodology, although some are renamed.

Component	Team Component Name	Description
Product Backlog	Release Planning	The release planning tends to be based on limited descriptions of the requirement. As the team is global, the release planning is not inclusive of all team members, technology is

		represented by technical leads rather than those who develop the story.
Sprint Backlog	Release Backlog	This is the list of components that are ready for the team to start work on. This work includes the technical design. Items in this list only appear after the BSA team has completed the business design and the business have approved it for development.
Sprint planning	Backlog Grooming	This is managed as a weekly meeting. The product owner will review the list of items which are past business approval and review them. Currently the team does not use story points, so the ordering is based purely on the business needs. The team does take the list into the first two days of the iteration, and focuses on planning. The team will strive to get as much of the prioritized work completed and will declare the stories and defects that they can bring into the iteration.
Sprint Execution	Iteration	The iteration is fifteen days in length. The iteration includes all development, QA and deployment effort to move the user story to the UAT environment where it is ready for UAT Testing.
Daily Meeting	Daily Scrum	As the team is global in nature,

		there are two Scrum meetings. The development team has a daily Scrum in which tasks are reviewed, including updates from overnight issues. There is then a second Scrum in which the full Development, QA and Deployment teams participate. In this Scrum each member describes the work they have completed, the work they plan to complete and any blocks that could hamper them in their workload.
Sprint Review	Retrospective	Due to time constraints, the team at each site conduct a retrospective. The Development and local QA team have a retrospective and then the Technical leadership, global QA team and deployment teams have a retrospective.
N/A	Demo	This is where the team demonstrate the work that they have completed to a representative of the business.
N/A	Iteration Close and committal	As part of the iteration close, the Scrum Master will discuss the previous iteration and the upcoming iteration with the product owner. The aim of this meeting is to describe the successes and issues of the iteration, and give an explanation of how they happened. As part of the meeting the Scrum Master will then describe the

		<p>stories and defects being taken into the next iteration. Any concerns that the development team have raised are passed on to the product owner.</p>
--	--	--

**Table 23: This shows the Scrum element, together with its equivalent from statistics project. A description of the component is given which focuses on describing the activity in the project.**

This section has highlighted the issues and methodology for the Financial Statistics team. The next section repeats the analysis for the Financial Cost Allocation team.

### 7.3.1.2 Financial Cost Allocation Data Warehouse

This section describes the outcome of the interview with the project’s Scrum Master. The section then details the Scrum methodology in use.

The aim of this interview was to establish the difficulties that the team was facing. The full transcript of the interview is available in Appendix B. The following are the areas that the team has struggled with:

- Team are only beginning to examine how metrics could be used to improve performance. Current process is very manual;
- Self-Organization: The team struggles to organize themselves in the event of an issue. This can result in some team members being overloaded and others with nothing to do. The team are not focused on the committal for the iteration, only their own work;
- Not self-sufficient: The team are reluctant to try to solve issues. They look for guidance immediately rather than trying to resolve issues themselves. They need training in how to troubleshoot issues.

The methodology for this group is based on Scrum. The team uses a number of the key components of the Scrum methodology, although some are renamed.

Component	Team Component Name	Description
Product Backlog	Product Backlog	This is the list of stories which the team has to work on. These stories will be prioritized by the business.
Sprint Backlog	Iteration Refinement	This is a meeting in which the team discuss the stories and determine the effort. The team will raise questions on the story, and highlight items that they need to commit to doing the story. The team will use Planning Poker to provide high level estimates.
Sprint planning	Iteration Planning Meeting	In this meeting the team establish what they are going to do in the upcoming iteration. They will look at the effort and ensure that all elements that they need to be able to complete the story are fully available.
Sprint Execution	Iteration	The iteration is based around a three week period, so 15 work days. This includes all development and QA work. It also includes the effort to deploy the stories to development, sit and UAT.
Daily Meeting	Daily Scrum	As the team is global in nature, there are two Scrum Meetings. The development team has a daily Scrum in which tasks are reviewed, including updates from overnight issues. There is then a second Scrum in which the full Development and QA teams

		participate. In this Scrum each member describes the work they have completed, the work they plan to complete and any blocks that could hamper them in their workload.
Sprint Review	Retrospective	The team uses a shared time to meet and discuss the previous iteration. In this meeting the team are looking for what went well and what did not go as well as they hoped.
	Demo	This is where the team demonstrate the work that they have completed to a representative of the business.

**Table 24: This shows the Scrum element, together with its equivalent from project. A description of the component is given which focuses on describing the activity in the project.**

There are differences in the projects, in both the characteristics and the Scrum methodologies. However, these differences are not significant enough to suggest that the projects are not comparable. To provide a meaningful comparison, it is necessary to decompose the projects into their major releases. As part of this, generic names have been applied to the project. This will make comparisons easier in the produced graphs. The following is the breakdown of the projects:

- Project A: This is a financial cost allocation data warehouse. This project related to a new business area being added to the existing warehouse. The project ran from June 2014 to October 2014. The team was global with customers in the US, technical leadership based in Ireland and development work completed in India;
- Project B: This is a financial cost allocation data warehouse. This project related to a new business area being added to the existing warehouse. The

project ran from October 2014 to April 2015. The team was global with customers in the US, technical leadership based in Ireland and development work completed in India;

- Project C: This is a financial cost allocation data warehouse. This project related to ongoing business enhancements. The project ran from October 2014 to April 2015. The team was global with customers in the US, technical leadership based in Ireland and development work completed in India and China;
- Project D: This is a financial statistics data warehouse. This project related to creating additional features to allow for business adoption of the existing warehouse. The project ran from January 2014 to June 2014. The team was global with customers in the US, technical leadership based in Ireland and development work completed in India;
- Project E: This is a financial statistics data warehouse. This project related to creating additional features to allow for business adoption of the existing warehouse. The project ran from July 2014 to November 2014. The team was global with customers in the US, technical leadership based in Ireland and development work completed in China;
- Project F: This is a financial statistics data warehouse. This project related to creating additional features to allow retirement of other existing reporting systems. The project ran from November 2014 to March 2015. The team was global with customers in the US, technical leadership based in Ireland and development work completed in China.

### 7.3.2 Data Filtered

This section of the document describes the data filtered from the projects as part of the process of producing the charts. The data shown is summary data with the actual data in Appendix D.

Project	Unassigned Items	Mean Items		StDev Items		Mean Items		StdDev	
		with Estimates	no Estimates	with Estimates	no Estimates	with Actuals	no Actuals	Items with Actuals	with Actuals
Project A	9	7.17		7.083		5.50		3.619	

Project B	21	3.83	3.545	4.00	5.692
Project C	19	3.33	2.733	1.33	1.366
Project D	22	0.33	0.516	0.83	0.983
Project E	47	2.00	1.549	1.50	1.049
Project F	45	19.17	10.797	7.67	3.327

**Table 25: Summary of data filtered, showing unassigned; missing estimates and actuals**

The summary shows items, which are stories or defects, which are unassigned. This means that the item is not connected to an iteration. The table also shows the mean and standard deviation for the number of items which are missing estimates or actuals.

	# of Stories	# of Defects	Available Work Units	Actual Work Units	%
Project A	129	39	168	108	64%
Project B	90	28	118	66	56%
Project C	65	33	98	58	59%
Project D	74	45	119	50	42%
Project E	52	31	83	24	29%
Project F	165	126	291	128	44%

**Table 26: Summary view of the project data. The actual that is usable for the charting and comparison is significantly reduced from the original data.**

This section has highlighted the filtering of data prior to comparison across the projects. The next section displays the resulting graphs used for comparison.



### 7.3.3 Metrics Results

This section of the document displays the metric results

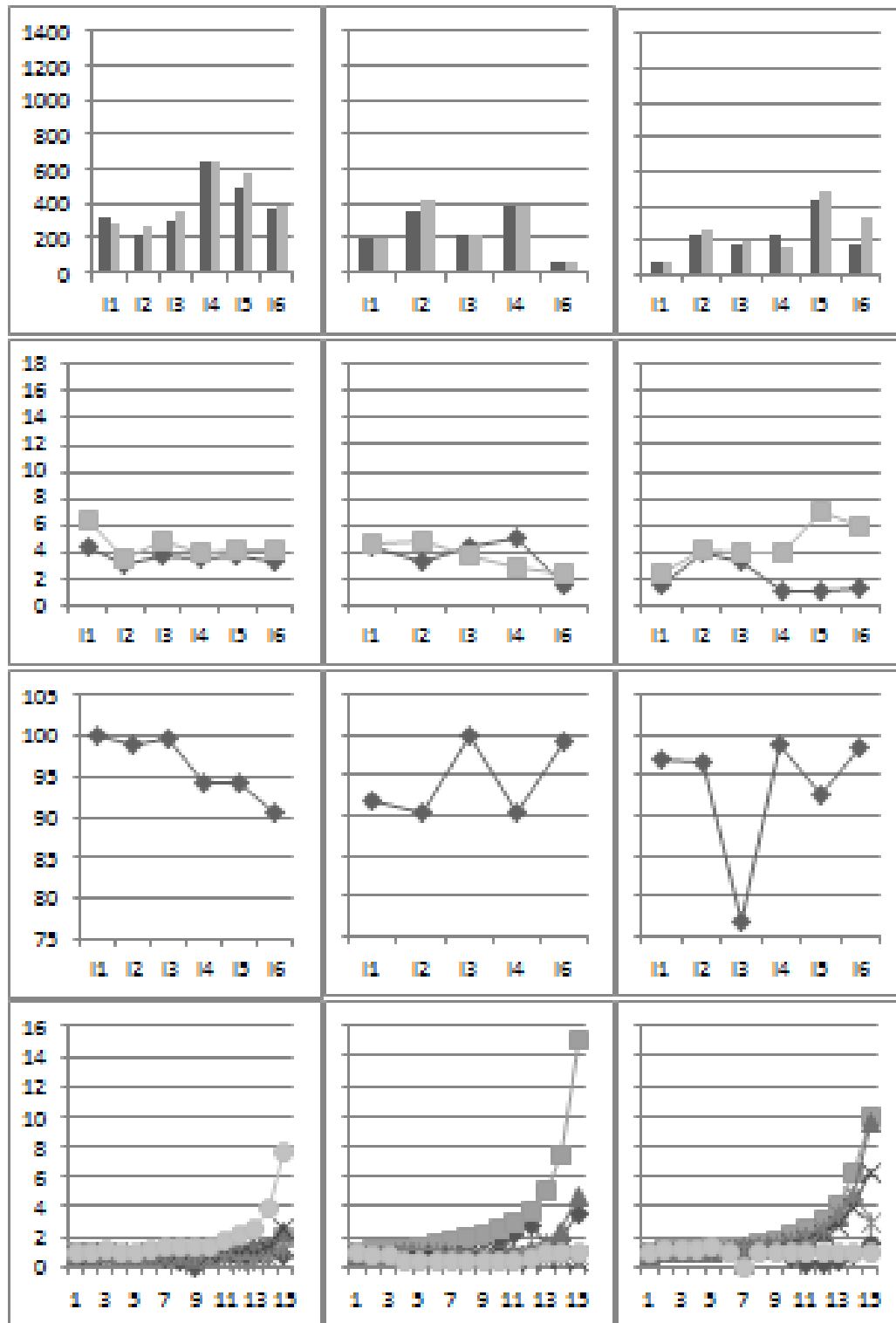


Figure 17: Shows the Actual versus Estimate; Velocity; Rework and Iteration Burndown for Projects A, B and C.

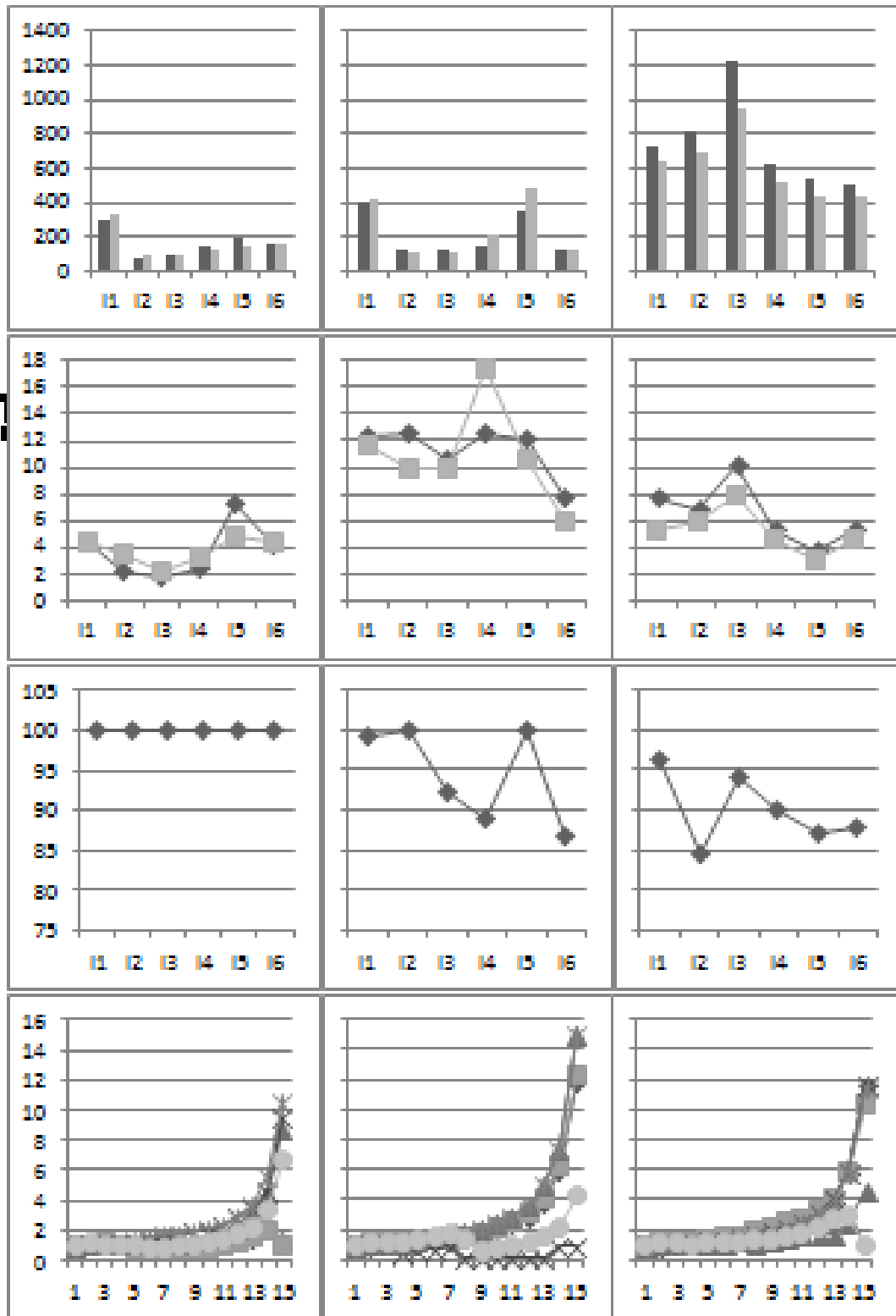


Figure 18: The Estimates versus Actuals; Velocity; Rework and Iteration Burndown for Projects D, E and F;

The data in figure 17 and 18 show four main graphs used for comparison across the project. The top line of graphs displays the actuals versus estimates. The Y-Axis is the number of hours, while X-Axis represents the iterations. The next line displays the velocity. The Y-Axis is the velocity in points, while the X-Axis represents the iterations. The third set of graphics displays the rework. The Y-Axis represents the percentage of effort doing actual development, in the selected projects this ranges from 75% to 100%. The X-Axis shows the iterations. The final line displays the combined Iteration Burndown. The Y-Axis represents the range of variance from the ideal Burndown. If the iteration was at the ideal then the line would be represented as a straight line with the Y-Axis value of 1. The X-Axis is the day of the iteration. The data highlights some anomalies, for instance in project D defects were recorded but no time (actuals or estimates) were captured for them. Another anomaly was project B when one iteration was not captured in the tool. This section has shown the data used for comparison, before discussing the data, the next section is used to highlight the context data that may have an impact of the data.

### 7.3.4 Context data

This section of the document presents the data captured to provide additional context for the iterations. The data has been summarized to present in a single table. The full set of data, detailed at the iteration level is available in Appendix C.

Metric	Project A	Project B	Project C	Project D	Project E	Project F
Capacity	N/A	N/A	11.50 (5.992)	10.50 (2.811)	10.50 (1.643)	16.50 (2.588)
Estimates	16.33 (3.502)	10.17 (7.223)	13.83 (2.137)	13.67 (3.559)	8.50 (4.506)	17.00 (2.000)
Actuals	15.17 (3.545)	9.33 (5.888)	13.17 (1.722)	12.33 (2.805)	7.83 (3.764)	16.50 (2.258)
Time zones	7.00 (0.000)	8.00 (0.000)	7.67 (0.516)	7.33 (0.516)	7.83 (0.408)	7.00 (0.000)
Dev Team Holidays	1.00 (2.000)	0.67 (1.211)	0.83 (1.169)	0.50 (0.548)	1.00 (2.000)	0.50 (1.225)
Tech Lead Holidays	0.17 (0.408)	0.67 (0.816)	0.67 (0.816)	0.67 (0.516)	0.33 (0.516)	0.33 (0.516)

**Table 27: Summarized context data for the historical iterations. The mean and (stddev) are shown**

Capacity is N/A for Project A and B as capacity was not entered. This section of the document has described the additional metrics needed to provide context to the iteration data. The next section evaluates the data.

### 7.3.5 Discussion

This section of the document reviews the data outlined above and discusses the impact to the project.

The first point of interest was the number of stories and defects without an iteration. These stories and defects were created in the iteration, but not assigned to the iteration. Although some requirements could have been dropped when, for example, they had estimates assigned which were too high, the number of the stories was very high. As can be seen in table 25, the number of stories and defects in a project without an iteration was on average as high a full iteration's work. It is likely that many of these work items were not correctly set in the application tool. This was put to the Scrum Masters of the projects, and there was a general concurrence that this was the case. This would suggest that the task of updating the metrics was not appealing to the team members. This kind of manual work is not motivating for the team members, it

would be seen as boring work, which is a de-motivating factor (Sharp, H *et al.*, 2009). It is hoped that by actively monitoring the iteration and highlighting when the data is not recorded, that the actuals and estimates will be more accurate for measurement.

The next point is the quality of the data within the iterations selected. As table 25 highlights, there was a significant amount of data excluded because it did not contain both actuals and estimates. While the lack of actuals might point to a number of stories which were dropped in mid iteration, the missing estimates would suggest that the tool was not being used to accurately track the iteration. While the team would be recording stories and defects in the iteration, the daily work effort does not seem to have been tracked. Table 26 summarizes this information, showing that at a project level Project E had the least amount of usable data at 29% while Project A had the most at 64%. Even Project A had only two thirds of the data available for analysis. This was put to the Scrum Master and the following responses elicited:

“At one stage in the project, the process was largely abandoned in pursuit of a couple of key stories. The team were so busy working these stories that it was decided not to pursue them in relation to the Rally updates. The decision to focus on metrics was taken in January and this led to an increased awareness in tracking times through the tool”.

So without the external pressure the team did not keep the project tracking information up to date. The team were more involved in the “technically challenging work” (Sharp, H *et al.*, 2009) of the key stories and this gave them licence to ignore the more menial tracking task.

The project analysis across the selected metrics did not identify a consistent trend in the projects. As shown in figures 17 and 18, projects A, B and C appeared more successful in the Iteration Burndown, in that more of the iterations resulted in the “To Do” completing in line with the ideal burndown. However, there was no obvious trend in actuals versus estimates, velocity or rework. The actuals versus estimates and velocity both show that the actuals and estimates are quite closely matched, though they vary significantly from iteration to iteration. This would suggest that the team structure and workload differed significantly across iterations, or that overtime efforts are being made to develop key requirements. A principal of Agile is that the team develop at a sustainable pace (Kent, 2001), but this does not appear to be the case in

these projects, Also, in Agile “small, collaborative teams of developers are able to share tacit knowledge about development processes”. (Schwaber, K, 1997) The amount of work committed to and completed varies together with the team size, (see table 27), from iteration to iteration. This is in conflict with the understanding of small teams envisioned with Agile, and necessary to build a consistent velocity. By monitoring the iteration, we will ensure that the data reflects correctly the state of the project and therefore will be able to assess the outcome of the iteration with confidence.

The amount of rework looks good. Iteration 3 of Project C was the lowest value for the projects. It dropped as low as 75% of work on the defects. However, the data for defects is particularly poor, with project D recording no usable defect data.

	Project A	Project B	Project C	Project D	Project E	Project F
Defects	39	28	33	45	31	126
Excluded	35	12	20	42	8	108

**Table 28: Poor quality defect tracking**

The defects that occurred within the iteration were not being tracked for the time spent, but instead just the number of defects. By monitoring the iteration, the entries against the defects will be tracked and so the data becomes usable.

The final point regarding the data reflects the team involvement in the iterations. The number of team members should be consistent from the capacity, estimates and actuals. That is not to suggest that all team members should be fully committed to an iteration, though that would be preferable. Instead, it would be anticipated in a planned iteration that the user would give an indication of the amount of time that they can commit to the project for the next period. They would then work within the process, by assigning estimates to the tasks that are identified for them, and record their actuals against the tasks. There will be variances from this, where someone gets taken unexpectedly from the project to work on some other task, for instance, supporting a production issue, or unexpected leave, or where someone has more time than anticipated to give to the project, due to a change in plans. That should be the

exception rather than the norm. However, as tables 27 shows there is no consistency in these projects for capacity, estimates and actuals. This would suggest that either there is no consistent process in place or that the use of the process has not been captured in the tool from which the metrics are being generated. A further point of note on this was that the capacity was not captured for Project A and Project B. The team were not measuring the results of iterations, so the capacity was an oversight in the process. Again, the monitoring of the iteration will highlight who is not entering the tracking correctly. This in turn will make the data usable for the gamification iteration.

The holiday metrics and time zone metrics when viewed with the iteration data did not give any valuable insights. The iterations with the high percentage of holidays did result in lower commitment, but this was not the only reason for other drops in iteration.

This section has provided some analysis on the metric results and context data. The next section concludes on the preliminary work.

#### ***7.4 Conclusion***

This chapter has outlined the preliminary work that was completed for this experiment. In this section the approach taken to retrieve the data was outlined and the results of the analysis were presented. Finally, there was a section which examined the data and discussed the meaning.

At the start of the analysis it was hoped that the data would present a trend which was consistent across the projects and iterations. The intention was then to use gamification and to determine if that changed the data in any significant manner. However, the data failed to show any obvious trend. It is hypothesized that this is due to the quality of the data in the tool rather than such consistencies not existing in the projects. In order to continue with the project, it was determined to improve the quality of the data before applying the gamification experiment. The next chapter described the approach taken to improve the data quality.

## **8 MONITORED ITERATION**

### ***8.1 Introduction***

This section of the document outlines the approach taken to improve the quality of the data in the project tracking tool. As discussed in the previous section, the data being extracted from the tool was not complete and it was determined to try a second approach to retrieving the data. This section of the document describes the methodology used to retrieve the data and then the results of the baseline iteration.

### ***8.2 Experimentation***

This section of the document examines the methodology used to improve the capture of data from the preliminary work. This section outlines the method used to improve the data and then the project selection.

#### **8.2.1 Methodology**

##### **8.2.1.1 Monitoring**

The method of improving the entry of data involved a significant monitoring of the daily updates. Within SDT (Deci, E.L. & Ryan, R.M. 2012), discussed in more detail in the chapter on motivating software engineers, this approach would be considered external regulation. The team were not applying the updates to the tracking tool. A requirement to do so was asserted by the team leadership and confirmed by the management team. The suggestion in this was that not complying with the request would result in punishment rather than reward to those who improved their effort. In persuasive motivation, the route taken was the peripheral route. There was no direct contact with the individuals to attempt to persuade them by engaging them in the benefits of updating the tool. The data was captured on a nightly basis and then presented to the team as part of the daily Scrum. The approach was as follows:

- Before the start of the iteration a mail was sent which indicated that all updates to the project tracking utility would be monitored. This mail was sent to all the team members. It included an explanation of why this task was completed. In



addition a subsequent email was sent from the management team to reinforce that the approach was agreed upon. As part of this mail, it was requested that management be notified of those who had not updated the data for two days running;

- The project tracking utility tool was used to extract the capacity for each team member at the start of the iteration;
- Calculate the average burndown for the team members. A simple calculation was used. The individual’s total capacity was divided by the number of days in the iteration. This did not allow for individuals days off, but was sufficient to be indicative of any problems;
- On a daily basis, the estimates, actuals and “to do” were captured. These were appended manually to a daily spreadsheet;
- The captured results were presented post Scrum to the team. The focus of the discussion was those updates that were not in line with the expected capacity burndown. In advance of the meeting these items were highlighted where the data was different from the individual’s capacity. Notes were taken to record the reasons for change from expectations.

Having discussed the methodology for monitoring the use of the Rally tool, the next step is to examine the new metrics that will be captured.

#### 8.2.1.2 New Metrics

This section outlines the additional metrics that will be captured. These metrics will be used to determine if the project monitoring is succeeding.

Metric	Description
Total Estimates	This metric captures the total estimated effort for the iteration. This metric is being captured so that it can be used to compare iterations.
Total number of tasks	The number of tasks in an iteration. This is used to calculate the average task size.
Average task size	The average task size is used as a comparison across iterations. This was captured for use in the gamification

	experiment.
Total Capacity	This is the total capacity for the team. This is used in conjunction with the total actuals to calculate the total percentage of capacity used.
Total Actuals	This is the total actuals for the team in the iteration.
% of capacity used	This is the percentage of capacity used. This should be reflective of the team's work. If the team worked overtime this should be above 100%, if the team was not fully utilized then the capacity will not be utilized.

**Table 29: Shows the additional metrics captured in this iteration**

### 8.2.2 Interview with the Scrum Master

As part of the process it was determined to have a second discussion with the project Scrum Master. The aim of this discussion was to extract opinion as to why the project data was so poor. The discussion was an unstructured interview. This format was chosen as the subject was familiar with the issues and it was considered better to let them guide the discussion.

### 8.2.3 Project Selection

The monitoring of the project through iterations is an involved task. It reduces the capacity on one team member as it involves nightly updates to the monitoring sheet. It also extends the daily Scrum time by approximately 5 minutes every day as the team reviews the outcome. Based on this impact it was determined that it would be only possible to proceed with one project using this monitoring. There were two active projects:

- Project G: an extension of Project C, which was based on the Financial Cost Allocation Data Warehouse;
- Project H: an extension of project F, which was based on the Financial Statistics Data Warehouse.

The project selected was Project H. The major reasons for the selection were as follows:

- The team in project H were leading the reporting of metrics. Historically, this was because this project was perceived as struggling and in need of attention.

The team were already producing an end of iteration metrics pack, and there had been some discussion on the validity of this data;

- The team in project H was the one closest to the author. It was therefore felt that monitoring the iteration would be easier for someone who was familiar with the team.

This section has outlined the reasoning for the selection of Project H. The next section of the document evaluates the results.

### **8.3 Evaluation**

This section of the document examines the results from the secondary analysis. The section describes the project; outlines the metrics captured and then reflects on their meaning.

#### **8.3.1 Project details**

This section of the document describes project H. As mentioned previously, this project relates to a Financial Statistics Data Warehouse. Project H is an extension to the previously described Project F. At time of writing the project is still in progress, but will have completed by the time this thesis is submitted. An initial iteration was completed with no monitoring. The second iteration is the first iteration being monitored. The third iteration is both monitored and contains the gamification experiment.

#### **8.3.2 Metrics**

This section of the document describes metrics that were captured for the monitored iteration and the unmonitored iteration.

Metric	Iteration 1	Iteration 2
# of users with capacity	13	15
# of users with estimates	13	15
# of users with actuals	13	15

Total Estimates	433	717
Total number of tasks	187	300
Average task size	2.31	2.39
Total Capacity	693	828
Actuals	475	711.5
% of capacity used	69%	87%
Actuals versus Estimate	110%	101%

**Table 30: The metric results captured for the new iteration**

### 8.3.3 Discussion

This section of the document discusses the additional iterations used to provide a baseline for the gamification experiment. The reasons for this were based on the data quality of the original project data. This section starts with a review of the interview results and then discusses the findings from the iteration.

#### 8.3.3.1 Interview Results

The aim of the monitoring was to improve the capturing of the data. From the interview with the scrum master, the main issues with the data capture were:

- Paucity of data, with members not capturing all their tasks successfully;
- Quality of data, team members were not reflecting their actual efforts in the tool. This was particularly true where the team members had felt that actuals above estimates would be frowned on.

#### 8.3.3.2 Key Findings

This section of the document highlights the key findings of the iteration. The first issue identified was that not all members were being tracked. The number of individuals in the two iterations varies from Iteration 1 to Iteration 2. There are two reasons for this:

- A team member returned from leave to re-join the team;

- A second member who assisted in the project based on the type of the requirements, specifically if a front end tool was being modified, was not recording time.

This data did not impact on the quality of the data that was included in the project tracking tool, but it did mean the data did not reflect the actual effort involved in iteration 1.

The second issue was that the team capacity in the tool was too high. The team were using a blanket capacity of six hours a day for every day they were present. However, meetings and other activities could impact their hours. The team reviewed the hours and established that the capacity varied from day to day but was on average 5.5 hours a day.

The third issue was that team members were reluctant to enter actuals when they surpassed the estimates. The explanation given was that the team felt that this would provoke a reaction from the team leads and management. However, in reality, the most important measure in Agile is delivered software, so one of the main purposes of capturing the actuals is to refine future estimates. This actually had a negative impact on the team as the metrics indicated that the team were working within capacity, while in reality they were working overtime. The change from 68% of capacity to 87% in the second iteration was a significant change in the behaviour towards the tool, however the process is not complete.

The final lesson learnt from this activity was that a number of tasks are missed in the original estimating process. These tasks were added during the iteration and not captured previously. This resulted in an increase in the total estimates from the original estimates.

The main benefits of completing this process were an improvement in the quality of the data. However, it would be anticipated that this can be improved further as the team adjust to being monitored.

One negative aspect was the team's reaction to the monitoring. In general it was felt that the approach was not in keeping with the Agile process that was being followed.

This was reflected by one team member who said “there was a lack of trust on team members to complete the forms“. However, a “self-organizing team” should be capable of holding each other responsible. The monitoring does this but, rather than being the team it is completed by one member of team. The monitoring did not result in anything being raised in scrum that any team member could not have raised. The reaction was in line with expectations from SDT model. The process was implemented without prior consent rather than something that was agreed to by all members. The controlled motivation is external to both the team and the individual members. There is no expectation that this approach will lead to an internalization of the value of tracking effort correctly. While the surveillance is on-going it is expected that the team will respond by maintaining the tracking data. It would be preferable if a means could be devised to persuade the team to internalize this need. This was considered out of the scope of this experiment as the effort would have been significant and it is anticipated that this would take a number of iterations to track and capture. This would represent a possible future project and is discussed further in the thesis conclusions.

#### ***8.4 Conclusion***

This chapter has looked at the process of monitoring the data being captured in the project tracking tool. This task was completed because the data was inconsistent and did not appear to accurately reflect project activity. The overall affect was a significant improvement in the quality of the data. Having completed this task the next step was to complete the gamification experiment. The next chapter explains the experiment in detail.

## **9 GAME ITERATION**

### ***9.1 Introduction***

The purpose of this section is to describe the gamification experiment. The section first describes the game and the experiment methodology. The next section describes the experiment results and then discusses the results.

### ***9.2 Experimentation***

This section of the document describes the gamification experiment. The aim of the experiment is first described. The next section describes the gamification itself. The next section describes the components of the experiment.

#### **9.2.1 Aim**

The aim of the experiment was as follows:

“To improve the project tracking and to determine if that has an impact on the estimation accuracy”.

This experiment is a direct test of the dissertation research question. This includes the sub-question to determine if gamification can be used to improve project tracking. The second part of this aim is to improve estimation accuracy. This relates to a second sub-question of the dissertation. This asks if gamification can positively impact the efficiency of a project. If the estimates improve, this will indirectly impact on the efficiency by highlighting of bottlenecks in the process.

To achieve this aim, we wanted to encourage the team members to break the tasks down to a lower grain. The first benefit of this is that the smaller tasks would be more easily tracked. A second benefit would be that by discussing the tasks at a lower level, the team would be able to provide more precise estimates.

### 9.2.2 Game

The game devised was a simple lottery with a reward for the winner. The method was to add the lottery game element to the existing estimation process. This was done by considering each completed task as a ticket into the lottery. At the end of the experiment iteration, all completed tasks, together with the name of the task owner, were put into a container and a winner picked out by chance.

The change meets the definition of gamification by adding a game element to an existing process. It is not a serious game because the process can be used without interacting with the game and the process already exists before the game element was added. The next table examines the process from the perspective of a game.

Framework Element	Description
Purpose of the game	The aim of the game is to win the lottery.
Procedure for action	The players play by decomposing their workload into tasks which are as small as the rules allow. They then complete as many task as they can in the iteration.
Rules governing action	It is a straightforward game with a limited number of rules. They are as follows: <ul style="list-style-type: none"> <li>• All tasks must have a minimum size of one hour. If tasks are smaller than that they need to be combined;</li> <li>• All tasks must have a maximum size of four hours. If tasks are larger than that they must be broken into sub tasks;</li> <li>• Tasks will be reviewed, by a core panel made up of the onshore and offshore team leads and the Scrum Master.</li> </ul>
Number of required participants	There is no limit to the number of participants. It is open to all team



	members except those on the core panel.
Roles of participants	The team member's role is to decompose the tasks to a level within the game rules and then to participate in the iteration and complete as many tasks as possible. The core panel's task is to review the tasks to ensure that all tasks are valid. The panel will also review that all completed tasks are genuinely completed.
Results or pay-off	The outcome of the lottery would be a prize.
Abilities and skills required for action	The ability to decompose tasks is the only ability that can advance a player in the game. Other than that, it is due diligence when updating the project tracking tool.

**Table 31: Describes the lottery game uses the games framework**

Rewards need to be designed to be equitable and to acknowledge effective performance without incorporating controlling elements such as competition among team-mates or pressure to meet the numbers. The lottery nature of the game left participation in the hands of the game player, and also took an element of competition out of the game.

Given that the game is based on a reward, it would seem to be categorized as external regulation. However, the fact that the prize was not given directly to the best performing member took an element of control from the process. As well as introducing the game, an attempt was made to describe the purpose of the game, both direct consequences, such as increasing the capture of tracking data, and indirectly such as being in a position to highlight to the management team the amount of overtime the team work to meet the requirements. For this iteration, the motivation was attempted using a more central route.

This section has described the game. The next section attempts to describe other games which were considered when designing the gamification experiment.

### **9.2.3 Other Game Ideas**

This section describes some of the other game ideas and gives reasons for why they were not included.

The first idea was to create a leader board based on the actuals completed in the iteration. This would reward the person who completed the most tasks in the iteration.

There were a number of concerns with this as an approach:

- The game would be open to being manipulated or played. It would be easy for someone to invent tasks or exaggerate the actuals to increase their chance of winning. Without examining every task closely it would be difficult to monitor that this manipulation was not happening;
- From the perspective of ethics, it may encourage individuals to work more than their job requires as they focus on the winning the prize;
- Controlled and external regulation motivation can often result in demotivating team members who are not in the top performers list.

A second, but similar idea was to create a leader board based on the variance between their actuals and estimates. Similar to the first idea this would be open to gamification and the other issues regarding ethics and demotivating other team members also apply.

One final idea was to apply time pressure to the iteration. This would be in the form of a clock which would count down to the completion of the iteration. The clock could be applied at the level of iteration, story or task. The lower level would be more difficult to implement as it would have to allow for breaks for meetings or going home in the middle of a task or story. The iteration clock would be easier to implement, but it was not clear if there would be any value when the team already talks about the iteration daily.

Having examined these and other ideas it was felt that the simple lottery game was the best approach for the experiment. Having described the game selection, the next section describes the methodology used in introducing the game.

### **9.2.4 Methodology**

This section of the document describes the components or elements of the experiment. The initial task was to garner support for the game and then establish the options for

the reward. The second task was to survey the team to establish the preferred prize. The third task was to describe the game to the team. The fourth task was to monitor the game when it was in progress. The fifth task was the completion of the game, with the lottery and awarding of the prize. The final task was to interview game participants to gain an understanding of how the game impacted them.

#### 9.2.4.1 Establish options for reward

The purpose of this section was to establish options of reward. Before doing this, management support for gamifying the iteration was requested. Given the status of the project and the continued drive for improvement this was viewed favourably. The only concern, which was raised by the Project Management Office as the choices of reward were agreed, was that “team members would be being rewarded for doing their basic job”.

Having achieved permission to conduct the experiment the next step was to establish the reward. The approach was as follows:

- Create a list of possible rewards;
- Pass the list to Product Manager for feasibility;
- Have the Product Manager select a limited amount.

When creating a list of possible rewards, the first step was to review the literature and establish motivational factors. Having reviewed this list the next step was to generate a list using these motivational factors.

#	Motivators of Software Engineers
1	Identify with the task (clear goals, personal interest, know purpose of task, how it fits in with whole, job satisfaction, producing identifiable piece of quality work).
2	Employee participation/involvement/working with others.
3	Good management (senior management support, teambuilding, good communication).
4	Career Path (opportunity for advancement, promotion prospect, career planning).

5	Variety of Work (e.g. making good use of skills, being stretched).
6	Sense of belonging/supportive relationships.
7	Rewards and incentives (e.g. scope for increased pay and benefits linked to performance).
8	Recognition (for a high quality, good job done based on objective criteria).
9	Development needs addressed (e.g. training opportunities to widen skills; opportunity to specialise).
10	Technically challenging work.
11	Job security/stable environment.
12	Feedback.
13	Autonomy Work/life balance (flexibility in work times, caring manager/employer, work location).
14	Making a contribution/task significance (degree to which the job has a substantial impact on the lives or work of other people).
15	Empowerment/responsibility.
16	Appropriate working conditions/environment/good equipment/tools/physical space/quiet.
17	Trust/respect.
18	Equity.
19	Working in company that is successful (e.g. financially stable).
20	Sufficient resources.

**Table 32: Motivational factors for software engineers (Sharp, H., *et al.*, 2009)**

Item #	Description
1	One to one time with a business representative. You could discuss aspects of project and present ideas. You would identify the specific area you are interested in and a session with the appropriate business representative organized.
2	Extra time with a business representative for the team. Rather than one to one you could earn extra time with a business representative for the team.
3	One to one time with the technical architects to discuss technical solution and present ideas. Again, you would select the area and a one to one

	session would be organized for you to discuss this with an appropriate representative
4	Extra time with the technical architect for the team. Rather than one to one you could earn extra time with a business representative for the team.
5	Given the opportunity to review the technical backlog and prioritize the work.
6	More time allocated in the following iteration for you to work on technical stories from the backlog.
7	Opportunity to take in tasks outside your normal work domain, for instance, a developer might work in design. This would be limited by your ability to complete the task.
8	Extra training opportunity. You would be fast tracked for training in areas related to your job.
9	Opportunity to spend time off the team, to working with another team to look to learn from their processes.
10	Opportunity to spend time on personal projects, for instance those from a community of practice, rather than on the project. This would give you time to focus on your personal development and standing in the company.
11	Access to relevant conferences. You would be fast-tracked for future conference visits.
12	Success raised to management team. When you complete an iteration successfully, your personal achievements be flagged to management.
13	Opportunity to prioritize process improvements. Rather than being done by the team, your success would allow you to determine which processes should be changed as part of the next iteration.
14	Opportunity to work on more critical tasks. In this instance you would get an opportunity to pick the tasks that you work on, so that the task is more relevant to you.
15	Opportunity to lead the Scrum in the next iteration. If you are the best performer in an iteration then you get to lead the Scrum in the next iteration.

**Table 33: Shows the options presented to the Project Manager as suggested rewards**

Reward Item	Motivation Factor
Item 1	1,2
Item 2	1,2,6
Item 3	1,2
Item 4	1,2,6
Item 5	15
Item 6	5,10
Item 7	4, 10
Item 8	9
Item 9	9,2
Item 10	9,4,5
Item 11	9
Item 12	3,8
Item 13	2,4
Item 14	5,9
Item 15	2,4

**Table 34: For each of the items in the list of rewards we have identified the motivational factor associated with this reward.**

Having established a list of possible rewards, the next step was to pass it to the Project Manager for review. This was done in the form of an email, followed by a discussion on the items. The main purpose of the discussion was to explain the reward options, but also to ask for a limited set of selected rewards. Following on from this, the Project Manager returned the list in time to be passed to the team.

Item #	Description
8	Extra training opportunity. You would be fast-tracked for training in areas related to your job.
10	Opportunity to spend time on personal projects, for instance those from a community of practice, rather than on the project. This would give you time to focus on your personal development and standing in the company.
11	Access to relevant conferences. You would be fast tracked for future conference visits.

12	Success raised to management team. When you complete an iteration successfully, your personal achievements be flagged to management.
15	Opportunity to lead the Scrum in the next iteration. If you are the best performer in an iteration then you get to lead the Scrum in the next iteration.

**Table 35: Items selected for presentation to the team. These items will be passed as a survey to the team for selection.**

Having established a list of possible rewards the next step was to allow the team to select the reward they were most interested in.

#### 9.2.4.2 Introduce the game

As part of the iteration planning the game was explained to the team. The rules of the game were explained and how and when the lottery would happen. The next step for the team was to complete the survey. The contact method and how to complete the survey was described. Overall, the response was quite muted, though there was some excitement at the thought of winning a prize.

#### 9.2.4.3 Survey

This section describes the approach for the survey. The survey was conducted with survey monkey. The possible rewards where put to the member and the members were asked to rate each of these in terms of how likely it was to motivate them. The survey was sent to all then members of the team. The team was given 5 days to respond.

#### 9.2.4.4 The game iteration

The game is straightforward. The team estimated as normal, in a two day planning session, with an awareness of the game rules. After the iteration planning session, the tasks were checked to see if all tasks where valid. There was no obvious attempt to manipulate tasks to achieve an advantage in the game.

During the iteration, a story was pulled from the iteration. This was an unusual occurrence for the project, and related to data volumes being created and the longer term impact of the cost of maintaining the data.

The story was split into two parts

- The tasks that had already been completed, together with the new tasks for the effort to rewind the work completed;
- The remaining tasks which had not been started.

New stories were identified to ensure that the team had enough work for the remainder of the iteration. The team estimated them using their normal estimation process. All tasks created as part of this rework were considered part of the game. These will be reviewed to insure that no manipulation of tasks has taken place. Again there was none.

The iteration proceeded as normal. At the end of the iteration all tasks completed were identified. As part of the retrospective a winner was drawn.

#### 9.2.4.5 Post-Game interviews

Following on from the game, interviews would be conducted with the team members. The interview was a semi-structured interview. A number of questions were prepared in advance, but the idea of the interview was that the team members could have relevant feedback which was not part of the original set of questions. The following were the prepared questions for the interview

Q1: How familiar are you with the concept of gamification? Can you give a rating, with 1 being not familiar and 5 being very aware. What is your opinion of it?

Q2: Did you understand the purpose of the game? Rate your understanding from 1 to 5. Can you state what you thought the purpose of the game was?

Q3: When you first heard the idea, did you think the game would work? Why?



Q4: How well was the game explained to you before the game began. Were the rules and instructions clear to you? Rate from 1 to 5. Please comment on your rating.

Q5: Do you think it was possible to manipulate the game to improve your chances of winning? If so, how would you do this?

Q6: Having played the game, would you play the game the same way or would you do something different?

Q7: What suggestions would you have to improve the game?

Q8: Are there any other comments you would like to add relating to the iteration or the game?

This section has described the interview. It is the final element in the methodology of the experiment. The next section proves an evaluation of the experiment.

### **9.3 Evaluation**

This section of the document describes the results of the experiment. The section first details the results of the preparation components. The results of the experiment metrics are then detailed. The next section then discusses the results of the experiment.

#### **9.3.1 Survey results**

Option	# of Responses	Not Interested	Somewhat Interested	Interested	Very Interested	Extremely Interested
8	6	0	0	0	3	3
10	6	1	2	1	2	0
11	5	0	1	2	1	1
12	6	0	0	4	1	1

15	6	2	3	0	1	0
----	---	---	---	---	---	---

**Table 36: The survey results from the team**

The take up of the survey was disappointing, with 7 respondents, however one respondent skipped all questions, so there was only 6 real respondents. Of these one respondent missed a question, but otherwise there was a full response. The survey was anonymous to allow team members privacy in their responses. The team members were made aware of the anonymity; however, despite this the response rate was poor. The results of the survey are displayed in the following table.

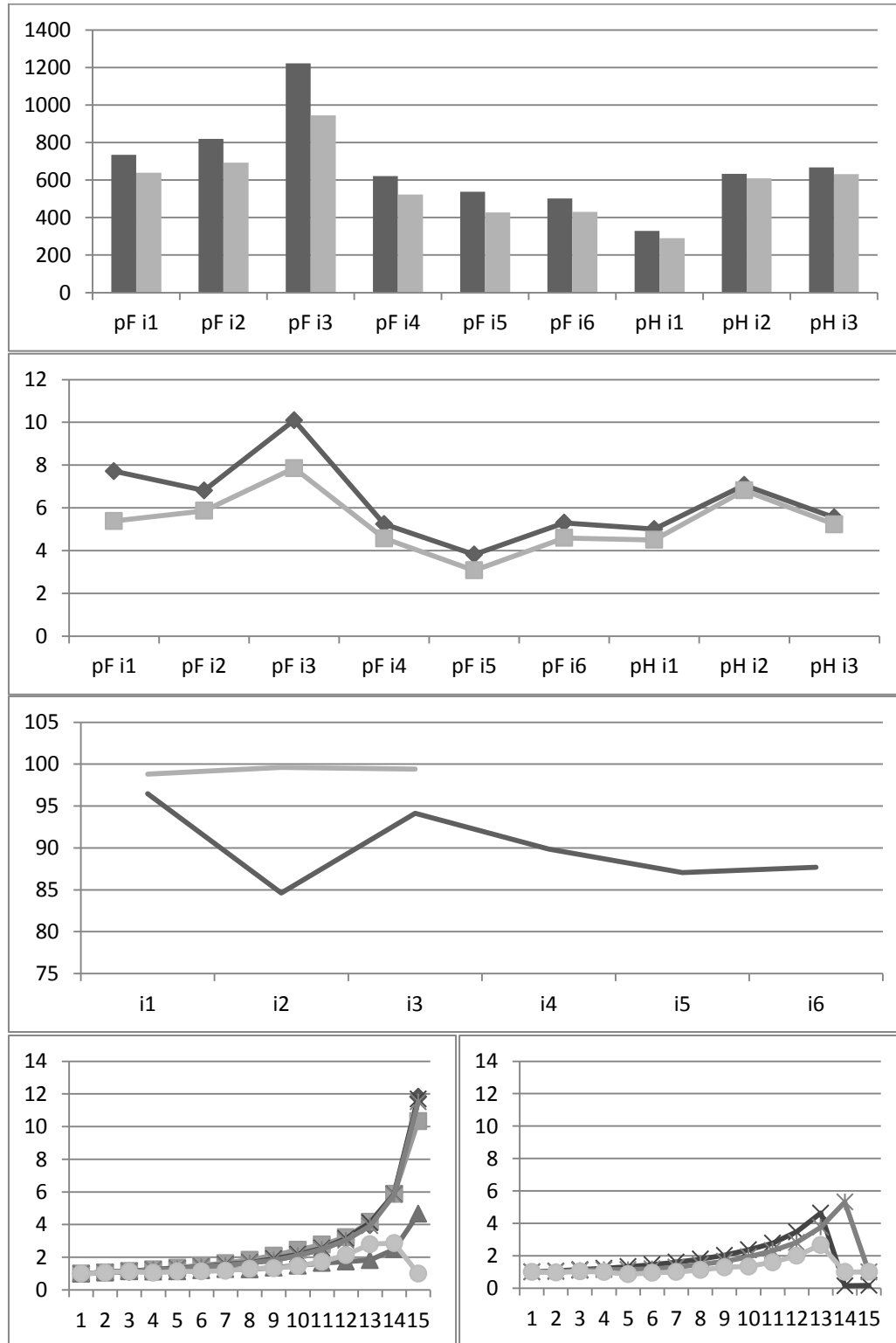
Description	Weighted Average
Extra training opportunity. You would be fast-tracked for training in areas related to your job.	4.50
Opportunity to spend time on personal projects, for instance those from a community of practice, rather than on the project. This would give you time to focus on your personal development and standing in the company.	2.67
Access to relevant conferences. You would be fast tracked for future conference visits.	3.40
Success raised to management team. When you complete an iteration successfully, your personal achievements be flagged to management.	3.50
Opportunity to lead the Scrum in the next iteration. If you are the best performer in an iteration then you get to lead the Scrum in the next iteration.	2.00

**Table 37: The weighted averages show that the extra training was the most popular selection**

The opportunity to run the Scrum was not seen as a popular option, with only one person interested. Surprisingly the opportunity to spend time off work, working on personal projects and within community of practice was not popular. The option selected related to training, while the other option which related to conference access also scored highly.

### 9.3.2 Metrics

This section of the document displays the metrics for the iteration



**Figure 19: Comparison of current project (G) with previous project (F)**

The top chart shows the actual versus estimates extended over 9 iterations.

The first six X-Axis entries show the project F (pF), the next three are project H (pH).

From this we can see that the new iterations (pH i2 and pH i3), which consist of monitoring followed by monitoring and gamification, have estimates and actuals which are closely aligned. The next chart shows the velocity of the combined projects. From this we can see that the gap between estimated and actual velocity is closing. The third chart shows the rework. This is the amount of time spent on actuals as compared with the total time in the iteration. The iterations from the current project, lighter grey have much less time spent on defects than the older iterations. Finally, the last level of graphs show the Iteration Burndown combination. These two graphs, the left is for project F while the right is project H. The graph shows that the Iteration Burndown is healthier, as the iterations are not spilling any stories. There is still room for improvement as the Burndown is not in line with the idea for most of the iteration.

Metric	Iteration 1	Iteration 2	Iteration 3
# of users with capacity	13	15	15
# of users with estimates	13	15	15
# of users with actuals	13	15	15
Total Estimates	433	717	735
Total number of tasks	187	300	380
Average task size	2.31	2.39	1.934211
Total Capacity	693	828	795
Actuals	475	711.5	774
% of capacity used	69%	87%	97%
Actuals versus Estimate	110%	101%	105%

**Table 38: The metrics from the gamified iteration**

These metrics are shown against the previous two iterations.

### 9.3.3 Interview results

Following on from the iteration, a number of interviews were conducted with the team members who had participated in the game. The aim of the interview was to understand how the gamification process had impacted the players approach to the iteration.

Question	Interviewee 1 – Development	Interviewee 2 – QA
How familiar are you with the concept of gamification? Can you give a rating, with 1 being not familiar and 5 being very aware. What is your opinion of it?	2 Not very familiar. No opinion given.	3 Somewhat familiar, had heard of it before.
Did you understand the purpose of the game? Rate your understanding from 1 to 5. Can you state what you thought the purpose of the game was?	2 – Purpose of the game was to ensure that project tracking tool was kept up to date.	1 – To help improve the team, in particular by altering the behaviour in the Scrum.
When you first heard the idea, did you think the game would work? Why?	No, did not think it would change the process we were following.	No, was not sure of the purpose of the game.
How well was the game explained to you before the game began? Where the rules and instructions clear to you? Rate from 1 to 5. Please comment on your rating	2 – Not particularly well described.	2 – Could have been described better.
Do you think it was possible to manipulate the game to improve your chances of winning? If so, how would you do this?	4 – Very likely. Breaking up tasks and pushing up the estimates for very small items that could be combined into one task.	3 – Maybe. As the tasks were reviewed, as part of the iteration and part of the game it did not seem possible to follow anything other than the estimation

		tool.
Having played the game, would you play the game the same way or would you do something different?	Yes, I would break tasks down to a lower level.	Having not understood the change to the estimation process, would look to review the tasks to see if they could be split further.
What suggestions would you have to improve the game?	Using the task breakdown is fine, no additional suggestions.	Game is unfair as some people cannot break their tasks down to same level. If the player does one task that was one hour long, and another player does one task that is three hours long, they both get one ticket into the lottery. Rather if the process was changed so that you get an ticket for every in completed tasks the draw would be fairer.
Is there any other comments you would like to add relating to the iteration or the game?	None.	No.

**Table 39: Sample Interview results**

The table shows sample results from the interviews. One sample was taken from a developer, while the other was taken from a QA member.

Having shown the interview, the next step is to review the results.

### 9.3.4 Discussion

This section of the document discusses the results of the gamification.

The most positive result from the gamification was that the size of the tasks decreased significantly, down from 2.39 in iteration 2 to 1.93 in iteration 3. This is a decrease of 19% which can be attributed to the gamification experiment. While it is possible that the type of story was the reason for this decrease, historically, the average size of task in this project was significantly higher at 4.2 hours and a StdDev of 1.9.

The actual hours as percentage of the capacity are indicative that the monitoring has had the desired impact on the logging of hours in the tracking tool. Over the past three iterations, this had moved from 69% in iteration 1 to 97% in iteration 3. There is still a feeling that the team is reluctant to record their actuals in the tool when they vary from the estimates. The actuals are not acting as a feedback to the next iteration session.

The actual versus estimates was not as effective as had been hoped. It was thought that the estimates would improve if the tasks were smaller. However, the iteration before had been more for the estimates versus actuals, 101% versus 105%. This may have been a by-product of the initial decision to monitor the iteration. The team may have been trying to get their actuals to match their estimates, and therefore only be inputting the actuals in until they meet the estimates, but not after they go over the value. As it is believed that the full capacity is being worked on the project it is possible to normalise the results between the three iterations and examine whether this has any impact on the results. The formula for this is:

$$\text{Capacity/Estimates} * 100$$

Iteration	Capacity	Actuals	Estimate	Actuals versus Estimates	Capacity versus Estimates
Iteration 1	693	475	433	110%	160%
Iteration 2	828	717	711.5	101%	116%
Iteration 3	795	774	735	105%	108%

3					
---	--	--	--	--	--

**Table 40: Capacity versus Actuals comparisons**

This table shows the difference between the “actuals versus estimates” and “capacity versus estimates”. If the assumption is that the team are still not applying their actuals completely is correct, then it is more correct to use the capacity rather than actuals. This would still not be completely accurate, because the capacity does not reflect the full amount of time spent on the project.

The developers do not appear to be doing their own estimates. They are using templates to help with the process, however, the review of the estimates revealed two items:

- Minute tasks where being kept separate. We had estimates for code reviews, and peer review assigned to one person, for very small tasks. The tasks were kept separate because the estimating tool had them as separate. However, the story change was so small it rendered the tasks trivial. However, the team were reluctant to drop one of the tasks and combine it with the other;
- The most recent member to join the team naturally needed more time than the more experienced members. However, the estimation tool had not taken this into account, and the team member was uncomfortable making a change to the estimated effort.

An issue with the game was that it was not well understood by the team. Although the game was seen as relatively simple and not needing much explanation it is clear from the interview results that the purpose of the game was not clear. In addition, the rules were not fully understood and team members did not try to “play” the game to their own advantage. This lack of understanding may highlight that the team communication is not as clear as was assumed. The team may be comfortable at speaking about technology related components, but the variation introduced by discussing something novel highlights deficiencies in communication.

Another aspect of this was the means in which the gamification was introduced. The initiative was from the leadership down, although there was discussion around the purpose. In the previous iterations, the majority of communication had been based on the issues of technical delivery. This communication would be serious and focused on problems. Therefore, despite attempts to introduce an element of fun, it may be that the



team where not comfortable with this change. The game was still considered part of an external regulation and the motivation would be considered controlled. The affordance between the game and team was not considered when introducing the game. The lack of attempts to play the game may suggest that the team were not comfortable acting out of bounds of the iteration norms. In this aspect it would be necessary to spend more time relaxing the iteration and introducing more elements of gamification before a lasting impact could be expected.

This completes the discussion on the gamification iteration. The next chapter provides a conclusion for the thesis.

## **10 CONCLUSION**

### ***10.1 Introduction***

This chapter concludes the dissertation. The next section is the research definition. This restates the research question and then discusses the success of the experiment. The next section discusses the contribution to the body of knowledge, to academia and to the experiment site. The next section summarizes the conclusions on the experiment. It outlines the limits of experiment and the limits of applying gamification. The final section outlines related work which could be undertaken to extend the experiment and other potential areas of study which were encountered during the project development.

### ***10.2 Research Definition & Research Overview***

This section of the document describes the research question. It then discusses whether this question was answered by the thesis. The research question was as follows:

RQ1: Can gamification be used in manner that has a positive impact on an Agile project? This can be decomposed into a number of sub-questions. Can gamification be used to improve the tracking of an Agile project? Can gamification be used to improve the efficiency of the team? Can gamification be used to have an impact on the motivation of the team?

There was a small improvement in the project tracking for the size of the tasks. So the first sub-question has had a positive result. However, this comes with the caveat that the gamification experiment was only applied to one iteration, and for one project. The research results were therefore inconclusive, but in need of further testing. If the experiment had been conducted over a number of iterations and with a number of different project teams, it may have been possible, using inductive reasoning, to conclude that gamification had impacted the agile project.

The second sub-question related to the impact on team efficiency. In this respect the experiment results has a positive result when compared against capacity. However, as the historical data was inaccurate and did not reflect the actual project there was a need

to introduce a monitoring iteration to improve the quality of the data. This was successful, but shortened the experiment. Only one iteration was available for the gamification experiment.

The final sub question related to the motivation of the team. The monitoring influenced in a negative manner. Once the initial data was found to be inaccurate, the project timeline was compressed. The introduction of monitoring was done in an expedient manner. The approach to this was to apply controlled regulation. This is an externally regulated form of motivation, so the process is unlikely to be internalized by the team members. It also resulted in some resentment from the team, who did not feel that the monitoring was part of Agile methodologies. The team did not “identify with the task”, which is a key motivating factor for software engineers, (Sharp, H *et al.*, 2009). The effort of inputting the tracking was seen as boring work, which is an example of a “poor working environment” de-motivator for software engineers. (Sharp, H *et al.*, 2009). It is also possible that the gamification was connected with the monitoring as both were introduced to the team by the same leadership group and were separated only by a single 15 day period. Under these conditions, the team have been suspicious of any additional activities which were added to the iteration cadence.

The experiment focused on some aspects of Agile projects, the use of project tracking tools, estimation and motivation. There are other areas of Agile against which gamification could have been applied, for example, Iteration Planning, Pair Programming or Scrum Updates. The context of the research question was narrowed once the quality of the initial data was determined. The data did not contain any clear trends which could be analysed. Without this, the experiment had to be designed to run after a monitoring iteration. This restricted the experiment by limiting the amount of time and the number of the projects.

In conclusion, although a small improvement of 19% on the average task size was found after gamification had been applied, it is uncertain if this was due to the gamification. There is a possibility that this change of task size was related to the type of stories in the iteration or some other aspect of the single iteration. It is also a possibility that the monitoring introduced impacted the willingness to partake in the gamification. The experiment only focused on one aspect of Agile, while many others are possible for experimentation. Having examined the research question, the next

section examines the contribution to the body of knowledge, highlights the impacts of the dissertation to the organization and the field of study.

### ***10.3 Contributions to the Body of Knowledge***

This section examines the contribution to body of knowledge. The first section examines contributions to the experiment team and organization. The second section examines the contribution to the field of study.

#### **10.3.1 Contributions to the organization**

This section examines how the experiment benefits that organization in which the experiment took place.

The experiment's main contribution to the experiment team was to highlight the deficiencies in the use of the project tracking tool. Although the team were beginning to examine the use of metrics to measure the team's progress, the data the metrics were based on was inaccurate. As a result of the data inaccuracies, the effort the team were expending was not being reported correctly. The monitoring process, which was not popular with the team, resulted in data which reflected the team's efforts during the iteration. The teams' actuals increased from 69% of capacity to 97% of capacity. This was achieved through more accurate capture of the data and by examining the capacity the team were committing to.

The experiment highlighted issues in the quality of the data in the project tracking tool. As part of the experiment a monitoring methodology was devised and introduced to the team. This positively impacted on the quality of the data. The approach was largely manual. Further work is required to automate the reporting, and to establish the balance between the benefits gained from monitoring and the negative impact on the team.

The experiment highlighted areas of the process that could be improved on. For instance, the data highlighted that the team may finish earlier than the iteration end date, but that they were unable to take in more work. When team members had completed their work for an iteration, they had a willingness to take in bonus stories, but they were unsure as to what to take in. The team is moving from iteration to

iteration using a “just in time” approach. The team needs to make better use of the Agile backlog features and build a clearly prioritized backlog.

The experiment highlighted communication issues within the team. The gamification experiment was a very simple lottery game, with few changes to the existing process. However, the members of the team who were interviewed did not feel that the game purpose and process was clear to them. When asked for the purpose of the game, the interviewee did not give answers that were in line with the actual explained purpose. The purpose of the game was to generate lower level tasks and to improve estimation as a result. However, the views ranged from “to improve the Scrum updates” to “to increase the capture of data in the tracking tool”. In addition to this, the lack of gaming of the game, is indicative of a team that are rigid in their approach. The team were not comfortable enough in their environment to try and play the game. During the experiment design, a significant amount of time was given to how to produce an experiment that could be used over time without the team members being able to manipulate the results. A number of experiment ideas were rejected because of the gaming element. However no attempt was made to do this. The gaming of the iteration would be the team, “acting out of bounds” (Deterding S, 2013).

Having examined the benefits to the organization, the next section examines the benefits to the existing academic body of knowledge.

### **10.3.2 Contributions to Academia**

This section of the document examines the contribution to the body of knowledge.

The paper has identified a process to add gamification into existing Agile projects. The process identified was as follows:

- Select a set of projects for inclusion in analysis. The project selection process should identify projects which have similar characteristics. If the projects are significantly different then the data produced will be different, as the tasks will vary significantly;
- Analyse the data for trends. This analysis should be based on existing Agile metrics that are well understood by the industry;

- Monitor iterations to improve data capture. Monitoring an iteration is labour intensive. However, as team members seem uncomfortable recording tracking data, it is necessary. The process followed in this experiment was to discuss the To Do hours and Actual hours entered into the system against the ideal Burndown;
- Perform gamification experiment. This involved a number of steps:
  - Identify a selection of possible rewards for the team;
  - Get managerial approval for a subset of the rewards;
  - Survey the team to determine which reward was most popular;
  - Provide a detailed explanation of the game to the team;
  - Run the game;
  - Interview the game players to establish how the game changed their attitude to the project.

This process could be followed by other projects which are attempting to add gamification into an Agile process.

The experiment aligned with the revised definition of gamification (Deterding, 2013). Introducing the mechanics of a game did not impact the team in a significant manner. The team working environment was not conducive with the adoption of game elements. The projects were running in a pressurised environment, where there was a focus on delivery. Adding gamification into this environment did not have an impact because the team were being asked to move significantly from their normal behaviour. The experiment also contributes to the work on SDT motivation. The monitored iteration introduced a controlled motivation strategy. The outcome of the approach was better metrics, which more accurately reflected the team's actual work. Although this benefited the team, by highlighting overtime being done to the management, the approach was unpopular. The team has not internalized the need to do this. A better approach to adopting monitoring could be established and a more central route used to ensure its adoption.

Having discussed the benefits to the academic arena, the next section examines the experiment and evaluates its success and limitations.

## ***10.4 Experimentation, Evaluation and Limitation***

This section of the document provides a summary of the dissertation. It summarizes the literature review and the experiment. It provides the final evaluation for the experiment and discusses the limitations.

### **10.4.1 Experimentation Overview**

This section of the document describes the approach taken to the experiment. It gives a brief overview of the literature review and then describes the experiment. The results of the experiment are then evaluated and the limitations of the thesis are discussed.

Project Management is an old discipline dating back to ancient times. It has evolved over time and there now exists many standards for using project management in business. In contrast, software engineering is a relatively new discipline, and by extension software project management is even newer. Building software is different from other engineering practices, it is non-deterministic and not transparent. Despite this software project management has developed with a focus on up-front analysis and verification at the end. Software projects have seen a high rate of failure, with a broad range of failure reasons. Agile methodologies have evolved to address these issues.

Agile represents a family of software engineering methodologies which share an IID background. These methodologies have been designed to resolve some of the issues with traditional software development methodologies. IID methodologies have been evolving alongside the traditional methodologies, but failed to gain the same widespread use, partially because they were more complex to understand and partially due to the standardization of traditional methodologies for large scale government projects, such as those for the US DoD. Agile has now gained traction in the industry, with 95% of organizations now using Agile in some form. There are however difficulties outstanding for Agile, most notably in global software development. Converting teams from Waterfall to Agile has also been a difficult issue.

Motivation theories are relatively new, stemming from the emergence of new disciplines such as human resources in the early 20th century. The focus of motivation

has been the fulfilment of needs which are defined in a hierarchy, which focuses on the basic requirements of survival, but extends to self-realization. As you satisfy base needs, motivation will focus on filling needs from the next level above in the hierarchy. Motivation can be intrinsic or extrinsic, with intrinsic motivation being recognized as having more influence over a person's behaviour. It has been found that some extrinsic motivators have a negative impact on the person's intrinsic motivators. A number of models have been created which attempt to explain the different types of motivation. SDT has categorized motivation in a manner that highlights which motivations are controlled and which are autonomous. Finally, software engineers are generally considered different from other work groups. Software Engineers need to identify with the tasks and value independence highly.

Initial gamification definitions focused on mechanics. The software industry seized on this and started producing gamified applications. These applications were created by taking an existing application and adding a gamified element. For example, adding a contribution leader board to an existing knowledge sharing Wiki. The industry largely ignored existing studies in motivation, which suggested that these external controls did not result in behaviour that was lasting as the individual would not internalize the behaviour. In some instances, these external controls could damage the individual's motivation. In addition to this, the ethics of changing a person's behaviour were ignored. The behaviour desired was often determined by the organization's preferred behaviour rather than that of the individual. Revising the definition, the focus was switched from the mechanics of gamification, to the user's experience. The new definition aligns gamification with motivational studies. A key aspect is that gamification must include setting the environment so that the users are comfortable in playing the game. The workplace must be an environment in which team members are comfortable in playing the game.

The experiment was completed in three phases. The first phase was data capture of existing historical data. Two major projects were used with each one containing three sub-projects. The original intention of the experiment was to use this data to identify a trend in the data. However, the data did not support any obvious trends. The main reason for this was that the data captured in the tool was not complete. The project teams had focused on delivery rather than maintaining the tracking tool. As a result of the poor quality of data, a new phase of the experiment was introduced. This phase



focused on monitoring the iteration to ensure that the data was maintained. The approach used was to extract data nightly and present it to the team as part of the daily Scrum. The variance between capacity and actuals was highlighted. High levels of outstanding effort were also raised for discussion. This activity was completed at the end of the Scrum, giving the team members time to highlight issues before they were raised by the monitor. The impact of this was significant, the actuals rose from 69% to 87% in one iteration. Having completed the iteration with monitoring in place, the next step was to run an iteration with a gamification experiment. The experiment was a simple lottery game, with the aim of decreasing the size of the tasks and as a result improving the accuracy of the estimates. The lottery prize was an award which had been selected from a list prepared and agreed with the management team. The team were surveyed for the reward that they felt was most valuable. The team was informed on how to play the game and the purpose of the game. The lottery draw occurred at the end of the iteration, and the prize was awarded.

#### **10.4.2 Evaluation**

The results of the experiment were inconclusive. The monitoring of the iterations appeared to have a larger impact than the gamification. Although for the gamified iteration the average task size decreased to 1.9 hours, this could have been related to the specific stories and defects that were included in the iteration, rather than as a result of the gamification.

A large part of the experiment was directed by the quality of data in the tracking tool. If the team was mature to the point of realizing the value of the tracking tool then it may be easier to run a gamification experiment. Without the data quality, the experiment focused on establishing a baseline to run the gamification experiment. As a result, the gamification element of the experiment was limited to one iteration.

The experiment needs to run in an environment in which gamification is seen as the norm. The environment into which the gamification was introduced was one which was focused on delivery. The other aspects of Agile, such as “sustainable development” and focus on technical excellence were ignored in favour of delivery. As a result of this, the team were working in a pressurized system. The introduction of a trivial element such as the lottery would not be relevant, particularly as the

introduction did not coincide with a change in focus from delivery. The experiment effectively looked for the team to “act out of bounds”. (Deterding,S, 2013)

The experiment relied on monitoring the iteration to try to ensure a high level of tracking data. The monitoring was introduced in a manner which was not likely to persuade the team members of the benefit of tracking their work correctly. This was in part due to the timelines of the dissertation. The experiment had only time for two more iterations, if it was to be completed before the dissertation deadline. As such, introducing the monitoring as an external regulation (Deci, E.L. & Ryan, R.M., 2012) was the only viable option. However, it would have been better to introduce the monitoring using a central route of persuasion. In this instance, the problem of the data quality would have been highlighted to the team. The negative impact of not providing valid data would have been clarified. A suggestion would have been made as to how monitoring could help resolve the problem. Volunteers from within the team would have been sought to do the monitoring and a clear timeline would have been identified for when the monitoring would desist.

### **10.4.3 Limitations**

This section examines the limitations of the experiment. The section first looks at the limitations specific to the actual experiments. The section then examines the limitations of gamification.

The experiment was limited to two data warehouse projects and gamification iteration focused on only one. This is a significantly limited selection of projects, even within the organization. It was necessary to limit the experiment to comparable projects, so that the metrics from the project could be compared. However, it is not clear that the characteristics of these projects had an impact on the experiment. For example, a project developed by Java developers might result in a different outcome.

A second limitation of the experiment was that it was time boxed. The monitoring aspect ran in isolation for one iteration. The gamification iteration only ran for the next iteration. As a result, it was not clear if the results were related to the content of the iteration, or were generally as a result of the experiment. It would have been preferable to run the experiment for more iterations.

The experiment was not run in isolation. The experiment was run on a live project. This was evidenced in the interruption to the gamification iteration when a major component was pulled and replaced with a different story. The issue with this was that the team were introducing other elements which could also improve the iteration data. For instance, between the historical iterations and the monitored data, an effort was made to improve the iteration planning. This resulted in improvements to capacity and estimation entry. The results of the experiment were not in isolation and so it is not conclusive that the difference in the metrics is all attributable to the experiment.

The experiment only focused on one aspect of Agile software development. The experiment did not examine whether gamification could assist in other aspects of an Agile project. For instance, gamification could be used in iteration planning, or in requirement discussions. In order to perform a meaningful experiment it was necessary to limit the scope of the experiment to one aspect.

One of the key limitations of gamification is that it needs to operate in an environment in which gaming is acceptable. Rather than using it as a bolt-on to an existing application, as has been done in the experiment, to be a long term successful strategy gamification needs to happen in an environment where the team are comfortable with games as a means of working. The dissertation experiment can be seen as a step in a strategy to adopt such an approach. There is also an aspect where in games it is acceptable to bend the rules while in the workplace it is not acceptable. For example, in football some of the most famous moments result from breaches of the rules going unpunished, Maradona's hand of God in 1986; Thierry Henry's World Cup qualifying handball 2009. There were no repercussions for these players. They had not been seen by the referees and the rules of the game did not allow subsequent punishment based on video evidence. But in the workplace, if a team took something unfairly from another team, then, regardless of whether the issue was captured at the time or at a later date, the issue would likely lead to disciplinary proceedings.

This section has summarized the project and evaluated the results. The section has then examined the limitations of the experiment. The next section provides some future work which could be done to extend the dissertation or the research.

### ***10.5 Future Work & Research***

This section highlights areas for future work which would extend the experiment and areas for further research.

The organization would be assisted if there were customized reports which could be extracted from the tool without the manual effort used in the monitoring iterations. A central warehouse with reports consistent across teams would be beneficial to the organization.

It would be possible to improve the existing experiment by extending the project and running it over a longer period. Using this approach the data retrieved could be compare across periods which lessen the likelihood that the results were related to a difference in the iteration. A further extension to this would be running the gamification in more than one team. This would increase confidence that the gamification impact is related to the experiment and not some external aspect. The final extension of the project would be to extend the project to Agile teams developing software in different environments. The experiment teams build data warehouses for financial applications. The gamification of Agile project tracking could be more or less successful for different project teams who work on different aspects of software development. Running the same experiment with a cross section of development teams and comparing the results may identify characteristics of teams which make gamification more acceptable to them.

In addition to extending the current experiment across time, projects and teams, it would be possible to extend the areas of Agile software development that the gamification is applied to. In the current experiment, the gamification is applied to project tracking. However, it would be possible to apply it to other areas. For instance, Planning Poker could be extended so that it is genuinely gamified. This could be done by applying time pressure, or using a leader board or another game element. The impact of gamification on iteration planning; code review and bug tracking are possible areas of study.

Another area of research would be how to introduce monitoring to a team in a persuasive manner. As discussed in the evaluation, the team benefited from accurate measurement of the project, but still resisted the idea of monitoring to help them to achieve accuracy. This presents an interesting research topic in the area of software motivation. Further research would also be possible in the area of software developers

and project tracking. Even without monitoring, the team is responsible for completing the tracking. However, there is not much appetite for this within the project. Providing an understanding of why these tasks are not completed would be beneficial research. Finally, although not included in this experiment, there is no clear indication of the impact of the culture of the participants on the acceptance of gamification. The team members in the experiment had different acceptance levels for the experiment and this may have been related to the importance of games and play in their culture.

This completes the future work and research section. The next section concludes on the dissertation.

### ***10.6 Conclusion***

The gamification experiment results proved inconclusive. However, the dissertation was still beneficial. The experiment highlighted the poor quality of the data in the project tracking tool. The experiment identified a method of changing this and successfully implemented it. The approach needs to be fine-tuned but the results were a significant improvement on historical data. The experiment also highlighted other areas for improvement including efficiencies within the iteration and communication issues within the team. An experiment methodology was devised and this can be used in future experiments which run over longer periods or with more project teams.

## **GLOSSARY**

**Actuals:** This is the amount of time completing a task. It may vary from the estimated task time.

**Capacity:** This is the amount of work the team can commit to during an iteration. This will allow for holidays, meetings and other time spent working off the project.

**Defect:** This is an issue related to a story. The defect may be related to an issue in the requirements, design, development or build. Defects are also recorded when the environment is down.

**Estimate:** This is a team member's guess as to the size of effort required complete a task.

**Gamify:** To gamify an application is to add an element of gaming to the application with the intent to alter the behavior of the users.

**HCI:** Human Computer Interface, an area of study and design related to how people interact with computers. This includes both hardware and user interface design

**IID:** Interactive and iterative development. This form of software development involves delivering working software frequently.

**Iteration:** This is the period of time which is used to track the work effort. This is a fifteen day period in the experiment projects.

**STD:** Self-Determination Theory. This is a motivation theory which divides extrinsic motivation into sub-categories to explain why some forms of motivation can be demotivating. It also includes amotivation, which is the lack of motivation.

**Story:** Sometimes called a user story. This is a collection of requirements which are combined to provide an usable change or feature to the customer.

**Task:** This is smallest unit of work within a story. Tasks are assigned to one person. In the experiment project, tasks are between one and four hours in duration.

## BIBLIOGRAPHY

- Ahmad, M.O., Markkula, J. & Oivo, M. 2013, "Kanban in software development: A systematic literature review", *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on IEEE*, , pp. 9.
- Avedon, E. 1981, "The structural elements of games", *The psychology of social situations. Selected readings*, , pp. 11-17.
- Awad, M. 2005, "A comparison between agile and traditional software development methodologies", *University of Western Australia*, .
- Basil, V.R. & Turner, A.J. 1975, "Iterative enhancement: A practical technique for software development", *Software Engineering, IEEE Transactions on*, , no. 4, pp. 390-396.
- Beck, K. 1999, "Embracing change with extreme programming", *Computer*, vol. 32, no. 10, pp. 70-77.
- Boehm, B.W. 1991, "Software risk management: principles and practices", *Software, IEEE*, vol. 8, no. 1, pp. 32-41.
- Boehm, B.W. 1988, "A spiral model of software development and enhancement", *Computer*, vol. 21, no. 5, pp. 61-72.
- Boehm, B.W. & Ross, R. 1989, "Theory-W software project management principles and examples", *Software Engineering, IEEE Transactions on*, vol. 15, no. 7, pp. 902-916.
- Boehm, B. & Turner, R. 2005, "Management challenges to implementing agile processes in traditional development organizations", *Software, IEEE*, vol. 22, no. 5, pp. 30-39.
- Bridgeland, J.M., DiIulio Jr, J.J. & Morison, K.B. 2006, "The silent epidemic: Perspectives of high school dropouts.", *Civic Enterprises*, .

- Cockburn, A. & Williams, L. 2000, "The costs and benefits of pair programming", *Extreme programming examined*, , pp. 223-247.
- Construction.com, 2015, [http://construction.com/CE/CE\\_images/0801x12.jpg](http://construction.com/CE/CE_images/0801x12.jpg)  
Accessed June 2015
- Costa, J.P., Wehbe, R.R., Robb, J. & Nacke, L.E. 2013, "Time's up: studying leaderboards for engaging punctual behaviour", *Proceedings of the First International Conference on Gameful Design, Research, and Applications*ACM, , pp. 26.
- Couger, J.D. & Zawacki, R.A. 1980, *Motivating and managing computer personnel*, Wiley.
- Csikszentmihalyi, M. & Csikzentmihaly, M. 1991, *Flow: The psychology of optimal experience*, HarperPerennial New York.
- Deci, E.L. 1971, "Effects of externally mediated rewards on intrinsic motivation.", *Journal of personality and social psychology*, vol. 18, no. 1, pp. 105.
- Deci, E.L. & Ryan, R.M. 2012, "Motivation, personality, and development within embedded social contexts: An overview of self-determination theory", *The Oxford handbook of human motivation*, , pp. 85-107.
- Desmet, P.M. & Pohlmeier, A.E. 2013, "Positive design: An introduction to design for subjective well-being", *International Journal of Design*, 7 (3), 2013, .
- Deterding, S. 2014, "Eudaimonic design, or: Six invitations to rethink gamification", .
- Deterding, S., Dixon, D., Khaled, R. & Nacke, L. 2011, "From game design elements to gamefulness: defining gamification", *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*ACM, , pp. 9.
- Egenfeldt-Nielsen, S. 2007, "Third generation educational use of computer games", *Journal of Educational Multimedia and Hypermedia*, vol. 16, no. 3, pp. 263-281.



- El Emam, K. 2008, "A replicated survey of IT software project failures", *Software, IEEE*, vol. 25, no. 5, pp. 84-90.
- Enterprise Gamification, [http://www.enterprise-gamification.com/index.php?option=com\\_content&view=category&layout=blog&id=26&Itemid=37&lang=en](http://www.enterprise-gamification.com/index.php?option=com_content&view=category&layout=blog&id=26&Itemid=37&lang=en), ACCESSED JUNE 2015
- Fairley, R.E. & Willshire, M.J. 2005, "Iterative rework: The good, the bad, and the ugly", *Computer*, vol. 38, no. 9, pp. 34-41.
- Fernandes, J., Duarte, D., Ribeiro, C., Farinha, C., Pereira, J.M. & da Silva, M.M. 2012, "iThink: a game-based approach towards improving collaboration and participation in requirement elicitation", *Procedia Computer Science*, vol. 15, pp. 66-77.
- Fowler, M. & Highsmith, J. 2001, "The agile manifesto", *Software Development*, vol. 9, no. 8, pp. 28-35.
- Frese, M. 2001, "Personal initiative (PI): The theoretical concept and empirical findings", *Work motivation in the context of a globalizing economy*, , pp. 99-110.
- Gagné, M. & Deci, E.L. 2005, "Self-determination theory and work motivation", *Journal of Organizational Behavior*, vol. 26, no. 4, pp. 331-362.
- Gilb, T. 1977, *Software metrics*, Winthrop Publishers.
- Ginsburg, K.R., American Academy of Pediatrics Committee on Communications & American Academy of Pediatrics Committee on Psychosocial Aspects of Child and Family Health 2007, "The importance of play in promoting healthy child development and maintaining strong parent-child bonds", *Pediatrics*, vol. 119, no. 1, pp. 182-191.
- Glass, R.L. 2006, "The Standish report: does it really describe a software crisis?", *Communications of the ACM*, vol. 49, no. 8, pp. 15-16.

- Grenning, J. 2002, "Planning poker or how to avoid analysis paralysis while release planning", *Hawthorn Woods: Renaissance Software Consulting*, vol. 3.
- Hackman, J.R. & Oldham, G.R. 1980, "Work redesign", .
- Hall, N.G. 2012, "Project management: Recent developments and research opportunities", *Journal of Systems Science and Systems Engineering*, vol. 21, no. 2, pp. 129-143.
- Haugen, N.C. & Moløkken-Østvold, K. "Combining estimates with planning poker", .
- Haugen, N.C. & Moløkken-Østvold, K. "Combining estimates with planning poker", .
- Herzberg, F. 2003, "One more time: How do you motivate employees?", *Harvard business review*, vol. 81, no. 1, pp. 87-96.
- Herzberg, F. 1966, "Motivate Employees?", *World*, , pp. 88.
- Herzig, P., Jugel, K., Momm, C., Ameling, M. & Schill, A. 2013, "GaML-A modeling language for gamification", *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing* IEEE Computer Society, , pp. 494.
- Hossain, E., Babar, M.A. & Paik, H. 2009, "Using scrum in global software development: a systematic literature review", *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on* IEEE, , pp. 175.
- Huotari, K. & Hamari, J. 2012, "Defining gamification: a service marketing perspective", *Proceeding of the 16th International Academic MindTrek Conference* ACM, , pp. 17.
- Javdani, T., Zulzalil, H., Ghani, A.A.A., Sultan, A.B.M. & Parizi, R.M. 2013, "On the current measurement practices in agile software development", *arXiv preprint arXiv:1301.5964*, .
- Jeffries, R., Anderson, A. & Hendrickson, C. 2001, *Extreme programming installed*, Addison-Wesley Professional.

- Johnson, L., Smith, R., Willis, H., Levine, A., & Haywood, K. (2011). The 2011 Horizon Report. Austin, Texas: The New Media Consortium. Retrieved from <http://net.educause.edu/ir/library/pdf/HR2011.pdf>, ACCESSED JUNE 2015
- Jurison, J. 1999, "Software project management: the manager's view", *Communications of the AIS*, vol. 2, no. 3es, pp. 2.
- Kanfer, R. 1987, "Task-specific motivation: An integrative approach to issues of measurement, mechanisms, processes, and determinants", *Journal of social and clinical psychology*, vol. 5, no. 2, pp. 237-264.
- Karlesky, M. & Vander Voord, M. 2008, "Agile project management", *ESC*, vol. 247, pp. 267.
- Ke, F. 2009, "A qualitative meta-analysis of computer games as learning tools", *Handbook of research on effective electronic gaming in education*, vol. 1, pp. 1-32.
- Kerzner, H.R. 2013, *Project management: a systems approach to planning, scheduling, and controlling*, John Wiley & Sons.
- Khan, Z. 2014, "Scrumban-Adaptive Agile Development Process: Using scrumban to improve software development process", .
- Laanti, M. 2014, "Characteristics and Principles of Scaled Agile" in *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation* Springer, , pp. 9-20.
- Larman, C. & Basili, V.R. 2003, "Iterative and incremental development: A brief history", *Computer*, , no. 6, pp. 47-56.
- Larman, C. & Basili, V.R. 2003, "Iterative and incremental development: A brief history", *Computer*, , no. 6, pp. 47-56.
- Llagostera, E. 2012, "On gamification and persuasion", *SB Games, Brasilia, Brazil, November 2-4*, vol. 2012, pp. 12-21.

- Locke, E.A. & Latham, G.P. 1990, "Work motivation and satisfaction: Light at the end of the tunnel", *Psychological science*, vol. 1, no. 4, pp. 240-246.
- Maslow, A.H. 1943, "A theory of human motivation.", *Psychological review*, vol. 50, no. 4, pp. 370.
- McClarty, K.L., Orr, A., Frey, P.M., Dolan, R.P., Vassileva, V. & McVay, A. 2012, "A literature review of gaming in education", *Gaming in education*, .
- Munassar, N.M.A. & Govardhan, A. 2010, "A comparison between five models of software engineering", *IJCSI*, vol. 5, pp. 95-101.
- Naceur, A. & Schiefele, U. 2005, "Motivation and learning—The role of interest in construction of representation of text and long-term retention: Inter-and intraindividual analyses", *European Journal of Psychology of Education*, vol. 20, no. 2, pp. 155-170.
- Petty, R.E. & Cacioppo, J.T. 1986, *The elaboration likelihood model of persuasion*, Springer.
- PMBOK 5th Edition, <http://www.pmi.org/PMBOK-Guide-and-Standards/pmbok-guide.aspx>, ACCESSED JUNE 2015
- Porter, L.W. & Lawler, E.E. 1968, "Managerial attitudes and performance", .
- Reeves, B. & Read, J.L. 2013, *Total engagement: How games and virtual worlds are changing the way people work and businesses compete*, Harvard Business Press.
- Riemenschneider, C.K., Hardgrave, B.C. & Davis, F.D. 2002, "Explaining software developer acceptance of methodologies: a comparison of five theoretical models", *Software Engineering, IEEE Transactions on*, vol. 28, no. 12, pp. 1135-1145.
- Royce, W.W. 1970, "Managing the development of large software systems", *proceedings of IEEE WESCON* Los Angeles, , pp. 328.
- Ruparelia, N.B. 2010, "Software development lifecycle models", *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8-13.

- Rupp, A.A., Gushta, M., Mislavy, R.J. & Shaffer, D.W. 2010, "Evidence-centered design of epistemic games: Measurement principles for complex learning environments", *The Journal of Technology, Learning and Assessment*, vol. 8, no. 4.
- Rus, I. & Lindvall, M. 2002, "Guest editors' introduction: Knowledge management in software engineering", *IEEE Software*, , no. 3, pp. 26-38.
- Scheff, T.J. 2003, "Shame in self and society", *Symbolic interaction*, vol. 26, no. 2, pp. 239-262.
- Schwaber, K. 2000, "Against a Sea of Troubles: Scrum Software Development", *Cutter IT journal*, vol. 13, no. 11, pp. 34-39.
- Schwaber, K. 1997, "Scrum development process" in *Business Object Design and Implementation* Springer, , pp. 117-134.
- Sharp, H., Baddoo, N., Beecham, S., Hall, T. & Robinson, H. 2009, "Models of motivation in software engineering", *Information and Software Technology*, vol. 51, no. 1, pp. 219-233.
- Singer, L. & Schneider, K. 2012, "Influencing the adoption of software engineering methods using social software", *Software Engineering (ICSE), 2012 34th International Conference onIEEE*, , pp. 1325.
- Singer, L. & Schneider, K. 2012, "It was a bit of a race: Gamification of version control", *Games and Software Engineering (GAS), 2012 2nd International Workshop onIEEE*, , pp. 5.
- Sjöberg, D.I., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A., Koren, E.F. & Vokác, M. 2002, "Conducting realistic experiments in software engineering", *Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium nIEEE*, , pp. 17.
- STANDISH CHAOS REPORT,  
[http://www.standishgroup.com/sample\\_research\\_files/chaos\\_report\\_1994.pdf](http://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf),  
ACCESSED JUNE 2015

- Treude, C. & Storey, M. 2010, "Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds", *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*IEEE, , pp. 365.
- Usman, M., Mendes, E., Weidt, F. & Britto, R. 2014, "Effort estimation in agile software development: A systematic literature review", *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*ACM, , pp. 82.
- Vargo, S.L. & Lusch, R.F. 2004, "Evolving to a new dominant logic for marketing", *Journal of Marketing*, vol. 68, no. 1, pp. 1-17.
- Wieggers, K. 2000, "Stop promising miracles", *Software Development*, vol. 8, no. 2, pp. 49-53.
- Yeret,y, <http://yuvalyeret.com/2012/04/28/so-what-is-scrumban/> ACCESSED JUNE 2015
- Zichermann, G. & Cunningham, C. 2011, *Gamification by design: Implementing game mechanics in web and mobile apps*, " O'Reilly Media, Inc."

## APPENDIX A

### Story Extract Fields

Accepted Date	The date the story was accepted
Creation Date	The date the story was created
Dev Complete Date	The date the development tasks were or will be completed
Formatted ID	The story identifier
Kanban State	The state the story is in;
Plan Estimate	The estimate for the story;
Target Release Date	The release date for the story
Task Actual Total	The total actuals for the story;
Task Estimate Total	The total estimates for this story;
Task Remaining Total	The total remaining or to do hours for this story;
Iteration	The iteration associated with the story;
Owner	The story owner;
Release	The release associated with the story.

### Task level data extracts.

Accepted Date	The date the defect was accepted
Creation Date	The date the defect was created
Dev Complete Date	The date the development tasks were or will be completed
Formatted ID	The defect identifier
Kanban State	The state the defect is in;
Plan Estimate	The estimate for the defect;
Target Release Date	The release date for the defect
Task Actual Total	The total actuals for the defect;
Task Estimate Total	The total estimates for this defect;

Task Remaining Total	The total remaining or to do hours for this defect;
Iteration	The iteration associated with the defect;
Owner	The defect owner;
Release	The release associated with the defect.

Task level data for iterations. This is used in calculating the actual versus estimates, velocity and rework metrics.

Iteration	The iteration associated with the task;
Owner	The task owner;
Task ID	The identifier for the task
Name	The quick description of the task
Creation Date	The date the task was updated
Last Update Date	The date the task was last updated
State	The state of the task;
Estimate	The estimate for the task;
To Do	The amount of work outstanding on the task;
Actuals	The actual effort to complete the task.

Capacity: The capacity for the iteration:

Iteration	The iteration associated with the capacity;
User	The user who the capacity refers to;
Creation Date	The date the capacity was created;
Capacity	The capacity for the iteration for the user;
Load	The load on the user. It is the estimates as a percentage of the capacity;



Task Estimates	The estimates associated with the user for the iteration.
----------------	---

## APPENDIX B

### Financial Statistics Data Warehouse Scrum Master interview.

To gather this information and interview was conducted with the scrum master

Can you describe the iteration cadence?

- *Daily Stand up - Daily*
- *Groom 2 times an iteration*
- *Planning and estimating: 2 days which is over the recommended from Agile of 8 hours for an iteration of this size.*
- *Demo – to BSAs and to product Bas before UAT*
- *Retrospective's are done in China and Ireland separately*
- *Iteration Committal. Extra meeting to discuss what was complete and what will be done as part of the next iteration, This is done after the planning and estimates*

How do you think this is working?

Not Working 1 - 5 Working Very Well

- *3 - Working reasonably well*

What is going well in the iteration?

- *Team are getting ownership and accountability. And are now aware of their committal*
- *Metrics are starting to work well. We are able to see what is happening in the iteration but some need to be captured better. We can explain the spillover from one iteration where in the past we were not able to do this*

What would you change in the iteration?

- *Global Model*
  - *Currently some people working within it and other working outside of it. There is a sense that sites are looking after themselves*
  - *Should do a review and get a buy in from everyone*

- *Staffing*
  - *Need to hire people where they can use their skills. Communications and cultural awareness*
  - *Need to ensure that the technical skill people have are correctly aligned with their work*
- *Automation*
  - *Need to look at the automation of deployment and releases as we use a lot of time in completing these tasks*

What are the major issues you are finding with the team?

- *Metrics are incorrect because the actuals were not entered correctly*
- *Estimates and actuals vary substantially. Estimates are not using the actuals that we did previously for similar tasks.*
- *Communications in scrum: People are not calling out the impact to committal. Instead the team are delivering a run through of what they have completed*
- *Still firefighting issues*
- *Environments are not stable*
- *Release management. There is no backlog. We have no visibility of what is coming to the team. This makes it impossible for the development team to do release planning.*

How likely do you think it would be that gamification will help the team?

Not likely 1 - 5 Very likely

- *4 – Yes I believe that would be beneficial. We may need to pick change agents and see can they influence the remainder of the team.*

### **Financial Cost Allocation Data Warehouse Scrum Master interview.**

To gather this information and interview was conducted with the scrum master

Can you describe the iteration cadence?

- *Planning and Estimating meeting*
- *Daily Stand up - Daily*
- *Weekly refinement meeting*
- *Design Review Meeting, once an iteration*
- *Local Site , pre-planning meeting*

- *Retrospective's are done in China and Ireland together*

How do you think this is working?

Not Working 1 - 5 Working Very Well

- *3 - Working well give the time constraints*

What is going well in the iteration?

- *Release Management, had a strong release manager in place. They made the effort to get involved and it made a big difference.*

What would you change in the iteration?

- *Planning not enough done. Work is just passing through.*
- *Commitment ensure stories are ready for commitment*
  - *Definition use of ready and definition of done adhered to*
  - *Add more guardrails*
  - *Extend the planning and refinement meetings*
- *At this stage they are trying to get the team running correctly first but would like to focus more on metrics*

What are the major issues you are finding with the team?

- *Cultural Differences*
  - *Trying to get the team to be more self sufficient. Currently they are very dependent on the technical lead. They seem reluctant to take on work and lead to mistakes. They need to empower the team;*
  - *Not a self-organizing team. If one member is struggling and others have little or no work, then the other team member will not help or reorganize their work;*
  - *Focus on quality over quantity. Currently the team are likely to deliver 90% of 9 stories rather than 100% of 3. There is a focus on quantity at present;*
- *Improve communications to be more effective;*

- *Improve troubleshooting skills. Trace back from problem rather than wasting 3 to 4 hours waiting for someone else to come in and do the same;*
- *Metrics are manual and need to automate the process.*

How likely do you think it would be that gamification will help the team?

Not likely 1 - 5 Very likely

- *4 – Yes I believe that would be beneficial. Particularly could be used as an icebreaker.*

## APPENDIX C

### Team member Metrics

This section shows the number of team members associated with each of the projects

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	N/A	N/A	N/A	N/A	N/A	N/A
Project B	N/A	N/A	N/A	N/A	N/A	N/A
Project C	0	14	16	12	16	11
Project D	7	7	13	12	13	11
Project E	13	10	11	10	8	11
Project F	18	21	16	15	15	14

**Table 41: The number of users who have capacity in the project. Project A and B did not set capacity for the users.**

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	12	14	16	21	20	15
Project B	12	18	8	18	0	5
Project C	10	15	16	15	14	13
Project D	13	7	15	14	17	16
Project E	16	6	4	5	11	9

Project F	20	18	18	15	15	16
-----------	----	----	----	----	----	----

**Table 42: The number of users who had estimates in the project**

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	12	13	14	21	18	13
Project B	11	17	8	14	1	5
Project C	13	10	13	15	14	14
Project D	13	7	12	13	15	14
Project E	13	6	4	4	11	9
Project F	20	17	18	15	14	15

**Table 43: The number of users who had actuals in the project**

**Time zone differences**

This section of the document highlights the time zones differences between the development team and the technology leadership.

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	7 hours	7 hours	7 hours	7 hours	7 hours	7 hours
Project B	8 hours	8 hours	8 hours	8 hours	8 hours	8 hours
Project C	8 hours	8 hours	8 hours	8 hours	7 hours	7 hours
Project D	8 hours	8 hours	7 hours	7 hours	7 hours	7 hours
Project E	8 hours	8 hours	8 hours	8 hours	8 hours	7 hours

E						
Project	7 hours	7 hours	7 hours	7 hours	7 hours	7 hours
F						

**Table 44: The time difference between the technical leadership and the development team**

**Holiday metrics**

This section shows the holidays in both sites during the iteration

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	0	0	0	0	1	5
Project B	0	0	0	1	0	3
Project C	0	0	3	0	1	1
Project D	0	0	1	1	1	0
Project E	0	0	0	1	5	0
Project F	0	0	0	0	3	0

**Table 45: The public holidays in the development site**

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	0	0	1	0	0	0
Project B	1	0	1	2	0	0
Project C	0	2	0	0	1	1
Project D	1	0	1	1	1	0



Project E	0	1	0	0	0	1
Project F	0	1	0	0	0	1

**Table 46: Public Holidays in the technical leadership site**

## APPENDIX D

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	No Iteration
Project A	18	28	35	16	16	9	7
Project B	9	0	40	11	13	6	11
Project C	8	11	9	8	12	10	7
Project D	5	9	16	5	11	9	19
Project E	6	3	3	1	9	5	25
Project F	22	22	20	22	29	20	30

**Table 47: Shows the number of stories in each iteration for each project analysed. The table also includes the count of stories not assigned an iteration.**

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	No Iteration
Project A	0	1	2	14	10	10	2
Project B	8	6	0	2	1	1	10
Project C	1	2	7	1	6	4	12
Project D	1	1	16	2	5	17	3
Project E	3	0	2	1	0	3	22

Project F	12	36	25	17	16	5	15
-----------	----	----	----	----	----	---	----

**Table 48: Shows the number of defects in each iteration for each project analysed. The table also includes the count of defects not assigned an iteration.**

The high number of stories and defects which are not in an iteration shows that the team is not successfully completing the targeted requirements or that the tool was not being used correctly to capture the results. There was a change in policy for the definition of defects in Project F. Previously defects were captured only after the deployment was complete and had passed basic testing. In Project F it was decided to capture issues in deployment as defect as they were impacting on delivery times.

Having extracted the data for use, the next step was to make the data suitable for use in the project.

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	10	9	19	4	1	0
Project B	5	0	9	0	3	6
Project C	1	1	7	1	6	4
Project D	0	1	1	0	0	0
Project E	3	1	3	0	1	4
Project F	13	37	24	18	18	5

**Table 49: Shows the number of stories and defect that did not have estimates assigned. These are filtered out as they cannot be used in the metrics.**

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	9	8	9	4	2	1

Project B	4	0	15	0	1	4
Project C	1	0	3	0	1	3
Project D	0	2	2	0	1	0
Project E	2	1	2	0	1	3
Project F	9	12	10	7	5	3

**Table 50: Shows the number of stories and defects with missing actuals**

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6
Project A	18	25	30	13	14	8
Project B	5	0	26	11	16	8
Project C	7	12	9	8	12	10
Project D	5	7	14	5	10	9
Project E	6	2	2	2	8	4
Project F	21	21	21	21	27	17

**Table 51: Shows the combined number of stories and defects used to produce the charts.**