# Beyond Continuous Integration: Factors Influencing Adoption of Continuous Delivery in Irish SMEs

Grahame Todd

A dissertation submitted to the University of Dublin in partial fulfilment of the requirements for the degree of MSc in Management of Information Systems

***1st September 2015***

## Declaration

I declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university. This research has been carried out in compliance with the research ethics protocol of the School of Computer Science and Statistics.

Signed: _____

## Permission to Lend or Copy

I agree that the School of Computer Science and Statistics may lend or copy this dissertation upon request.

Signed: _____

## Acknowledgements

I would like to thank my supervisor Paula Roberts for all her help, guidance and words of encouragement during the research and writing process.

To those who participated in this research, thank you for giving up your time to share your knowledge and experiences.

Thanks to all my family and to my parents for their support.

Most of all, I would like to thank my wife Alison for all her incredible support, help and patience during the research process… and for keeping me smiling throughout.

## Abstract

Modern software companies are under pressure to develop high-quality software faster than ever before, while keeping risks to a minimum. Adding to this pressure, experimentation and rapid feedback are playing an increasingly important role in software product development, allowing the most agile companies to gain a competitive advantage.

Continuous Delivery is a promising addition to modern software development practices, offering practitioners the ability to deliver quality software to the customer in shorter iterations, while learning and experimenting with every release.

This research investigates the extent of Continuous Delivery use in small to medium-sized companies in Ireland, and uncovers factors influencing its adoption. A large variation in Continuous Delivery adoption was observed. The research establishes that a company's total number of employees, and its number of software-related employees, significantly influence Continuous Delivery adoption. Furthermore, the research shows that there is a significant difference between Continuous Delivery practices used in the development of mobile software, and those used in the development of web-based software. Finally, a number of additional factors that influence Continuous Delivery adoption emerged, which included: software development culture; the use of legacy systems; personal perceptions of Continuous Delivery practices; and the alignment of business processes with software development and delivery processes.

Companies adopting Continuous Delivery should consider not only the associated technical aspects, but also the organisational constraints in the form of culture; business process; and employee perception; in order to fully realise Continuous Delivery's potential.

## Table of Contents

## List of Tables and Diagrams

### List of Tables

**List of Diagrams**

## Abbreviations

| | |
|---|---|
| CD | Continuous Delivery |
| CDMM | Continuous Delivery Maturity Model |
| CI | Continuous Integration |
| IES | Innovation Experiment System |
| KPA | Key Process Area |
| LSD | Lean Software Development |
| SME | Small to Medium-sized Enterprise |

# 1 Introduction

## 1.1 Background

Continuous Delivery (CD) is a set of principles and practices which aim to sustainably and quickly deliver incremental changes to users in a low-cost, low-risk manner (Humble and Farley 2010; Humble 2014). CD promotes a method of software development in which software is kept in a state that allows it to be released to the customer at any time (Chen 2015a; Fowler 2013). The term *Continuous Delivery* originates in the first principle of the Agile Manifesto, which promotes iterative software development and "continuous delivery" of value to customers (Beck et al. 2001; Cockburn 2006).

Research shows that adoption of CD practices can help companies to gain a competitive advantage by making product experimentation practical and enabling near-immediate feedback from product iterations (Leppanen et al. 2015; Karvonen et al. 2015). This research investigates the extent to which SMEs in Ireland have adopted CD practices and explores the factors which influence adoption. The study was conducted between December 2014 and August 2015.

## 1.2 Research Context

Software-producing businesses are under increasing pressure from consumers, competitors and advances in technology to produce and deliver software quickly and reliably, while keeping costs and risk exposure low (Olsson et al. 2012; Forrester Research 2014, p.3). Many businesses have adopted Agile development methods in an attempt to increase the speed of software delivery, but Agile alone may not be sufficient to meet the innovation and quality needs of modern businesses (Akerele et al. 2013; Forrester Research 2013, p.23). Furthermore, the increasing use of software product experimentation in competitors of all sizes is a growing concern for modern software businesses (Bosch 2012; Fagerholm et al. 2014; Ries 2011).

## 1.3 Objective and Scope of the Research

The scope of this research is limited to software-producing SMEs in Ireland. The research seeks to determine the level of adoption Continuous Delivery practices in Irish SMEs and aims to discover the factors that influence adoption.

## 1.4 Research Questions

This study addresses the following research questions:

- *To what extent have small to medium-sized companies in Ireland adopted practices that contribute to Continuous Delivery?*

This question will address the extent of adoption of CD practices by investigating software development at companies in the target demographic.

- *What factors influence adoption of Continuous Delivery within small to medium-sized companies in Ireland?*

It is intended that this research will reveal demographic factors that predict CD adoption by a given organisation, as well as factors that promote or prevent adoption within organisations.

## 1.5 Relevance of this Research

While literature exists describing Continuous Delivery specifically in the context of startups (Ries 2011) and large enterprises (Chen 2015a; Feitelson et al. 2013; Claps et al. 2015), little research exists which specifically investigates adoption of CD in the SME demographic.

The lack of CD research aimed at SMEs influenced the decision to limit the research to companies of this size. Additionally, it has been suggested that smaller businesses are at an advantage when implementing CD (Karvonen et al. 2015; Ries 2011), provoking further interest in this area.

Finally, with recent research showing that SMEs employ 68% of the workforce and represent 99.7% of active enterprises in Ireland (Central Statistics Office 2014), a research focused on SMEs has considerable practical relevance.

## 1.6 Beneficiaries

This research may be of interest to current and prospective practitioners of Continuous Delivery, especially those in SMEs. However, the findings of this research could be used to assist in the effective implementation of CD in an organisation of any size. The research may also assist in rating an existing CD implementation, by providing a model for assessment. The study may be of interest to researchers of CD and those involved in related areas; the findings of this research could be used as the basis for future work in the field.

## 1.7 Dissertation Roadmap

Chapter 1 presents the context of the research and provides background information on the research topic. The objectives and scope of the research are outlined and the research questions are presented. This chapter also notes the relevance of the research and lists its potential beneficiaries.

Chapter 2 reviews and critically examines relevant literature in the research area. The chapter defines CD and outlines its benefits. CD promises a number of benefits, both for software development teams and organisations. Accounts of real-world implementations of Continuous Delivery are explored and it is placed in context with modern software development practices. Adoption of CD is discussed and models of the practice are evaluated.

Chapter 3 discusses the methodological approaches that were considered; presents the philosophy, approach and strategy of this research; and justifies the methodological choices that were made. The research design is presented, the model used is outlined and the rationale for the chosen design are presented. This chapter also discusses sampling and data collection methods used in the study. The overall methodology of the research is evaluated, presenting a consideration of its shortcomings and the methods used to

counteract these. Finally, ethical considerations are discussed and lessons learned from the research process are outlined.

Chapter 4 presents and analyses quantitative data gathered in this research, showing demographic data; findings related to adoption of CD practices; and findings related to factors influencing adoption of CD practices.

Chapter 5 discusses the major themes emerging from qualitative data analysis and evaluates these in the context of Continuous Delivery adoption ratings.

Chapter 6 highlights the findings of the research and discusses how the research questions were answered. The study's findings are summarised and examined in the context of existing literature. Limitations of the research are also considered. Finally, opportunities for future research are explored and the conclusions of the research are clearly presented.

# 2 Literature Review

## 2.1 Introduction

This chapter aims to provide a detailed analysis of published works related to software release and Continuous Delivery. An explanation of CD is presented alongside its benefits and commercial reasons for adoption. Real-world case studies and reports of CD implementations are also explored.

In this chapter, CD is placed in context with modern software development practices. Existing studies of CD adoption are also considered.

Finally, the chapter examines research related to Continuous Delivery Maturity Models, which have been developed to measure Continuous Delivery adoption.

## 2.2 Continuous Delivery

### *2.2.1 Introduction to Continuous Delivery*

In the context of software, Humble et al. (2015, p.156) describe Continuous Delivery as "the ability to get changes – experiments, features, configuration changes, bug fixes – into production or into the hands of users safely and quickly, in a sustainable way." Chen elaborates further, describing Continuous Delivery as "a software development discipline where the software is built in such a way that it can be released to production at any time." (Chen 2015a).

The concept was first applied in this manner by Humble and Farley (2010), who use the term "Continuous Delivery" from The Agile Manifesto (Humble and Farley 2010, p.xxiv). The first principle of The Agile Manifesto states: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." (Beck et al. 2001).

Continuous Delivery is achieved through a number of practices and processes which combine to enhance the software development process as a whole. These include (Humble and Farley 2010, p.419):

1. Continuous integration of software changes.

2. Control over creation and management of software environments and deployment of software to these environments.
3. Detailed monitoring and continuous improvement of release and compliance processes.
4. Extensive software test automation.
5. Mature data management procedures and processes.
6. Version control of software changes in a manner that supports the above processes.

The goal of these processes is to make software development and release reliable, predictable and transparent, while enabling swift delivery of completed software to the customer (Humble and Farley 2010, p.xxiv). CD can therefore be seen to reduce a number of risks that are inherent in the software development process. Additionally, CD aims to make developing and releasing software in smaller batches viable (Humble et al. 2015, p.156), the benefits of which are discussed in section 2.3.

### *2.2.2 Continuous Integration and the Deployment Pipeline*

Continuous Delivery builds on the practice of Continuous Integration; a process in which code changes are continually integrated with a known-good version and any resulting problems are immediately acted upon. This concept was first described by Booch (1991, p.209), who later described an "evolution" of software implementation (1994, p.246). Evolution involved a number of internal releases, with each release addressing problems identified in those prior to it. The overall process was described as "a sort of continuous integration of the system." (Booch 1994, p.256). Continuous Integration became a key practice of Extreme Programming (XP) in the late 1990's; Beck (1998) stated that CI and associated automated testing techniques allowed developers to be more "aggressive", and recommended that code should be integrated within a few hours of being written (Beck 2000, p.97). According to later assertions by Beck, integrating frequently means that programmers notice bugs early, reducing project risk. (Beck 2000, p.98).

Expanding on CI, CD proposes a deployment pipeline (see Figure 2.1), which Humble and Farley describe as "continuous integration taken to its logical conclusion." (Humble and Farley 2010, p.4). The deployment pipeline builds upon CI by subjecting the code in each integration event to a series of automated tests, which culminate in an automated software deployment stage. Humble and Farley describe the process as "fundamentally, an automated software delivery process." (Humble and Farley 2010, p.109).

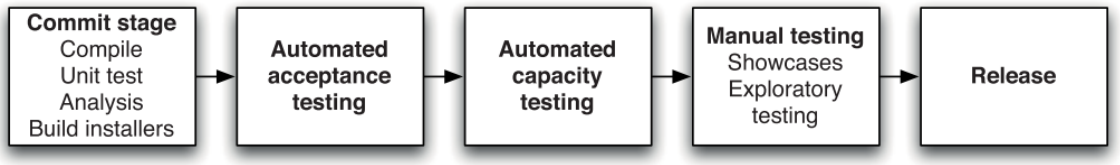


FIGURE 2.1 – The deployment pipeline (Humble and Farley 2010, p.4)

The pipeline consists of four core stages, as adapted below from Humble and Farley (Humble and Farley 2010, p.110):

1. Commit Stage (Version Control)

At this point, code is submitted by the developer. Unit tests and code analysers are typically run.

2. Automated Acceptance Test Stage(s)

This stage may consist of a series of functional and non-functional testing, to ensure that software operates as intended. Figure 2.1 contains two testing phases at this stage; automated acceptance testing and automated capacity testing.

3. Manual Test Stages

Verification that the application meets the needs of the users and delivers value as intended. This stage may include exploratory testing and user acceptance tests, for example.

4. Release Stage

This stage delivers software to users. It often consists of automated deployment of software to servers, but may alternatively include preparation of packaged software.

The deployment pipeline rejects software that fails at any stage of the process. The sequence diagram in Figure 2.2 illustrates this, showing how changes are intended to move through the pipeline. Shaded boxes in this diagram indicate events taking place at each of the four stages in the pipeline, as described above, including a feedback loop to the delivery team at each stage. When a software build passes all testing phases and reaches the

release stage, it is ready to be deployed and released to customers, if the company so wishes.

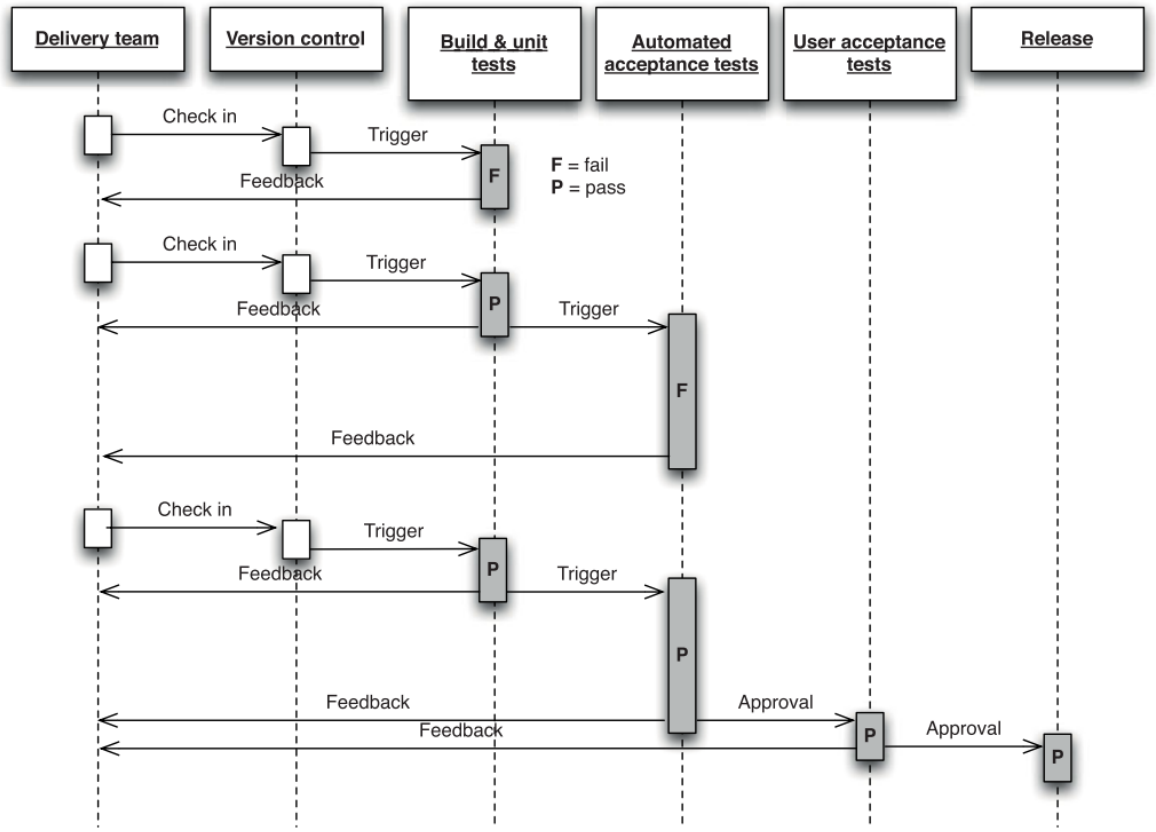


Figure 2.2 – Software changes traversing a deployment pipeline. (Humble and Farley 2010, p.109)

### *2.2.3 Continuous Deployment*

Software builds that have passed the testing stages of the deployment pipeline (see section 2.2.2) are deemed fit for release (Humble and Farley 2010, p.24). However, Continuous Delivery does not require that all passing builds be deployed to the production environment (Humble et al. 2015, p.156; Humble and Farley 2010, p.266). Some practitioners extend the practice of Continuous Delivery by allowing all passing builds to immediately progress to the production environment via an automatic deployment process; a practice is known as Continuous Deployment (Humble and Farley 2010, p.266). In this sense, Continuous Deployment implies Continuous Delivery.

The concept of automatic deployment is not new (Coupaye and Estublier 2000), but the additional practices that Continuous Delivery encompasses can make deployments truly valuable to organisations and software development teams, as outlined in section 2.3.

## 2.3 Benefits of Continuous Delivery

With recent research describing Continuous Delivery as the “Holy Grail” (Bellomo et al. 2014) and its adoption as the “Stairway to Heaven” (Olsson et al. 2012), it is evident that researchers think highly of the practice. In this section, the benefits of Continuous Delivery are examined in order to understand why it might be held in high regard; and why it might be sought after by an organisation.

### *2.3.1 Benefits for IT and Software Development Practitioners*

As software development practices and the systems they produce evolve, so too must the approaches taken to delivering software (Bosch 2010). CD promises a number of benefits to software development teams that can help to address commonly-encountered issues in modern software development. These benefits and their theoretical roots are examined in this section.

- Reduced Software Defects

It is evident that the deployment pipeline’s multiple stages of testing allow for greater confidence in the software itself: regression tests have been shown to be effective in detecting faults (Memon and Xie 2004; Wong et al. 1997) and automated acceptance tests are effective in confirming that the system operates as intended (Haugset and Hanssen 2008). The pipeline can additionally be customised to add further tests; for example, capacity testing can be employed, as seen in Figure 2.1. In this sense, the pipeline can be customised to the needs of the organisation.

Fitzgerald and Stolt (2014) examined a Continuous Delivery implementation by Neely and Stolt (2013), and compared aspects of Continuous Delivery to the Lean concept of “poka-yoke” (translated as *mistake-proofing or fool-proofing*). In this analogy, the software delivery process has been *fool-proofed* due to comprehensive testing that runs after code integration to verify its integrity, reducing the risk of producing unsatisfactory software.

- Reduced Deployment Risk

The software deployment process has traditionally been a point of high risk. Even in modern systems, van der Burg (2010) notes that deployments of systems adhering to a Service Oriented Architecture can be particularly troublesome, describing deployment as "complex and laborious." The recent emergence of Microservices as an architectural choice has the potential to complicate software deployment further, since a Microservices architecture divides a system into smaller, single-purpose services, which are intended to be deployed separately (Thönes 2015; Fowler 2014).

The risk of releasing software is addressed by Continuous Delivery in two ways. First, Humble and Farley advocate the adoption of automated software deployment and configuration practices (Humble and Farley 2010, p.419). Changes often have to be made to production systems in order to install and configure software; using the same automated process on pre-production and production environments ensures the process is robust (Humble and Farley 2010, p.25). Secondly, Continuous Delivery encourages an increased frequency of software release. This counterintuitive practice reduces risk by ensuring releases occur in smaller batches; by reducing the number of changes in each release, the overall risk is reduced (Humble et al. 2015, p.168; Claps et al. 2015).

- Reduced Cycle Time

Recent research by Forrester, commissioned by IBM, showed that of 600 IT professionals with application development responsibilities in Europe and North America, 87% reported pressure to reduce development cycle times in their organisation (Forrester Research 2014, p.17).

Cycle time refers to the average time taken to progress from one end of a process to the other (Poppendieck and Poppendieck 2003, p.77) or, in the particular case of software development, "the path from idea to realised business value" (Humble and Farley 2010, p.xxiv). With regard to cycle time, it is interesting to note the similarities between Continuous Delivery and research carried out into Lean Software Development (LSD). Although LSD has a broader scope (see section 2.4.1), the effects of shorter cycle times and smaller batches on the development process have been investigated.

As noted in section 2.2.2, Continuous Delivery promotes small, frequent changes which make their way through the deployment pipeline's stages in order; code changes enter the pipeline as small, frequent batches of change. Ferreira and Langerman (2014) researched

the effects of the release management process on throughput in software development projects; findings showed that a frequent release strategy increased overall throughput. Ferreira and Langerman cite Poppendieck and Poppendieck (2009), stating that smaller batches also result in higher-quality software.

$$Cycle\ Time = \frac{Average\ Number\ of\ Items\ in\ Queue}{Average\ Completion\ Rate}$$

FIGURE 2.3 – A simplified version of Little's Law, adapted from (Allen 1990, p.259).

Little's Law (Allen 1990, p.259) provides academic support for the findings of Ferreira and Langerman (2014). Figure 2.3 shows that cycle times can be reduced by increasing the average completion rate, or reducing the average number of items in the queue.

Reducing batch size in the deployment pipeline would naturally increase the average number of items in the pipeline, but Poppendieck and Poppendieck (2003, p.80) suggest that multiple smaller batches will be faster to process than one large batch, due to increased resource utilisation. Figure 2.4 illustrates the effect of queue resource utilisation and batch size on cycle time. This graph represents the application of Queueing Theory to the software development process.

The example used by Poppendieck and Poppendieck in Figure 2.4 is a queue for software testing in an organisation; as noted in section 2.2.2, this may be a stage in the deployment pipeline. In this context, the graph can be understood to show that when the testing department is running at 50% resource utilisation for a queue receiving large batches, a given batch may take 25 hours. However, at 85% utilisation, this time increases to 100 hours. By reducing batch sizes, Poppendieck and Poppendieck suggest that throughput of the queue – in this case throughput of the testing department – will not suffer until almost 90% utilisation is reached, meaning that the queue can run continuously at a higher capacity. A department that can continuously operate at a higher capacity is undoubtedly of greater use to management and therefore to the organisation itself.

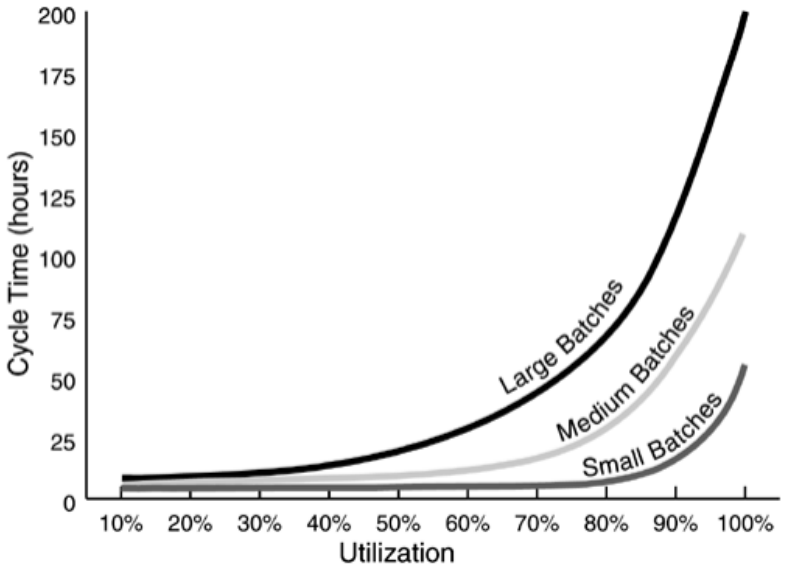


FIGURE 2.4 – The effect of resource utilisation and batch size on cycle time. (Poppendieck and Poppendieck 2003, p.80).

- Development Transparency

It is interesting to note that of the "Eight Key DevOps Practices" noted by Forrester (Forrester Research 2014, p.6), four are observed to be Continuous Delivery practices:

1. Deliver in small increments of functionality
2. Automate environment provisioning
3. Continuously integrate code
4. Continuously test

Additionally, Krusche et al. (2014) note that obtaining feedback from the development process has recently become possible through Continuous Delivery, contributing to another Forrester recommendation; "provide real-time transparency". (Forrester Research 2014, p.6). Forrester recommends transparency into development, claiming that it provides insight into progress and risk, while maintaining productivity.

*2.3.2 Benefits for Organisations*

Forrester claims that pressure from consumers and competitors is driving organisations to be able to experiment with new software ideas within a timeframe of a few days, while keeping costs low and risk to a minimum (Forrester Research 2014, p.3). This claim is supported by Bosch (2012), who observes an important change in the development and deployment of software products and services, driven by both customer expectations and technological advances. In response to this, Bosch proposes the use of "R&D as an innovation experiment system (IES)" and defines the first step in this process to be a software development approach of continuously and frequently deploying new versions (Bosch 2012).

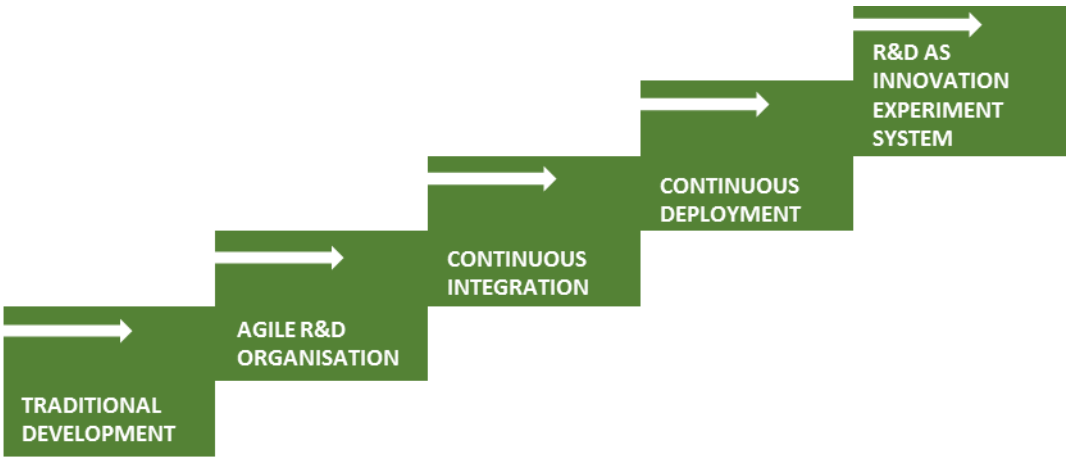


FIGURE 2.5 – The "Stairway to Heaven" model, as proposed by Olsson et al. (Olsson et al. 2012). Adapted from Karvonen et al. (2015).

Bosch contributed to further research that proposed a model of practices which lead to the use of R&D as an IES (Olsson et al. 2012). Continuous Deployment was listed as a key step, implying Continuous Delivery (as discussed in section 2.2.3). The proposed "Stairway to Heaven" model can be seen in Figure 2.5. Although the research that proposed this model found that no companies had reached the level of R&D as an IES (Olsson et al. 2012), recent research by Leppanen et al. (2015) has shown that it is indeed achievable. Leppanen et al. (2015) found that Continuous Delivery acts as a driver for R&D as an IES.

The IES combines a swift, iterative approach to software development with customer feedback and usage data from as many ideas as possible, in order to positively impact customer satisfaction and drive revenue, benefitting the organisation (Bosch 2012). This experimental approach to product development is also advocated by Humble et al. (2015, p.171).

It has been suggested that, even without R&D as an IES, improvement of IT practice through Continuous Delivery can benefit the organisation as a whole, though more rigorous research may be needed in this area. Forrester observes that "software success is increasingly indistinguishable from business success." (Forrester Research 2014, p.3). A report by Puppet Labs indicates a strong correlation between IT performance and the use of Continuous Delivery practices (Velasquez et al. 2014). In considering the business value of these practices, the report additionally draws on data from previous research by Puppet Labs (Puppet Labs 2013), stating a strong confidence that "high IT performance correlates with strong business performance, helping to boost productivity, profitability and market share." (Velasquez et al. 2014, p.10). It should be noted that Puppet Labs has a direct commercial interest in promoting Continuous Delivery practices, though the research has been carried out by academics and well-known experts in the field of Continuous Delivery.

### *2.3.3 Real-World Implementations*

Sections 2.3.1 and 2.3.2 have shown theoretical backing for Continuous Delivery practices and outlined reports from collective studies. Complementing these, a number of case studies have been conducted, illustrating some real-world benefits of CD.

Chen (2015a) describes the introduction of CD practices in Irish company Paddy Power, which operates in the bookmaking industry. The company relies heavily on custom software running on thousands of servers worldwide. Chen's work outlines and evaluates the implementation of CD practices, and use of a deployment pipeline in the development of 20 applications within the organisation.

In adopting CD practices, the following "huge benefits" were reported, as adapted from (Chen 2015a):

1. Accelerated Time-to-Market
    - Release frequency increased to one release per week. Chen notes that each application was released once every two months before CD adoption.

2. Faster Product Feedback
    - Due to increased release frequency, user feedback was obtained early and product decisions could be made based on this. This is similar to the concept of "R&D as an IES" (Bosch 2012), discussed in section 2.3.2.
3. Improved Productivity / Efficiency
    - Chen reports that developers previously spent 20% of their time configuring test environments. These environments are now configured automatically by the deployment pipeline.
4. Reliable Releases
    - Chen reports a reduction of risk and increased reliability in the software deployment process.
5. Improved Product Quality
    - A large reduction in the number of bugs was observed after the transition to CD.
6. Improved Customer Satisfaction
    - Customers in this case were other areas of the business. Chen reports a reestablishment of trust with these areas, which had previously waned due to issues with software quality and release.

An interesting observation by Chen is that "engineers do not feel the same level of stress ... on the release day. The release day becomes just another normal day." This lowering of stress is also reported as a benefit of Continuous Delivery by Humble and Farley (2010, p.20). While Chen considers the overall implementation to be a success, it is noted that implementation of CD did result in organisational, process and technical challenges. Despite these, Chen reports that the company will prioritise investment and adoption of CD across the organisation.

Further real-word Continuous Delivery experiences are described by Neely and Stolt (2013), who evaluated the implementation of CD practices in SaaS-focused company, Rally Software. In short, Neely and Stolt report "greater control and flexibility over feature releases, incremental delivery of value, lower risks, fewer defects, easier on-boarding of new developers, less off-hours work, and a considerable uptick in confidence." Neely and Stolt note that the company was already considered to be an expert in Agile development; the iterative nature of Agile works well with Continuous Delivery (see section 2.4.2) and may have helped the company's transition. (Neely and Stolt 2013).

Summarising software development and deployment at Facebook, Feitelson et al. (2013) describe the use of Continuous Delivery to drive innovation, but state that Facebook

deliberately stops short of Continuous Deployment. According to Feitelson et al., this is due to privacy concerns over customer data and the need for increased oversight. As a result, the company uses a combination of daily and weekly deployments, which Feitelson et al. (2013) describe as balancing innovation and risk. Deliberately delaying release allows for greater testing, which fits into the "manual testing" stage of Humble and Farley's (2010, p.4) deployment pipeline (see section 2.2.2). Humble and Farley note that the deployment pipeline allows practices such as deployment authorisation and auditing to be enforced without great effort, and without affecting the efficiency of the delivery process. (Humble and Farley 2010, p.437).

## 2.4 Continuous Delivery and Modern Software Development Practice

### *2.4.1 Lean Software Development*

Lean Software Development is based on seven principles, which are presented below in their simple form with short explanations, quoted directly from Poppendieck and Poppendieck (2003, p.66):

1. Eliminate waste: Spend time only on what adds real customer value.
2. Amplify learning: When you have tough problems, increase feedback.
3. Decide as late as possible: Keep your options open as long as practical, but no longer.
4. Deliver as fast as possible: Deliver value to customers as soon as they ask for it.
5. Empower the team: Let the people who add value use their full potential.
6. Build integrity in: Don't try to tack on integrity after the fact—build it in.
7. See the whole: Beware of the temptation to optimize parts at the expense of the whole.

Continuous Delivery takes influence from Lean Software Development (as seen in Figure 2.6) and addresses many of Lean Software Development's core principles directly.

Allowing the business to serve the customer's needs quickly is an *elimination of waste* (Poppendieck and Poppendieck 2009, p.xxv). Rapid releases enabled by Continuous Delivery (Chen 2015a; Neely and Stolt 2013) help to eliminate waste in this manner. CD helps to *amplify learning* by incorporating feedback into the development phase (Krusche et al. 2014) and providing early customer feedback on software changes (Chen 2015a).

In Lean Software Development, the principle *decide as late as possible* states that there is benefit in delaying commitment to a decision until the "last responsible moment", which is

described as “the moment at which failing to make a decision eliminates an important alternative.” This often allows decisions to be made on facts, rather than on speculation (Poppendieck and Poppendieck 2003, p.56).

Delaying until the last responsible moment, according to Poppendieck & Poppendieck, is enabled by developing in short iterations, with feedback. (Poppendieck and Poppendieck 2003, p.57). Continuous Delivery encourages development in small batches (see section 2.3.1), allowing for rapid feedback from iterative development. (Krusche et al. 2014; Olsson et al. 2012).

Poppendieck & Poppendieck note that the principle *deliver as fast as possible* complements the principle *decide as late as possible*. (Poppendieck and Poppendieck 2003, p.70). Continuous Delivery shortens cycle times, which enables this. (See section 2.3.1).

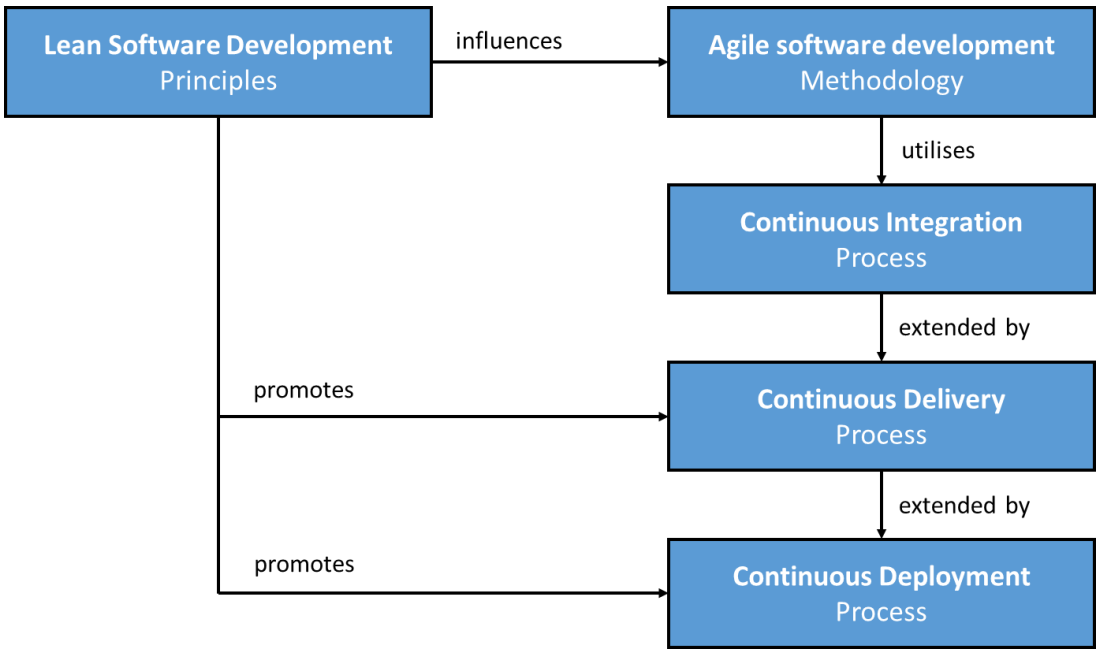


FIGURE 2.6 – Concept map of the CD and Continuous Deployment processes, showing the influence of Lean Software Development. Adapted and expanded from Claps et al. (2015).

Humble and Farley address the principle *Build Integrity In*, referring to this principle as “Build Quality In” (Humble and Farley 2010, p.27). Humble and Farley state that many CD practices (for example, Continuous Integration, comprehensive automated testing, and

automated deployment) are specifically designed to find defects early in the development process. This early feedback alerts developers, who are able to fix defects as early as possible; ensuring overall software quality. (Humble and Farley 2010, p.27). This assertion is supported in findings by Neely and Stolt (2013), Chen (2015a) and Krusche et al. (2014).

Finally, Continuous Delivery's consideration of the entire delivery process, from development to release (see section 2.2.2) relates to the Lean practice of *optimising the whole*. Humble and Farley state that "a holistic approach is necessary that ties together every part of the delivery process and everybody involved in it." (Humble and Farley 2010, p.xxv).

In addition to the parallels outlined above, Fitzgerald & Stol (2014) argue that Continuous Integration and Continuous Delivery both have strong links to Lean concepts.

### *2.4.2 Agile Software Development*

As noted in section 2.2.1, the name "Continuous Delivery" originates in the first principle of The Agile Manifesto (Humble and Farley 2010, p.xxiv), which states, "our highest priority is to satisfy the customer through early and continuous delivery of valuable software." (Beck et al. 2001). Continuous Delivery's fully-tested changes in shorter cycle times (see section 2.3.1), as well as implicit early feedback to the organisation on those changes, support this goal of delivery speed and value. This principle is the "highest priority" of Agile; making Continuous Delivery an excellent fit to support the methodology. (Beck et al. 2001)

Humble's prior work describes the "Deployment Production Line" (Humble et al. 2006) – a precursor to Continuous Delivery. Humble suggests that deployment automation "embodies a key agile practice – making your code (in this case your deployment scripts) your documentation."

In a study conducted for Forrester Research, Hammond et al. (2011, p.2) claim that Agile alone is not sufficient to enable rapid software delivery. Hammond et al. (2011, p.12) subsequently recommend Continuous Delivery practices and the adoption of the deployment pipeline concept.

### *2.4.3 Scrum*

Hossain et al. (2009) describe Scrum as "an iterative and incremental project management approach" and note that it is a popular Agile approach. In Scrum, each iteration is called a "Sprint", lasting 2-4 weeks. The Scrum Guide (Schwaber and Sutherland 2013) states that the product of each Sprint must be useable, but is not necessarily released. This is similar to CD's notion of software that can be released to production at any time; however, CD does not advocate time-constrained "Sprints". Scrum focuses on project management and does not include any technical practices (Fowler 2009), such as those advocated by CD.

As noted by Scrum.org (Gfader 2012, p.6), the use of Continuous Delivery alongside the Scrum process allows the cadence of Sprints and releases to be separated. CD can even facilitate release of the software before all Product Backlog items in a particular Sprint are complete. (Gfader 2012, p.6).

Krusche at al. (2014) criticise the time-constrained nature of the Sprint cycle in Scrum, especially during requirements elicitation, stating that not allowing requirements to change during a sprint cycle is overprotective. According to Krusche et al. requirements should be clarified during Sprints if necessary, and developers should be able to deploy evolving versions of the software as the Sprint progresses, in order to receive feedback. Addressing this, Krusche et al. (2014) propose "Rugby", which is described as "an agile process model based on continuous delivery". The model leverages Continuous Delivery in its main workflow. Krusche et al. claim Rugby improves communication across teams; improves communication with the customer; provides transparency; and aids requirements elicitation.

### *2.4.4 DevOps and Release Engineering*

As described by Dyck et al. (2015), both DevOps and Release Engineering are concerned with delivering high-quality software while managing change. Dyck et al. propose the following definitions:

> "Release engineering is a software engineering discipline concerned with the development, implementation, and improvement of processes to deploy high-quality software reliably and predictably."

> "DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organisations, in order to operate resilient systems and accelerate delivery of changes."

Cois et al. (2014) note that "iterative software development processes can make use of…DevOps concepts and tools to enable internal continuous delivery of software", while Puppet Labs (Velasquez et al. 2014) lists Continuous Delivery as a well-known DevOps practice. Humble and Farley state that Continuous Delivery and the DevOps movement have the same goals. (Humble and Farley 2010, p.28).

Continuous Delivery promotes many practices related to IT operations, such as configuration management, provisioning of environments and deployment of software. (Humble and Farley 2010, p.419). However, Continuous Delivery also promotes development practices, including software version control, Continuous Integration and unit testing; showing that the philosophy of Continuous Delivery spans both development and operations, supporting the DevOps movement (Humble and Farley 2010, p.419). Likewise, Continuous Delivery is concerned with the reliable release of high-quality software (Humble and Farley 2010, p.28), placing Continuous Delivery as a practice of both DevOps and Release Engineering.

## 2.5 Continuous Delivery Adoption

There is little academic research aimed at gaining insight into common adoption rates of Continuous Delivery, though some case studies do exist (see section 2.3.3). It is the researcher's belief that no research exists that attempts to examine widespread CD adoption in Ireland.

Olsson et al. (2012) proposed the "Stairway to Heaven" model, describing the evolution of an organisation on the way to using R&D as an experiment system (see section 2.3.2), and also explored the barriers to Continuous Deployment, as experienced by four companies. Case studies were used to explore practices at participating companies and data was gathered via semi-structured interviews. The study found various levels of adoption and no participating company had achieved Continuous Delivery.

Similar research was conducted very recently, which involved two of the original researchers. Karvonen et al. (2015) used case studies to investigate practices at five Finnish companies and found that four had reached the level of Continuous Integration, while only one had achieved Continuous Deployment. However, the study proposes that customer resistance to Continuous Deployment is a barrier to adoption. This interesting criticism of CD is echoed by Blank (2014).

Ries (2011, p.191) states that Continuous Deployment was gaining popularity in startups in 2011, but no research has been found that specifically targets the whole SME category. Combined with recent research, which shows that SMEs employ 68% of the Irish workforce and account for 99.7% of active enterprises in Ireland (Central Statistics Office 2014), this highlights CD in SMEs as an important research area.

## 2.6 Models of Continuous Delivery

While the “Stairway to Heaven” model of “R&D as an IES” (Olsson et al. 2012) features Continuous Deployment and some related processes (see section 2.3.2), it does not provide a complete model of CD. Continuous Delivery Maturity Models (CDMMs) provide a comprehensive model of Continuous Delivery and offer a means of determining a company’s level of adoption.

Poeppelbuss et al. (2011) describe maturity models as “conceptual multistage models that describe typical patterns in the development of organizational capabilities” and note that they are important not only for assessment of organisations, but also in Information Systems research. Maturity models define levels of maturity for a number of key process areas within a given field. Key practices and common features are defined for each key process area at a given maturity level (Paulk et al. 1993; Mettler et al. 2010). Similarly, CDMMs define key process areas, key practices, and common features that comprise Continuous Delivery. The Humble and Farley CDMM, with key process areas; key practices; and maturity levels indicated, can be seen in Appendix 3 – The Humble and Farley Maturity Model.

CDMMs were critically evaluated from Humble and Farley (Humble and Farley 2010, p.419), Forrester Research / ThoughtWorks (Forrester Research 2013), Xebia Labs (Xebia Labs 2015, p.7), Diabol (Rehn et al. 2012), IBM (Minick 2014), Praqma (Praqma A/S 2015, p.6) and Bekk Consulting (Bekk Consulting AS 2013).

Although all CDMMs differed somewhat, some common features were noted.

1. All of the examined CDMMs identified software build practices as a key process area. Testing (or quality assurance) was also identified as a key process area across all examined CDMMs.
2. Reporting and report visibility featured heavily in the CDMMs examined, whether as a key process area (Xebia Labs 2015; Rehn et al. 2012; Minick 2014), or as a key practice (Humble and Farley 2010; Forrester Research 2013).

Interestingly, both the Diabol and Xebia models include culture or DevOps as a separate key process area (Rehn et al. 2012; Xebia Labs 2015). The Bekk Consulting model, similarly, considers "Process & Organisation" separately (Bekk Consulting AS 2013).

The Humble and Farley model is very similar to earlier work by Humble and Russell of ThoughtWorks, which attempted to apply an Agile Maturity Model to the overall process of building and releasing software (Humble and Russell 2009). However, the Humble and Farley model includes additional key practices and maturity levels for the practice of configuration management; see Appendix 3 – The Humble and Farley Maturity Model. A further, more recent ThoughtWorks CDMM was created by Forrester (Forrester Research 2013).

## 2.7 Summary

Continuous Delivery consists of a number of practices and processes that allow organisations to sustainably and quickly deliver incremental changes to users in a low-cost, low-risk manner. These practices build on the foundation of CI and the concept of a staged "deployment pipeline". As a result, software changes are continually integrated and all software changes are subject to a series of increasingly-rigorous tests in controlled environments, prior to release.

CD promises to be a practical and worthwhile solution to real problems faced by software companies today. Research has indicated clear benefits for software development teams and organisations as a whole. Extensive automation throughout the deployment pipeline allows development to take place in small batches, reducing the risk involved with each batch of change. The use of application monitoring and rapid feedback in conjunction with CD's rapid delivery of small changes can enable organisations to use software development

as an experiment system, influencing product development. Real-world implementations of CD practices have shown a wide range of benefits, including improved customer satisfaction; improved product quality; and accelerated time-to-market.

CD is strongly influenced by Lean Software Development and has roots in the Agile Manifesto. Accordingly, CD can work well with Agile methodologies and has been recommended as a solution to some of the shortcomings of using Agile alone. For example, CD has been used with Scrum to separate release and sprint cadences; and further research has extended Scrum to better leverage CD practices.

Academic research on CD adoption is limited, though practitioner research has been conducted by organisations closely involved with CD.

Finally, a number of Continuous Delivery Maturity Models have been produced that enable adoption of CD practices to be measured. Although these models vary somewhat, evaluation showed that common features exist across all models. CDMMs will be discussed further in chapter 3 as the methodology and research model used to address this study's research questions are addressed.

# 3 Methodology and Fieldwork

## 3.1 Introduction

Research philosophies and methodologies that were considered for this research are detailed in this chapter, followed by an explanation and justification of the chosen research methodology, strategy and approach.

This chapter also discusses the research design and data collection methods that were used to support the chosen strategy. Existing Continuous Delivery Maturity Models (CDMMs) are considered and the model used in this research is presented.

Finally, the chapter outlines the sampling and data collection processes, before considering potential issues with the chosen methodology and ethical issues. Lessons that were learned while conducting the research are also presented.

## 3.2 Research Philosophies

Saunders et al. (2012, p.127) state that research philosophies relate to "the development of knowledge and the nature of that knowledge" and note that the philosophy adopted can be thought of as the researcher's assumptions about their view of the world. (Saunders et al. 2012, p.128). Saunders et al. (2009, p.108) emphasise the importance of these intrinsic assumptions, referencing work by Johnson & Clark (2006): "as business and management researchers we need to be aware of the philosophical commitments we make through our choice of research strategy since this has significant impact not only on what we do but we understand what it is we are investigating."

This section will examine philosophies that were considered for this research.

### *3.2.1 Positivism*

Positivists hold an ontological assumption that "an understandable reality is assumed to exist, driven by immutable natural laws" (Khazanchi and Munkvold 2003) and that objective cognition is possible (Becker and Niehaves 2007, p.203). As summarised by Khazanchi & Munkvold (2003), positivism assumes "a belief that only observable things are real and worthy of study" and that "any knowledge claim or scientific explanation must be arrived at by means of sensory experience."

A positivism, then, assumes that an underlying reality exists independent of human interpretation and that knowledge of reality can be acquired through measurement and experimentation.

### *3.2.2 Interpretivism*

In contrast to positivism, Khazanchi and Munkvold note that interpretivism is built upon on an "epistemological notion that objective observation is not possible" (Khazanchi and Munkvold 2003). Interpretivists subscribe to the ontological assumption that "the social world is produced and reinforced by humans through their action and interaction." (Khazanchi and Munkvold 2003).

Therefore, in the interpretivist view, humans and human behaviour cannot be separated from reality. As a result, interpretivist research looks to explain human behaviour, rather than discover an underlying, independent and generalisable truth.

### *3.2.3 Realism*

Direct realism (or "naïve realism") states that the senses provide humans with a direct awareness of an independent, external reality. Critical realism contests this "direct" awareness, taking into account the cognitive processes that occur in human understanding of external reality.

For the critical realist, then, while external reality does exist, it is subject to "value-laden observation" (Dobson 2002). Our knowledge of reality through experience and research is a result of social conditioning; therefore, this knowledge must be understood by taking into account the social actors that were involved in deriving that knowledge (Dobson 2002).

### *3.2.4 Pragmatism*

In outlining pragmatism, Saunders et al. (2009) state "pragmatism argues that the most important determinant of the epistemology, ontology and axiology you adopt is the research question." Tashakkori and Teddlie (1998) suggest that it is productive for researchers to

consider quantitative and qualitative research as a spectrum; noting that research combining these designs is possible.

That is to say, there is no fundamentally correct research approach; an appropriate approach or approaches should be adopted to address the research question at hand. Extrapolating from this, a pragmatic philosophy would dictate that an appropriate research method (or an appropriate combination of research methods) should be adopted to address the research question.

### *3.2.5 Selected Research Philosophy*

Pragmatism was chosen as the research philosophy for this research, because the principles of pragmatism are in line with the objectives of this research. The research aims to provide a worthy and constructive contribution to Continuous Delivery literature by appropriately addressing the research questions. Saunders (2012, p.130) quotes Kelemen and Rumens (2008) in stating that pragmatists "use the method or methods that enable credible, well-founded, reliable and relevant data to be collected that advance the research."

As noted in Section 3.2.4, pragmatists adopt the most appropriate approach to address the research question.

## 3.3 Research Questions

The research will address two questions:

1. RQ1: To what extent have small to medium-sized companies in Ireland adopted practices that contribute to Continuous Delivery?
2. RQ2: What factors influence adoption of Continuous Delivery within small to medium-sized companies in Ireland?

Addressing RQ1 considers the extent to which practices related to Continuous Delivery are being used in SMEs in Ireland. These practices are examined further in Section 3.7.1.

RQ2 also addresses SMEs in Ireland, and explores the factors that influence adoption of Continuous Delivery within them.

## 3.4 Research Approach

An inductive approach to research moves from specific observations and empirical data to identify patterns, infer meaning and produce a general theory. A deductive approach, conversely, uses a general theory and moves to produce a hypothesis, which is tested and confirmed through observation and measurement. A third approach, abduction, is less commonly used and is described by Suddaby (2006) as "the process by which a researcher moves between induction and deduction". In this manner, abduction can be seen as a combination of inductive and deductive reasoning.

Peirce, a pragmatist and proponent of abductive reasoning, eloquently compares all three approaches: "Deduction proves that something *must* be, Induction shows that something *actually* is operative, Abduction merely suggests that something *may* be." (Peirce and Houser 1998, p.216).

This research addressed RQ1 using a deductive approach, while the research undertaken to address RQ2 used elements of deduction, induction and, ultimately, abduction. Further details on the research method used can be seen in section 3.6.

## 3.5 Research Strategy

Saunders et al. argue strongly that the researcher should choose the strategy that best suits the research that is being undertaken, while taking into account other practical factors, such as the extent of existing knowledge, time available and access to potential participants (Saunders et al. 2012, p.173)

After evaluating alternatives as detailed below, the survey strategy was chosen. The survey was administered in the form of a combined structured and semi-structured interview. These interviews therefore gathered quantitative and qualitative data to address both research questions. Further details on the research method used can be seen in section 3.6.

### *3.5.1 Strategies Considered*

Saunders et al. (2012) outline a number of strategies that are available when conducting business research. Each of the following was considered with regard to its suitability in addressing the research questions.

- Ethnography
- Action Research
- Case Study
- Archival Research
- Experiment
- Survey
- Narrative Enquiry
- Grounded Theory

### *3.5.2 Ethnography*

Ethnography involves deep, observational social study and is mainly focused on description above explanation (Babbie 2012, p.333). Saunders (2012, p.181) notes that ethnography is focused on social studies of groups within which subjects interact and share physical space. Taking this into account, ethnography was not considered to be well-suited to address the research questions.

### *3.5.3 Action Research*

Action research is typically conducted over the course of an activity, change or process. It is an interactive and iterative research process in that the researcher participates in the change, activity or process, while conducting research (Saunders et al. 2012, p.183).

Action research involving a relevant Continuous Delivery implementation would certainly provide insight into the adoption of Continuous Delivery practices and may also contribute to addressing the research questions. However, due to lack of an opportunity to study such an implementation and the time that would be required, action research was not chosen for this research.

### *3.5.4 Case Study*

Case Studies explore a single example of a topic to be researched. This single case is often studied deeply and therefore case studies may provide a rich awareness of the research topic.

While a case study may have addressed the research questions somewhat, a single case study would not have provided a broad insight into the adoption of Continuous Delivery practices and influencing factors across multiple companies in Ireland.

### *3.5.5 Archival Research*

Archival research – involving the examination of records or documents (Saunders et al. 2012, p.178) – was not considered applicable for this research due to lack of access to appropriate source materials on the research subject.

### *3.5.6 Experiment*

The experiment strategy seeks to investigate how changes in a variable – known as the independent variable (IV) – affect another variable, known as the dependent variable (DV) (Babbie 2012, p.272). Prior to experimentation, the researcher formulates a hypothesis that there will be a significant relationship between the dependent and independent variables. The contrasting null hypothesis states that there will be no such relationship. (Saunders et al. 2012)

Corbetta (2003, p.94) notes that "deliberate human intervention to produce change" – which is required to measure the relationship between the dependent and independent variables – separates the experiment strategy from natural observation of change.

Since this research will not intervene to produce change, the experiment strategy is not appropriate in this case.

### *3.5.7 Narrative Inquiry*

Narrative inquiry examines complete stories provided by participants, which are usually collected during qualitative interviews. Saunders et al. (2012, p.188) note that this strategy is used when the researcher feels there is value in gathering a complete account from participants, instead of a fragmented questioning. In this respect, narrative inquiry differs from grounded theory. For this research, more value is placed on structured and semi-structured questioning to address the research questions.

### *3.5.8 Survey*

The survey strategy is common to business and management research (Saunders et al. 2012, p.176). This is possibly due to the versatility of the strategy; surveys can be used to describe, explain and explore phenomena (Babbie 2012, p.229).

In a further example of the versatility of the survey strategy, Saunders (2012, p.177) notes that while surveys collect quantitative data from respondents for later analysis, they can also provide insight into possible reasons for relationships between variables, and may help in modelling these relationships. Babbie (2012, p.229) states that surveys are "probably the best method available" to the social researcher for collecting data from a population that cannot be directly observed.

It is clear, given these applications of the survey strategy, that it is applicable to this research and is relevant in addressing research question RQ1 in particular.

### *3.5.9 Grounded Theory*

Grounded theory methodology is a strategy which attempts to produce a theory through analysis of observed data. By comparing or coding collected data, the researcher can identify patterns and commonalties that may lead to development of a theory. (Babbie 2012, p.336)

In this sense, grounded theory methodology is inductive. However, the researcher may then – in a deductive manner – test and subsequently refine this theory, hence taking an abductive approach (Saunders et al. 2012, p.186).

Originally described by Glaser & Strauss (1967), the methodology has since evolved, with the two original authors holding different views on the approach that should be taken. Glaser has generally advocated the more inductive approach of concentrating on the observed data and those theories arising from it (Heath and Cowley 2004), while Strauss & Corbin (2008) have taken a more abductive stance (Reichertz 2009).

Constructivist grounded theory (Charmaz 2006) emphases role of the researcher in constructing the theory while analysing data. Summarised by Charmaz (2014), constructivist grounded theory "sees both data and analysis as created from shared experiences and relationships with participants and other forms of data."

Given grounded theory's focus on theory arising from analysis of data, it is applicable to this research; in particular, it can help to address research question RQ2, which is most suited to this approach.

## 3.6 Research Method

Research that exclusively uses a quantitative or qualitative research method is said to employ the "mono method", while "multiple methods research" involves a combination of both of these methods (Saunders et al. 2012, p.160). This research will use a mixed-methods research design, for reasons outlined below. The research questions and existing literature were considered when deciding which research method best suited this research.

### *3.6.1 Consideration of Research Questions*

To answer research question RQ1, it was noted that quantitative data could be used and that the survey strategy was appropriate (see section 3.5). The question can be answered using numerical data and does not need insight into meaning or social constructs, which are usually reasons to use a qualitative research method. (Saunders et al. 2012, p.163)

However, in considering the data required answer research question RQ2, it was noted that multiple methods were appropriate. Factors that influence Continuous Delivery adoption may consist of:

1. Factors implied by the existing literature on Continuous Delivery.
2. Currently-unknown factors.

To confirm whether factors implied by existing literature do indeed influence adoption of Continuous Delivery, quantitative data is appropriate. Data on these factors can be gathered via the survey strategy, in a similar manner to RQ1.

To investigate currently-unknown factors that influence Continuous Delivery adoption, analysis of qualitative data is more appropriate, since the research in this area is exploratory in nature (Babbie 2012, p.90). Grounded theory methodology is suited to this task.

A mixed-methods approach was therefore used for this study. Bryman (2006) investigated how quantitative and qualitative data may be integrated and noted that there many reasons to use mixed-method research. In Bryman's terms, mixed methods will be used in this research to answer *different research questions*, as well as for *enhancement*; *explanation*; *completeness;* increasing *utility* of results; and to *offset* the weaknesses of research methods (Bryman 2006).

### *3.6.2 Consideration of Existing Research*

While considering the existing literature and research, Continuous Delivery Maturity Models (CDMMs) were identified as an excellent rating method which can help to determine an organisation's Continuous Delivery adoption level. Available CDMMs were critically analysed (see section 2.6) and a model was adapted to suit this research (see section 3.7).

## 3.7 Research Design

Combined structured and semi-structured interviews were used to gather data for this research. Interviews consisted of:

1. Structured questions based on an adapted Continuous Delivery Maturity Model (see section 3.7.1). Data gathered by these questions was used to address RQ1.
2. Structured questions based on CD-influencing factors that are implied by existing literature. Data gathered by these questions was used to assist in addressing RQ2.
3. Semi-structured questions to explore currently-unknown factors. Data gathered by these questions provided further data to assist in addressing RQ2.

### *3.7.1 Continuous Delivery Maturity Models*

All available CDMMs were critically evaluated (see section 2.6). After assessment, a model was developed for this research that is heavily based on the original Humble and Farley CDMM (Humble and Farley 2010, p.419), with some input from Forrester's more recent model (Forrester Research 2013). Combining maturity models with appropriate consideration is not uncommon; Poeppelbuss et al. note prior examples of consolidation of maturity models in Information Systems (Poeppelbuss et al. 2011). The Humble and Farley model can be seen in Appendix 3 – The Humble and Farley Maturity Model.

Humble and Farley's model was adjusted to remove a key practice: "automated tests written as part of story development". This was removed due to the fact that measuring a practice that takes place "as part of story development" restricts the model to companies that are following story-based, (usually iterative) methodologies such as Scrum or Extreme Programming. This may not necessarily be the case in all SMEs in Ireland. Although Humble and Farley do advocate iterative development practices for reasons of software quality, they state that the Continuous Delivery practices described are beneficial "to any delivery team, whatever process they use" and indicate that CD practices may be used regardless of methodology. (Humble and Farley 2010, p.194). Therefore, removal of this key practice does not misrepresent their concept of Continuous Delivery.

The removed practice has been replaced with a practice from Forrester's CDMM: "production-like testing environments are available for projects early on" (Forrester Research 2013). This fits appropriately under the "testing" category in both CDMMs; appears in both at the same maturity level; and does not conflict with any of the other practices in the Humble and Farley model.

It is intended that the resulting CDMM is more applicable to SMEs in Ireland, while remaining true to the intentions of the original authors and reflective of Continuous Delivery practices in use today. The CDMM used in this research can be seen in Appendix 4 – Maturity Model Used in this Research.

### *3.7.2 Influencing Factors Implied by Existing Literature*

- Number of Employees and Company Age

Ries (2011, p.191) states that Continuous Delivery was gaining popularity in startups in 2011. Startups initially have smaller numbers of employees; it is possible that CD is easier to achieve with a smaller number of employees. This factor will be investigated, hypothesising that there will be an association between total number of employees and Continuous Delivery adoption. It is similarly hypothesised that there will be an association between number of technical (or software-related) employees and Continuous Delivery adoption. Alternatively, the claims of Ries (2011, p.191) may be due to young companies adopting new practices; a hypothesis is made that there will be an association between company age and Continuous Delivery adoption.

- Software Environment

Humble and Farley note a number of Continuous Delivery-related practices that may be difficult in some environments. For example, a number of activities: deployment automation; data management; and deployment orchestration; would be difficult if software was deployed at a remote site, without network connectivity. It is hypothesised that there will be an association between software environment and adoption of Continuous Delivery practices.

### *3.7.3 Currently-Unknown Factors Influencing CD Adoption*

As noted in section 3.6.1, grounded theory method is an appropriate research method to assist in addressing RQ2. Open-ended, semi-structured interview questions will be used, which can generate qualitative data for analysis using grounded theory method (Charmaz 2006, p.26). Saunders et al. list semi-structured interviews as particularly beneficial in collecting data when questions are open-ended. (Saunders et al. 2012, p.379). In-depth or semi-structured interviews are likely to be used when the researcher needs to understand "attitudes and opinions" or "reasons for decisions", which suits this research (Saunders et al. 2012, p.378).

### *3.7.4 Interview Design*

Following the advice of Saunders et al. for structured interviews (or "interviewer-completed questionnaires"), the structured section of the interview was designed to be clear, appropriate and comprehensive in its coverage of the topic (Saunders et al. 2012, p.452).

Importantly for this research, structured questions were asked before semi-structured questions, so that exploratory inquiry would not influence participants' answers to structured questions. The interview began with a series of structured demographic questions, before moving on to structured and semi-structured questioning on key areas of Continuous Delivery.

Non-demographic, structured questions were based directly on the key practices in the Continuous Delivery Maturity Model described in section 3.7.1. Formulating these questions was generally straightforward; for example, the key practice "database upgrades and rollbacks tested with every deployment" was split into two questions; "do you test database upgrades on every deployment?" and "do you test database rollbacks on every deployment?" The interview questions used can be seen in Appendix 2 – Interview Questions.

The completed interview was tested on two individuals who were not involved with this research, to ensure face validity. (Saunders et al. 2012, p.451). Some explanatory statements were added as a result of these tests to ensure clarity. Testing also helped to increase the researcher's familiarity with the interview, as suggested by Babbie (2012, p.251). The duration of the test interviews was recorded, so that a realistic approximation of interview length could be given to prospective interviewees.

Further actions to reduce bias were taken while administering the interview; these are discussed in section 3.8.

## 3.8 Sampling and Data Collection

The research used convenience sampling to target companies in Ireland of SME size. Senior technical employees with knowledge of the company's release processes were sought as participants, due to the highly-technical nature of the interview questions. Ten interviews were conducted in person by the researcher, at nine participating companies, between March 2015 and May 2015. A total of twelve participants were interviewed. (See Table 3.1.) Interviews lasted approximately one hour each.

TABLE 3.1 – Interview schedule

| Interview | Interview Date | Company Alias | Participant Alias |
|---|---|---|---|
| A | 28th March 2015 | Company A | Participant A |
| B | 6th April 2015 | Company B | Participant B |
| C | 7th April 2015 | Company C | Participant C1 |
| | | | Participant C2 |
| D | 17th April 2015 | Company D | Participant D |
| E | 26th April 2015 | Company E | Participant E |
| F1 | 28th April 2015 | Company F | Participant F1 |
| F2 | 28th April 2015 | Company F | Participant F2 |
| G | 29th April 2015 | Company G | Participant G1 |
| | | | Participant G2 |
| H | 13th May 2015 | Company H | Participant H |
| I | 22nd May 2015 | Company I | Participant I |

Interviews were administered while following advice of Babbie (2012, p.251). To reduce bias, the wording of structured questions was followed exactly. Interview audio was also recorded so that data could be extracted from the recording after the interview, reducing the potential for error. Appearance and demeanour at the interview were also taken into account, as suggested by Babbie.

In an abductive manner, interview F2 was undertaken specifically in order to test a theory that emerged from data analysis through grounded theory method. This is discussed further in chapter 4.

## 3.9 Potential Problems with the Chosen Methodology

1. Sample size

In addressing research question Q1, structured interviews were chosen to collect quantitative data. It was recognised that the time-consuming nature of interviews would result in a smaller sample than would have been possible with questionnaires.

Questionnaires were ruled out due to the highly-technical nature of the questions to be asked. Structured interviews also tend to offer a high completion rate (Saunders et al. 2012, p.421; Babbie 2012, p.250), which was reflected in this research.

2. Interviewer bias

Structured interviews are potentially subject to interviewer bias and this was kept in mind throughout the research.

3. Research method

Semi-structured interviews were chosen as the data collection method to assist in addressing RQ2, with data to be analysed using grounded theory method. An argument could be made against this research method, since time constraints may not allow for a comprehensive grounded theory study to be conducted in the traditional manner.

However, grounded theory method was chosen in order to apply rigour to the analysis of interview transcripts. Analysis did provide an opportunity for unforeseen themes to emerge and allowed the researcher to test an emerging theory in the field.

## 3.10 Research Ethics

Ethical approval was obtained from the Research Ethics Committee at the Trinity College Dublin School of Computer Science and Statistics.

The submission for ethical approval included the following documents:

- Completed ethical approval forms;
- Research proposal;
- Information sheet for individuals;
- Information sheet for organisations;
- Informed consent form for individuals;
- Informed consent form for organisations;
- List of proposed interview questions.

See Appendix 1 – Ethics Application and Supporting Documentation.

Care was taken to ensure that participants understood that all questions were optional, by stating this verbally before each interview began. A full debriefing was also offered and accepted by a number of participants. Care was also taken throughout the course of the research to preserve participant anonymity; Saunders et al. (2012, p.242) note that this is a potential problem for research using interviews as the data collection method.

Ethical approval for the research was granted by the Research Ethics Committee in March 2015.

## 3.11 Lessons Learned

Regarding the use of mixed structured and semi-structured interviews, it was found that participants naturally elaborated on answers to structured questions. In the case of this research, this resulted in additional qualitative data for analysis.

Due to the nature of the organisational questions asked during data collection, ethical approval required consent from the organisation itself, as well as from participants. The researcher found it difficult to recruit participants who were in a position to obtain this consent, which ultimately limited the number of interviews that were conducted. However, the inclusion of these questions and obtaining organisational consent has resulted in higher quality, ethically-sound research.

## 3.12 Summary

This research adopted a pragmatist research philosophy, which influenced the research approach, strategy, research method and data collection methods used. Elements of deductive and inductive approaches were used, and the research ultimately used abduction, testing an emerging theory in the field. (See chapter 4.)

The research questions and existing literature were used to establish a mixed-methods design, using interviews for data collection. In order to gather quantitative data to assess the extent of Continuous Delivery adoption, structured interview questions were formulated based on a Continuous Delivery Maturity Model. Further structured interview questions were formulated based on existing literature, in order to gather data on factors influencing CD adoption. Finally, semi-structured interview questions were devised to gather qualitative data which would be best suited to explore currently-unknown factors that influence CD adoption.

Convenience sampling was used to target high-level technical employees at companies within the target demographic and ethical approval was obtained for the research prior to the commencement of data collection.

Chapter 4 presents the findings and analysis of this research.

# 4 Findings and Analysis

## 4.1 Introduction

The results of data analysis and interpretation are presented in this chapter, showing how the research questions were answered. The chapter first discusses how analysis was carried out for both quantitative and qualitative data. The results of demographic questions are then presented, followed by analysis to determine the extent of CD adoption in participating companies. Some concepts arising from analysis of qualitative data are also presented.

A theory emerged during constant comparative analysis related to the use of CD when developing mobile applications. This theory was tested in the field; the results of this investigation are also presented in this chapter.

A total of ten interviews were conducted: nine directly collected data to answer the original research questions, and one additional interview – Interview F2 – investigated the emerging theory.

## 4.2 Data Analysis

As noted in Chapter 3, this research required quantitative analysis of structured interview responses and qualitative analysis of interview transcripts.

### *4.2.1 Quantitative Data Analysis*

Data was checked before analysis, which revealed that although four some companies declined to report turnover, valid answers were provided for all other questions.

Participants were asked structured questions in order to gauge their company's adoption of Continuous Delivery practices. These questions investigated the company's use of key practices and common features as outlined in the chosen Continuous Delivery Maturity Model (see section 3.7).

Where appropriate, answers to structured questions were analysed to determine where the participant lies within three levels of adoption of each practice:

- Adopted – the participant's company implements the key practice or common feature. Answers of this nature were scored 1.
- Adopting – the participant stated that the company does has not adopted the key practice or common feature, but the company is in the process of adopting it. Answers of this nature were scored 0.5.
- Not adopted – the participant stated that the company has not adopted the key practice or common feature. Answers of this nature were scored 0.

The "adopting" category was not appropriate for all questions; where it was not, only the "adopted" and "not adopted" categories were used. Combining answer scores with answer weightings resulted in a score for a given key practice as seen in Table 4.1. Weightings were intentionally devised to produce integer values for each key practice. Further information on scoring can be seen in the key practice notes in Appendix 6 – Interview Answer Weightings. A matrix was created to calculate key practice scores for all companies, similar to the example seen in Table 4.1.

TABLE 4.1 – Example of scoring for an adopted key practice

| Key Practice No. | Practice | Answer Score | Weighting | Practice Score |
|---|---|---|---|---|
| 11 | Any build can be recreated from source | 1 | 2 | 2 |

Each key practice residing at a positive CDMM maturity level was equally weighted for the purposes of this study. However, when formulating interview questions, it was noted that some key practices in the CDMM implied other practices that were lower in maturity level. (For example, "automated testing on every commit" implies "regular automated testing", since the former is the more frequent practice of the two.) In these cases, the key practices were appropriately treated as mutually exclusive; and the weighting (4) for the practice at the higher maturity level was defined as double that of the lower practice (2). Examples of these weightings can also be seen in Appendix 6 – Interview Answer Weightings.

Calculating total scores for all practices in participating companies allowed for quantitative analysis to be carried out, and the position of each company on the spectrum of Continuous Delivery adoption to be identified. The research stops short of classifying each company's maturity against the model's maturity levels due to the broad ranges that these categories cover. (See Appendix 3 – The Humble and Farley Maturity Model).

Interview F2 was excluded from quantitative data analysis directed at answering the initial research questions, since this additional interview was carried out specifically to test an emerging theory arising from constant comparative analysis.

### *4.2.2 Qualitative Data Analysis*

Qualitative data was analysed using grounded theory method, following constructivist grounded theory as outlined by Charmaz (2006). Coding was applied to the entire interview transcript for all participants. Quantitative data and findings were also taken into account; development of the theory considered all data available. (Charmaz 2014). The following phases were undertaken:

1. Initial Coding

Incident-to-incident coding was used in this research and codes consisted of actions and gerunds where possible, per Charmaz's recommendations (Charmaz 2006, p.136). The research employed coding during the data collection phase, in the hope of directing research as codes emerged (Charmaz 2006, p.48). The constant comparative method was used (Glaser and Strauss 1967; Charmaz 2006, p.54); new data was compared with data in current and previous transcripts – as well as with the emerging theory, categories and codes – as the research progressed.

2. Focused Coding

Focused coding reanalysed and refined codes to cover significant categories. Input from the researcher determined which codes made most analytic sense using previously-coded data (Charmaz 2006, p.57).

3. Memos

Memos were written early in the research process in order to capture the researcher's ideas (Charmaz 2006, p.82). These ultimately highlighted a common concern participants had related to CD in mobile environments.

4. Theory Development

The aim of grounded theory is to produce a theory from the data gathered. Charmaz notes that the researcher's construction of theory depends on what is found in the field (Charmaz

2006, p.135). In this research, a theory was developed regarding the use of CD in mobile development, and was also tested in the field. See section 4.5.4.

## 4.3 Demographic Questions

Participants were asked demographic questions related to their company's business: company size in terms of number of employees and turnover; their role in the company; and the nature of the software developed at the company.

### *4.3.1 Number of Employees*

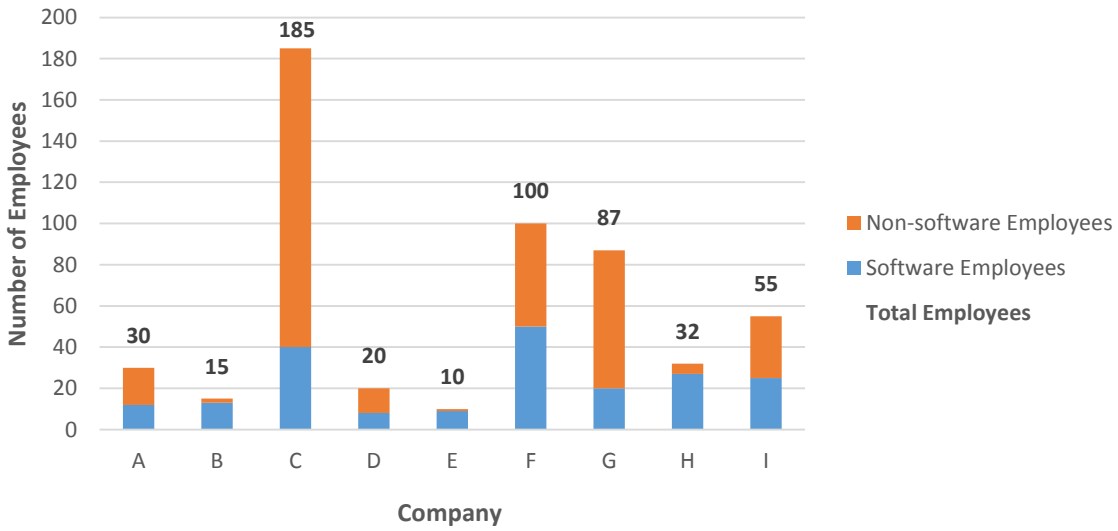


FIGURE 4.1 – Graph showing number of employees in participating companies

Figure 4.1 shows the number of employees reported by each company. Participants were also asked how many employees were involved in the production of software ("software employees") and how many were involved in other activities ("non-software employees"). See Table 4.2.

TABLE 4.2 – Number of employees in participating companies

| Company | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| **Software Employees** | 12 | 13 | 40 | 8 | 9 | 50 | 20 | 27 | 25 |
| **Non-software Employees** | 18 | 2 | 145 | 12 | 1 | 50 | 67 | 5 | 30 |
| **Total Employees** | **30** | **15** | **185** | **20** | **10** | **100** | **87** | **32** | **55** |

*4.3.2 Turnover*

Four of the companies respectfully declined to provide information on turnover. Prior research and the stated number of employees were used to determine that participating companies were approximately of SME size.

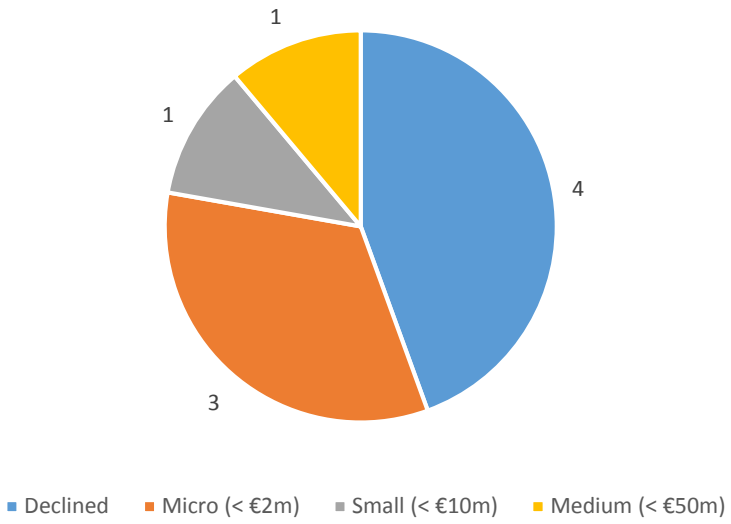


FIGURE 4.2 – Turnover in participating companies

Reported turnover was used to categorise participating companies under the EU-defined categories of “Micro”, “Small” and “Medium” in order to provide a good indicator of company size, while preserving anonymity (European Commission 2005). The results of this categorisation can be seen in Table 4.3, and Figure 4.2, with those companies that declined to provide turnover categorised as “declined”.

TABLE 4.3 – Participating companies and associated EU size category

| Company | EU Size Category |
|---|---|
| **Company A** | Small (< €10m) |
| **Company B** | Micro (< €2m) |
| **Company C** | Medium (< €50m) |
| **Company D** | Declined |
| **Company E** | Micro (< €2m) |
| **Company F** | Declined |
| **Company G** | Declined |
| **Company H** | Declined |
| **Company I** | Micro (< €2m) |

### *4.3.3 Software Produced*

Participants were asked about the nature of the software produced and the environment in which the software was run.

Both interviews at Company F are included in these demographics, and are listed as F1 and F2 in Table 4.4, but Interview F2 was excluded from statistical analysis to address the research question, as explained in section 4.2.1.

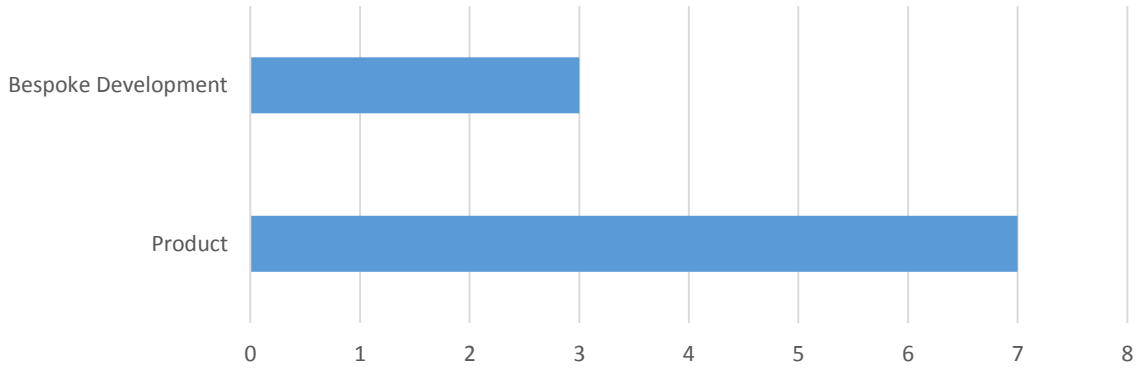


FIGURE 4.3 – Type of software produced in participating companies

The nature of software produced by companies was categorised as either:

1. Bespoke Development: Companies produced bespoke software for their customers in a business-to-business model.

2. Product: Companies produced their own software product or service, which was then provided business-to-business or business-to-consumer.

The number of companies whose software falls into each category can be seen in Figure 4.3, with results shown in Figure 4.4.

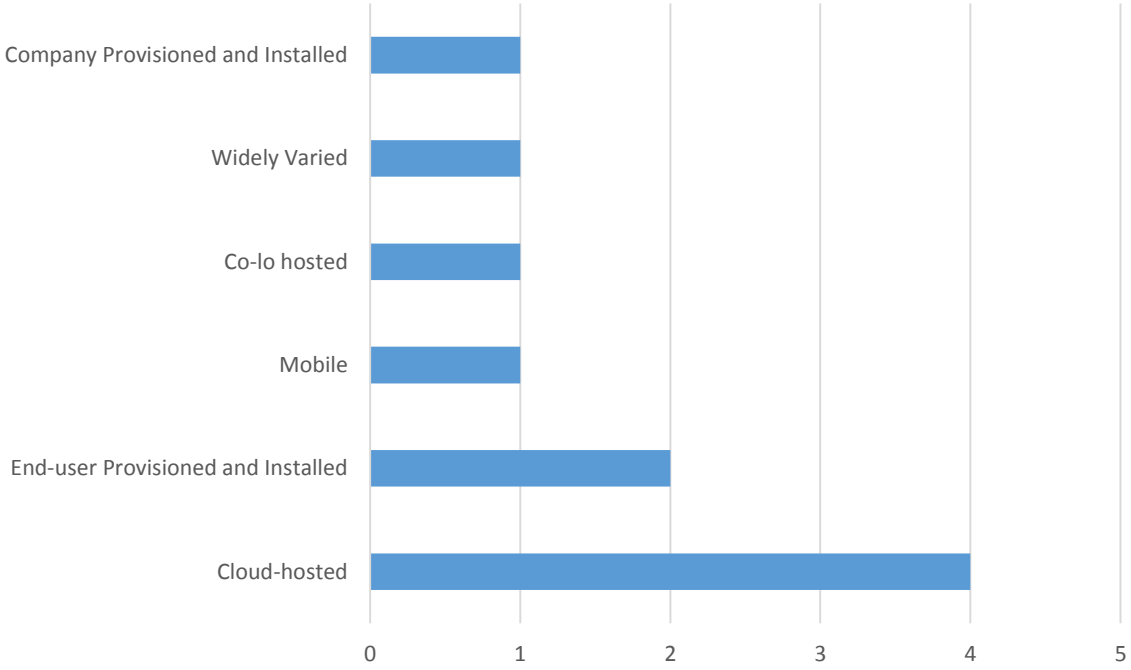


FIGURE 4.4 – Environment in which software is run in participating companies

TABLE 4.4 – Type of software produced and software environment

| Company | Software Produced | Software Environment |
|---|---|---|
| Company A | Product | Co-lo hosted |
| Company B | Bespoke Development | Cloud-hosted |
| Company C | Product | End-user Provisioned and Installed |
| Company D | Bespoke Development | Cloud-hosted |
| Company E | Product | Company Provisioned and Installed |
| Company F (Interview F1) | Product | Cloud-hosted |
| Company F (Interview F2) | Product | Mobile |
| Company G | Product | Cloud-hosted |
| Company H | Bespoke Development | Widely Varied |
| Company I | Product | End-user Provisioned and Installed |

### *4.3.4 Company Age*

The number of years each company has been in business is shown in Figure 4.5.

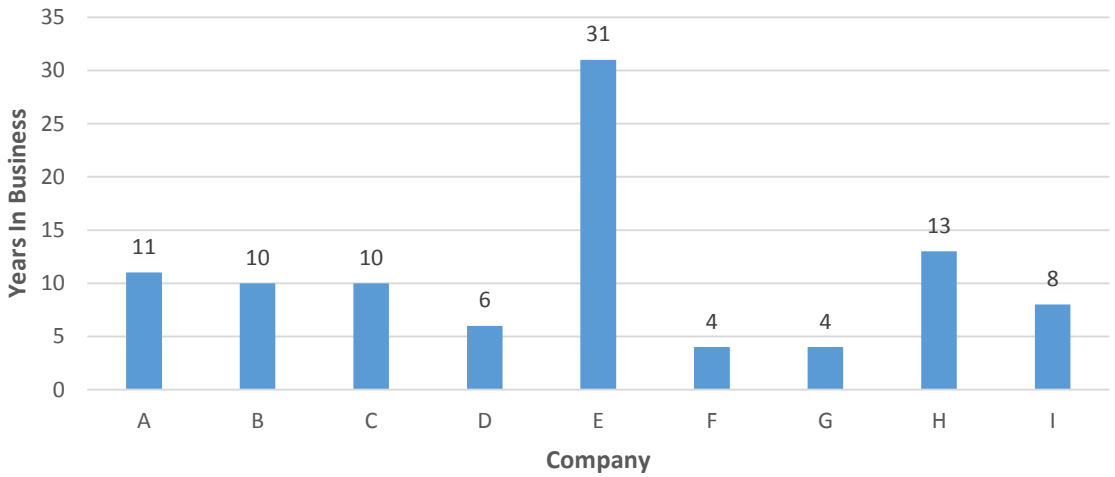


FIGURE 4.5 – Number of years each company has been in business

### *4.3.5 Interviewee Roles*

In two companies (Company C and Company G), two participants were interviewed.

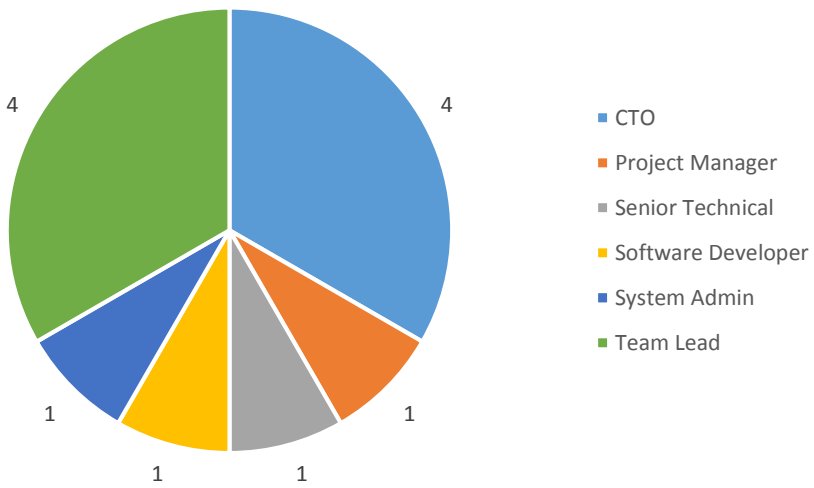


FIGURE 4.6 – Interviewees by role

The role of interviewees in their company is show in Figure 4.6. The role of Participant F1 has been listed as “Senior Technical” to preserve anonymity.

## 4.4 Adoption of Continuous Delivery Practices

As noted in section 4.2.1, analysis of participants’ answers to structured questions allowed their company’s adoption of CD practices to be rated against the model, providing an overall rating of adoption and directly addressing research question RQ1. (See section 3.7). This overall rating is referred to as “CD Rating”. In addition, a rating could be determined for adoption in each key process area of the CDMM by analysing questions asked specifically in those areas.

As noted previously, two interviews were conducted at Company F. In measuring overall adoption and addressing research question RQ1, only the interview with participant F1 was used. This interview concerned Company F’s main software product, in keeping with interviews at other companies.

### *4.4.1 Overall Adoption*

Figure 4.7 shows the overall CD rating of each participating company.

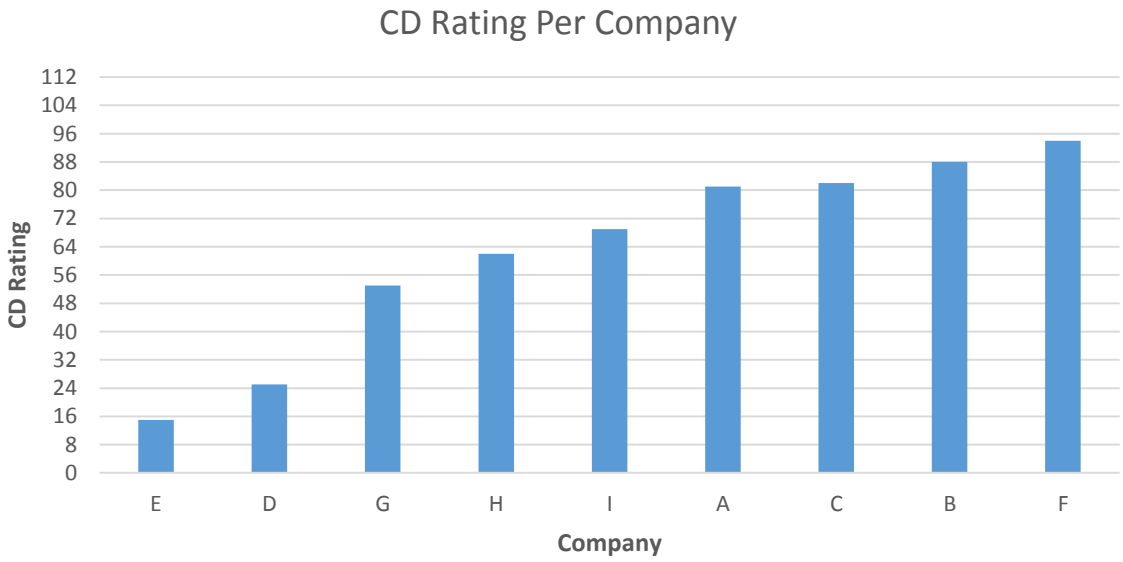


FIGURE 4.7 – Overall Continuous Delivery rating per company

The results show that adoption of Continuous Delivery practices varies greatly among the sampled companies. With the maximum achievable CD rating being 112, standard deviation was calculated at σ=27.7389, with a mean of 63.2222. See Table 4.5.

TABLE 4.5 – Companies rated by adoption of Continuous Delivery practices

| Company | CD Rating |
|---|---|
| **Company A** | 81 |
| **Company B** | 88 |
| **Company C** | 82 |
| **Company D** | 25 |
| **Company E** | 15 |
| **Company F** | 94 |
| **Company G** | 53 |
| **Company H** | 62 |
| **Company I** | 69 |
| **Mean** | **63.222** |
| **SD** | **27.739** |

*4.4.2 Key Process Areas*

Company's ratings in each key process area of the CDMM can be seen in Table 4.6.

TABLE 4.6 – Key process area ratings per company

| Company | Build Mgmt. and CI | Environments and Deployment | Release Mgmt. and Compliance | Testing | Data Mgmt. | Config. Mgmt. | Overall CD Rating |
|---|---|---|---|---|---|---|---|
| **A** | 20 | 9 | 18 | 12 | 13 | 9 | **81** |
| **B** | 22 | 14 | 20 | 15 | 11 | 6 | **88** |
| **C** | 20 | 12 | 17 | 14 | 12 | 7 | **82** |
| **D** | 2 | 5 | 13 | 3 | 0 | 2 | **25** |
| **E** | 1 | 0 | 5 | 3 | 2 | 4 | **15** |
| **F** | 23 | 16 | 19 | 14 | 11 | 11 | **94** |
| **G** | 7 | 9 | 15 | 8 | 10 | 4 | **53** |
| **H** | 7 | 6 | 16 | 10 | 12 | 11 | **62** |
| **I** | 19 | 11 | 11 | 9 | 10 | 9 | **69** |
| **Total KPA Rating** | **121** | **82** | **134** | **88** | **81** | **63** | |
| **Mean** | **13.444** | **9.111** | **14.889** | **9.778** | **9.000** | **7.000** | |
| **SD** | **9.015** | **4.910** | **4.676** | **4.522** | **4.664** | **3.240** | |

Results in each key process area were normalised to 100 to aid comparison when discussing standard deviation. The mean in Table 4.7 therefore represents the percentage adoption in each key process area. Charts in this section display the underlying values.

TABLE 4.7 – Key process area and overall CD rating per company, normalised to 100

| Company | Build Mgmt. and CI | Environments and Deployment | Release Mgmt. and Compliance | Testing | Data Mgmt. | Config. Mgmt. | Normalised CD Rating |
|---|---|---|---|---|---|---|---|
| **A** | 83.33 | 50.00 | 75.00 | 75.00 | 81.25 | 64.29 | **72.321** |
| **B** | 91.67 | 77.78 | 83.33 | 93.75 | 68.75 | 42.86 | **78.571** |
| **C** | 83.33 | 66.67 | 70.83 | 87.50 | 75.00 | 50.00 | **73.214** |
| **D** | 8.33 | 27.78 | 54.17 | 18.75 | 0.00 | 14.29 | **22.321** |
| **E** | 4.17 | 0.00 | 20.83 | 18.75 | 12.50 | 28.57 | **13.393** |
| **F** | 95.83 | 88.89 | 79.17 | 87.50 | 68.75 | 78.57 | **83.929** |
| **G** | 29.17 | 50.00 | 62.50 | 50.00 | 62.50 | 28.57 | **47.321** |
| **H** | 29.17 | 33.33 | 66.67 | 62.50 | 75.00 | 78.57 | **55.357** |
| **I** | 79.17 | 61.11 | 45.83 | 56.25 | 62.50 | 64.29 | **61.607** |
| **Mean** | **56.019** | **50.618** | **62.037** | **61.111** | **56.250** | **50.001** | |
| **SD** | **37.564** | **27.279** | **19.482** | **28.260** | **29.148** | **23.146** | |

*4.4.3 Build Management and Continuous Integration*

This key process area examined software build automation, build output, build metrics and the processes governing these. Mean adoption of practices in this area was calculated at 56.019%, but companies varied widely in their adoption of these practices, as indicated by the normalised standard deviation of σ=37.564%, seen in Table 4.7.

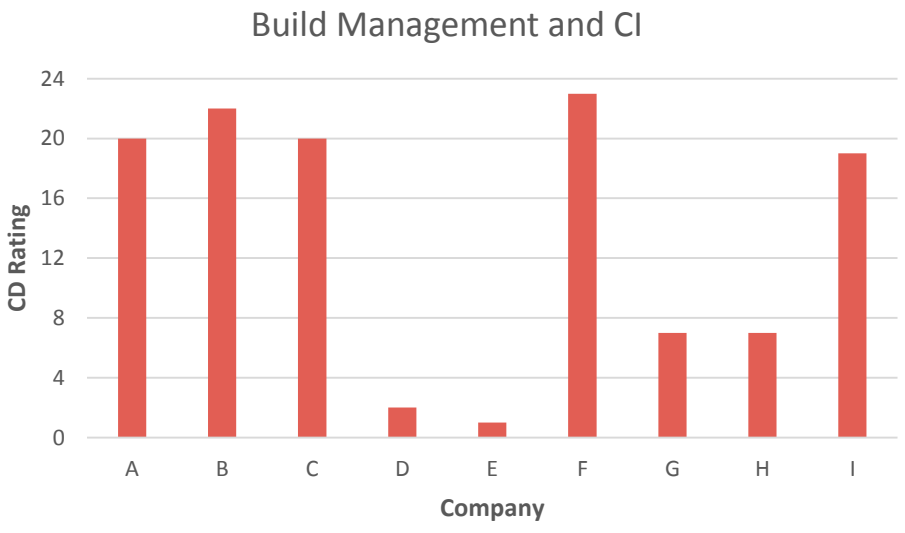

FIGURE 4.8 – Companies rated on the “Build Management and CI” KPA

A high rating in the “Build Management and Continuous Integration” key process area was best predictor of overall Continuous Delivery Rating in the sample, with a correlation of r=0.944 between the two sets of data.

### *4.4.4 Environments and Deployment*

Questions in this area targeted practices related to software deployment automation; provisioning of environments; release and rollback procedures; and orchestrating deployment of dependent services.

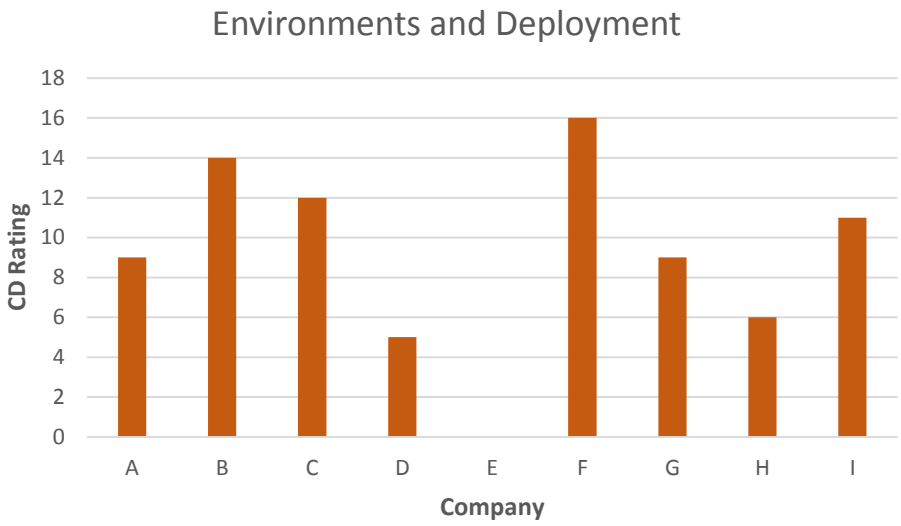


FIGURE 4.9 – Companies rated on the “Environments and Deployment” KPA

Company E is of particular note in this key process area; the company had not adopted any of the practices listed by the CDMM.

The “Environments and Deployment” key process area highlighted the differences approach when trying to employ Continuous Delivery for a mobile-based application, as opposed delivering a web platform or service that relies on server provisioning. This is discussed in more detail in Section 4.5.

### *4.4.5 Release Management and Compliance*

Investigating the “Release Management and Compliance” key process area addressed the frequency of software releases; change management processes; compliance with regulatory measures; and monitoring of cycle time in development.

This key process area showed the highest rate of adoption of practices, with a mean of 62.037% (See Table 4.7). In addition, the standard deviation for this key process area was the smallest of all areas, indicating a lower dispersion of ratings. However, the standard deviation does remain large, at σ=19.482% (See Table 4.7).

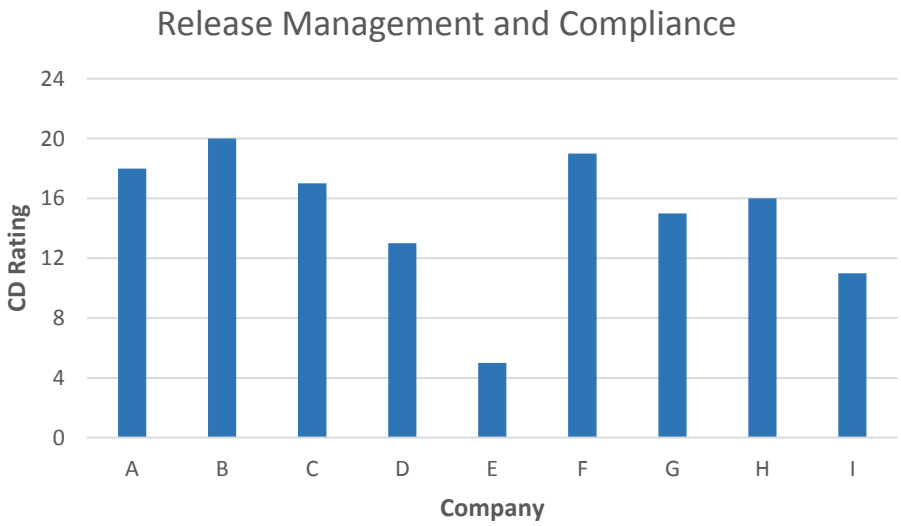


FIGURE 4.10 – Companies rated on the “Release Management and Compliance” KPA

### *4.4.6 Testing*

Mean adoption of practices in the “Testing” key process area was 61.111%, with a standard deviation of σ=28.260%. (See Table 4.7.)


FIGURE 4.11 – Companies rated on the “Testing” KPA

### *4.4.7 Data Management*

Mean adoption of practices in the “Data Management” key process area was 56.250%, with a standard deviation of σ=29.148%. (See Table 4.7.)


FIGURE 4.12 – Companies rated on the “Data Management” KPA

Quantitative and qualitative analysis and comparison revealed that some difficulty was expressed in this area in general, with four of the interviewed companies using manual processes in deployment of database schema changes, contrary to the recommendations of the CDMM. Company D had not adopted any practices related to Continuous Delivery in this key process area.

### *4.4.8 Configuration Management*

Participating companies rated lowest in this key process area, with mean adoption of practices calculated at 50.001%. Normalised standard deviation was σ=23.146%. See Table 4.7.

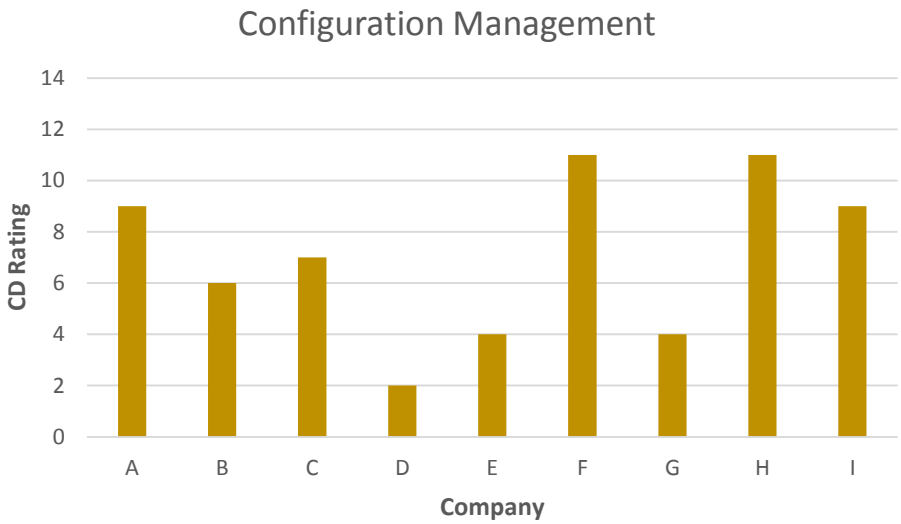


FIGURE 4.13 – Companies rated on the “Configuration Management” KPA

Although the correlation is relatively strong, ratings in the “Configuration Management” key process area were the worst predictor of overall Continuous Delivery rating in the sample, with a correlation of r=0.705 between the two ratings.

## 4.5 Factors Influencing Adoption of Continuous Delivery Practices

Hypotheses were formulated in section 3.7.2 based on existing literature, stating that there would be an association between level of adoption of CD practices and a number of potentially-influencing factors. These factors were: number of software employees; total number of employees; company age; and target software environment. Investigating these factors directly addressed research question RQ2. A number of observations were made when considering these factors, which are supported by both quantitative and qualitative analysis.

### *4.5.1 Number of Employees*

Participants were asked how many employees in total were employed in their company and how many of those employees were directly involved in software development (with the latter referred to in this study as "software employees").

In order to determine the effect of the number of software employees on each company's overall Continuous Delivery rating, data from each of these variables was appropriately categorised such that cross-tabulation could be performed, the results of which can be seen in Table 4.8.

TABLE 4.8 – Contingency table: categorised CD rating and number of software employees

| | | Number of Software Employees | | | | |
|---|---|---|---|---|---|---|
| | | **0 to 10** | **11 to 20** | **21 to 30** | **31+** | **Total** |
| **CD Rating** | **<40** | 2 | 0 | 0 | 0 | **2** |
| | **41-80** | 0 | 1 | 2 | 0 | **3** |
| | **80+** | 0 | 2 | 0 | 2 | **4** |
| **Total** | | **2** | **3** | **2** | **2** | **9** |

Cross-tabulation permitted analysis using Fisher's exact test, which returned a result of 9.332 and p-value of $p = 0.033$. This is valid at the significance level $\alpha = 0.05$; the result was therefore deemed to be statistically significant and the hypothesis that number of software employees is associated CD adoption is supported.

Furthermore, a strong positive correlation of $p=0.618$ was observed between these two variables. Hypothesising that an increasing number of employees would result in a higher

Continuous Delivery rating, this correlation would result in a one-tailed significance of p=0.038.

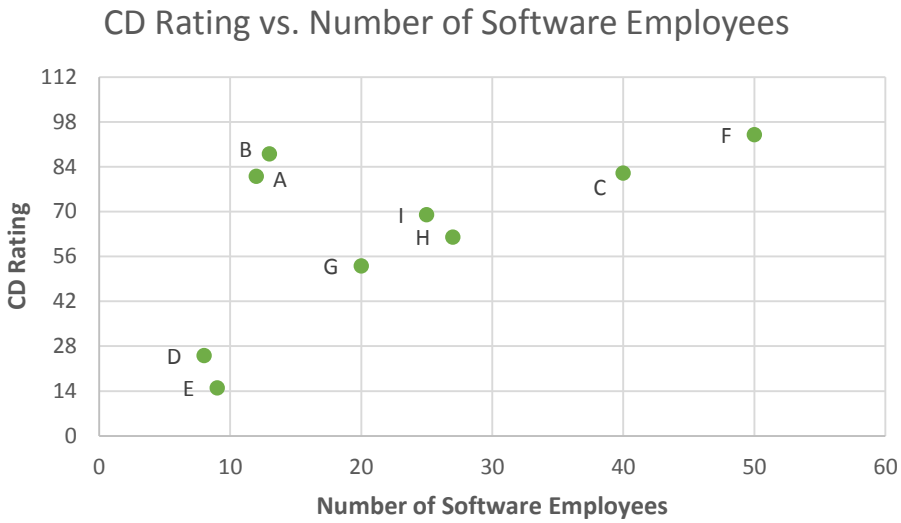


FIGURE 4.14 – Graph showing overall CD rating vs. number of software employees

This positive correlation and the significant results from Fisher's exact test indicate that companies with a greater number of software employees tended to have a higher Continuous Delivery rating.

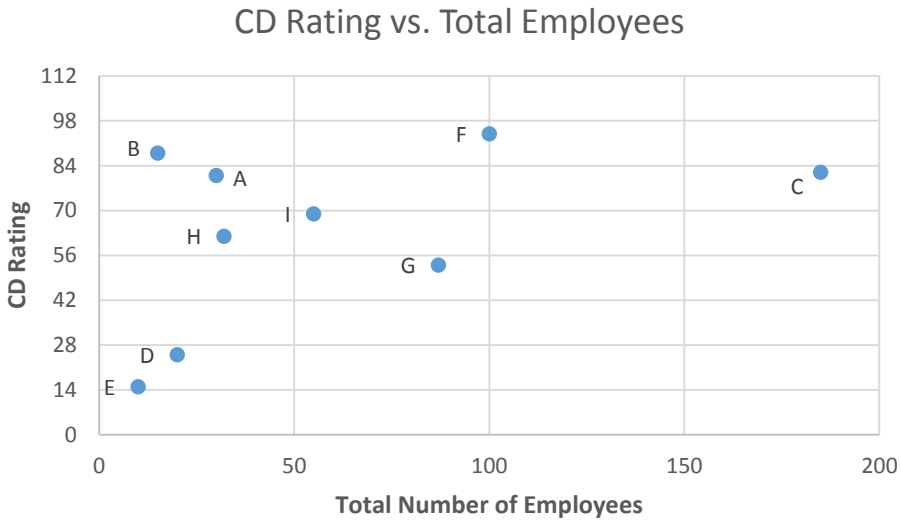


FIGURE 4.15 – Graph showing overall CD rating vs. total number of employees

A statistically significant association between the total number of employees and their Continuous Delivery rating was also found using Fisher's exact test, supporting the hypothesis made in section 3.7.2. A result of 8.757 was returned, with a p-value of p = 0.048. Again, this is valid at the significance level α = 0.05, showing significant association between the variables.

TABLE 4.9 – Contingency table: categorised CD rating and number of employees

| | | Number of Employees | | | | |
|---|---|---|---|---|---|---|
| | | **0-30** | **31-60** | **61-90** | **91+** | **Total** |
| **CD Rating** | **Under 40** | 2 | 0 | 0 | 0 | **2** |
| | **41 to 80** | 0 | 2 | 1 | 0 | **3** |
| | **80+** | 2 | 0 | 0 | 2 | **4** |
| **Total** | | **4** | **2** | **1** | **2** | **9** |

*4.5.2 Company Age and Legacy Systems*

It was hypothesised that there would be a significant association between company age and CD adoption, based on implications in existing literature. (See section 3.7.2).

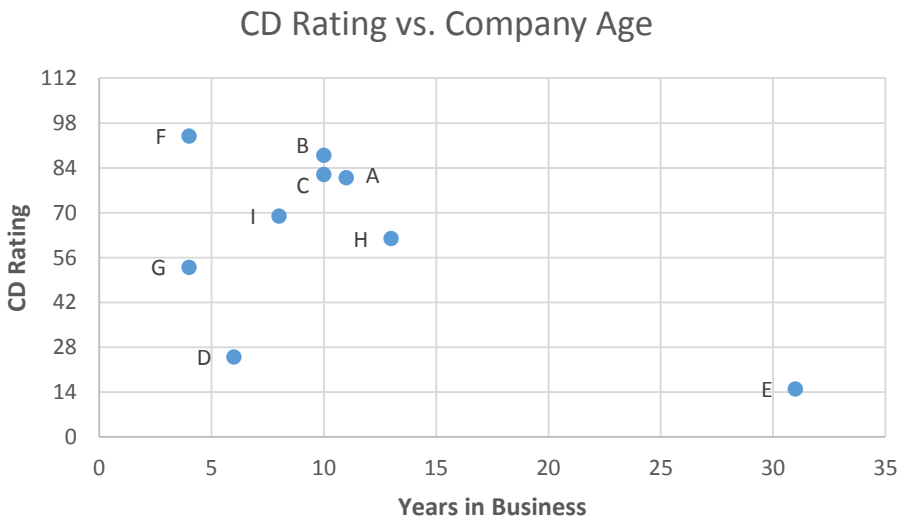


FIGURE 4.16 – Graph showing overall Continuous Delivery rating vs. company age.

The result produced by Fisher's exact test (6.795, p=0.352) when comparing age categories with CD rating categories was not significant (see data in Table 4.10) and a moderate

negative correlation of -0.539 was observed between the two variables. The fact that Company E is an outlier in this sample should also be noted; with Company E excluded, a weak positive correlation of 0.222 is observed.

However, Company A and Company C both stated that a project was currently underway to improve build and release processes, with Company A reporting vast improvements in the areas exported in the last 6 months in particular. The CD ratings for these companies may therefore be higher than could be expected for companies of this age in the wider population.

TABLE 4.10 – Contingency table: categorised CD rating and company age in years

| | | Years in Business | | | | |
|---|---|---|---|---|---|---|
| | | 0 to 4 | 5 to 9 | 10 to 14 | 15+ | Total |
| CD Rating | <40 | 0 | 1 | 0 | 1 | **2** |
| | 41-80 | 1 | 1 | 1 | 0 | **3** |
| | 80+ | 1 | 0 | 3 | 0 | **4** |
| **Total** | | **2** | **2** | **4** | **1** | **9** |

While considering company age as a CD-influencing factor, it is interesting to consider that comparative analysis revealed issues with legacy systems and CD. Company A, Company G and Company I all highlighted issues with legacy systems in implementing some of the CD key practices. Company G, at just four years old, is of particular note. Information for all companies can be seen in Table 4.11.

TABLE 4.11 – Number of years each participating company has been in business

| Company | Years in Business |
|---|---|
| **A** | 11 |
| **B** | 10 |
| **C** | 10 |
| **D** | 6 |
| **E** | 31 |
| **F** | 4 |
| **G** | 4 |
| **H** | 13 |
| **I** | 8 |

While older companies will, in general, have older software systems, it is recognised that this is not always the case. Company B provides bespoke software for customers and noted the advantages that this affords the company from a delivery perspective: "…when we learn of a new tool that might provide us some advantage, whether that's for build, static analysis or whatever, we tend to try it out, maybe for one project, to see what it's like. Then if it's worthwhile, we'd tend to recommend that across all the [future] projects." This practice, combined with participant reports of difficulty with legacy systems, suggests that adoption of these practices may be easier for young software projects.

Finally, Company F, the highest rated of all participating companies, employed Continuous Delivery practices very early in the company's formation and specifically built an internal tool for release automation. Although the company is not experiencing issues with their current system, Participant F1 does indicate that "if we built this again, we'd probably build on lightweight containers", again supporting the idea that implementation of these practices is easier early in the development process. The company is now four years old and routinely deploys software to the production environment "between 50 and 100 times a day".

### *4.5.3 Software Environment*

No significant association was found between company software environment (categorised as "hosted" or "provisioned") and the Continuous Delivery rating achieved. (Fisher's exact test result=0.857, p=1.000. See Table 4.12). The hypothesis that there is an association between this factor and CD adoption is therefore rejected. A null hypothesis (no association between software environment and CD adoption) is accepted.

TABLE 4.12 – Contingency table: categorised CD rating and software environment

| | | **Software Environment** | | **Total** |
|---|---|---|---|---|
| | | **Hosted** | **Provisioned** | |
| **CD Rating** | **<40** | 1 | 1 | **2** |
| | **41-80** | 2 | 1 | **3** |
| | **80+** | 3 | 1 | **4** |
| **Total** | | **6** | **3** | **9** |

However, the most commonly-occurring concept arising from constant comparison of interview transcripts was related to companies not adopting certain Continuous Delivery practices due to the nature of the software they were producing; with 17 coded segments

over 6 interviews contributing to this concept. Cited reasons for lack of adoption were related to both the nature of the software projects being undertaken and the processes supporting them.

For example, Company B produces bespoke software for clients and noted some differences in approaches taken when developing mobile software and developing web software. Company C produces a software product that is installed remotely on customer sites. Customers initiate software installs, meaning the company is not able to truly deliver continuously to the production environments. As Participant C1 noted of upgraded versions of the software, "customers usually have a sandbox where it's run for up to a few months before even considering moving it to production." Despite this, the company's development practices mean that it still rates highly for Continuous Delivery, as seen in Table 4.6. Company I is in a similar situation, with remote installs on customer sites preventing truly continuous delivery per the chosen CDMM. Company E installs hardware and software on customer sites, many of which are not internet-connected, with some software upgrades requiring a site visit. This prohibits many of the practices listed in the Continuous Delivery Maturity Model.

### *4.5.4 Developing Software for Mobile Devices*

Codes and memos emerging from comparative analysis had indicated difficulty in adopting CD on different platforms (see section 4.5.3). Early analysis indicated that web-based development was most conducive to CD adoption, and detailed accounts of difficulty with mobile development had been collected.

For example, Company B produces bespoke software for customers on various platforms, including mobile and web platforms. Participant B reported that CD practices were not adopted to the same extent for mobile projects due to platform difficulties, while web-based projects had the most mature CD practices associated with them. A similar view was expressed in Company A.

A theory was developed, hypothesising that adoption of CD practices in mobile software development is significantly lower than in web-based development. The null hypothesis states that there is no significant difference in CD adoption between mobile and web platform development.

TABLE 4.13 – Key process areas and overall CD rating for interviews at Company F

| Company | Build Mgmt. and CI | Environments and Deployment | Release Mgmt. and Compliance | Testing | Data Mgmt. | Config. Mgmt. | Rating |
|---|---|---|---|---|---|---|---|
| **F1** | 23 | 16 | 19 | 14 | 11 | 11 | **94** |
| **F2** | 23 | 8 | 13 | 14 | 12 | 12 | **82** |

In order to test this hypothesis, two interviews were conducted at Company F. The first interview (listed in Table 4.13 as F1) investigated practices related to the delivery of the company's main software platform, which consists of web applications and web-based APIs. The company also produces mobile apps for iOS and Android; practices related to these are investigated in the second interview (listed in Table 4.13 as F2). (As explained in section 4.2.1, the company's main software platform – Interview F1 – was used for inter-company comparison, in keeping with the investigations in other companies.)

No significant difference was found between the overall CD rating for mobile and web platform development and therefore the null hypothesis was accepted. However, differences were found within key process areas; the ratings for each key process area in mobile and web platform development can be seen in Figure 4.17, normalised to 100. Most notable are the differences in the "Environments and Deployment" and "Release Management and Compliance" key process areas.

A paired t-test was used to determine whether these differences were significant. The results seen the "Environments and Deployment" key process area were significant at $\alpha = 0.05$ level, with a result of $t = 2.602$, providing a two-tailed p-value of $p = 0.025$, with degrees of freedom $df = 11$. Differences in other key process areas were not significant, though it is noted that weightings given to answers at each maturity level could further accentuate these differences. Further details on this analysis including source data is available in Appendix 5 – Results of Statistical Analysis.

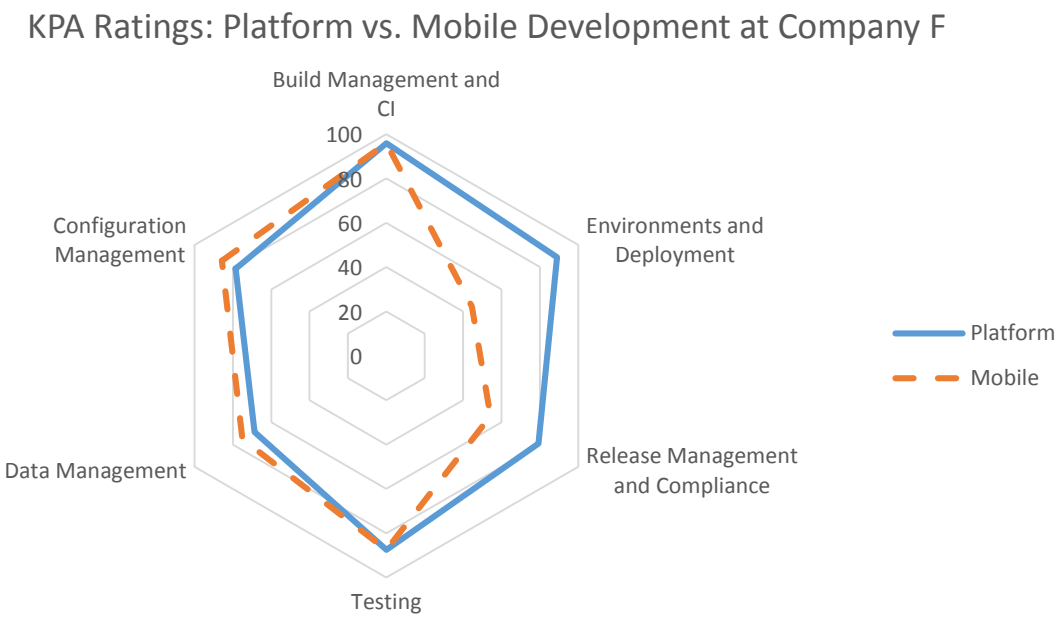


FIGURE 4.17 – KPA ratings for web platform and mobile development in Company F

Participant F2 provided some insight into the differences between web-focused software development and mobile software development, which accounts for these differences and provides insight into Continuous Delivery on a mobile platform. The participant remarks that a number of internal processes are constrained by the app store approval processes that take place before an update to a mobile app can be released on major app stores. App updates are reviewed and approved by major app store providers such as Apple and Google before they are made available to the public. The time taken for approval can vary; the interviewee reports that it can be as short as 30 minutes, but at the time of interview the company had been waiting for 11 days for an app update to be approved.

Deployment processes, release cadence, build processes and rollback procedures are all restricted by app store review policies (most notably, the participant states “a rollback is basically removing the app from sale until you’ve got an update to fix it”). These restrictions limit the CD rating that can be obtained by mobile software in the chosen CDMM.

Participant P2 summarised, “I think, in terms of Continuous Delivery [in mobile development] …the gate is guarded by the people who own the distribution channels. But if you were to imagine that those guys didn’t exist, what’s stopping you from doing it? Probably nothing.” Asked if Company F would hypothetically embrace continuous

deployment for mobile applications (in addition to their web platform), the interviewee responded, “If we didn’t have Apple standing in the way, or Google standing in the way, we would.”

In addition to constraints related to app review, Participant F2 noted platform-specific issues which further limit the adoption of key practices in the CDMM, such as the inability to heavily monitor application performance due to its effects on device battery life. These findings strongly indicate that the same CD practices not apply equally to web- and mobile-focused software development.

## 4.6 Summary

In order to measure CD adoption and address research question RQ1 (see section 3.3), interview questions were formulated based on a Continuous Delivery Maturity Model (see section 3.7.1). Analysis of participant responses showed that adoption of CD practices varies greatly among SMEs in Ireland.

To address RQ2, a number factors that were implied to influence CD adoption were identified in existing literature. (See section 3.7.2). Interview questions were formulated based on the implied factors and answers were analysed in conjunction with CD adoption data. Results showed that a given company’s number of employees and its number of software employees were significantly associated with CD adoption; a greater number of either predicts greater CD adoption.

Company age was hypothesised to be significantly associated with CD adoption. Although statistical analysis did not show a significant association, participants reported issues implementing CD practices in legacy systems. This was a common theme, which emerged from constant comparative analysis of transcripts.

It was hypothesised that software environment and CD adoption would be significantly associated. Statistical results did not show significance between these two variables. However, comparative analysis convincingly showed that participants struggled to adopt CD practices in certain environments. This led to an emerging theory that predicted a lower adoption of CD practices in mobile software development. Testing this theory in the field showed a statistically significant difference in the adoption of practices within the CD key process area related to pre-production environments and application deployment.

As noted in section 3.7, addressing research question RQ2 also involved exploring unknown factors related to CD adoption. The research did uncover additional influencing factors and emerging themes, which are discussed in chapter 5.

# 5 Discussion

## 5.1 Introduction

Coding and constant comparative analysis of interview transcripts using grounded theory method revealed a number of emerging themes and concepts, the most important of which are considered and discussed in this chapter. Factors that influence CD adoption were indicated by these concepts, directly addressing research question RQ2. (See section 3.7.3).

## 5.2 The Nature of Continuous Delivery

Consideration of which companies are specifically using the key practices to gain the benefits listed in existing literature (Chen 2015a; Neely and Stolt 2013; Humble and Farley 2010) suggests that only Company B, Company G and Company F have truly employed Continuous Delivery, despite high ratings achieved by other companies.

TABLE 5.1 – Table showing software release frequency in participating companies

| Company | Software Release Frequency |
|---|---|
| **A** | Daily |
| **B** | Fortnightly |
| **C** | Quarterly |
| **D** | Monthly |
| **E** | Quarterly |
| **F** | 40 – 100 times per day |
| **G** | Daily |
| **H** | Weekly |
| **I** | Monthly |

Company C, Company H and Company I scored well against the maturity model but are prevented from benefiting from Continuous Delivery's accelerated time to market, faster feedback and reliable releases (Chen 2015a), due to the nature of the software they develop. (See Section 4.5, page 55).

Company A has recently adopted many of the practices listed and the company reports that it is already seeing benefits, but developers are not focused on making software deliverable at any point in time; the goal of Continuous Delivery. For example, the company does not

have a formal or informal policy around fixing broken builds immediately, though Participant A states that this is a practice that will be adopted in future. Interestingly, Company F tackles this issue through "culture" – see Section 5.3.

Company G, despite scoring third-lowest against the maturity model, has developed a daily release cadence, allowing it to gain many of Continuous Delivery's benefits as listed above. The company also reports reliable releases despite a lack of adoption of some key practices; and suggests that frequent releases themselves may reduce risk: "We don't deploy hundreds of features at once. We do it as they're available – we don't try to batch them up, which can cause problems." The company plans to adopt some of the absent practices in order to reduce risk around software changes. Lack of adoption in the meantime, however, indicates that the company currently values Continuous Delivery's frequent releases and faster feedback over the unknown reduction in deployment risk.

While Company B's release frequency is lower, as seen in Table 5.1, the company has adopted key Continuous Delivery practices to ensure the reliability and performance of releases – factors which may be more important to this company since it engages in bespoke software development. However, the company also benefits from continuous deployment to internal environments, which provides a feedback loop for developers. Finally, the company does engage in true continuous delivery in collaboration with some customers, which allows it to benefit from continuous feedback.

Company F has adopted Continuous Delivery to the extent of Continuous Deployment – testing and automated deployment processes are mature enough to allow the company to deploy every software change to the production environment. This occurs between 40 and 100 times per day.

## 5.3 Continuous Delivery Culture

"Culture" emerged as an in-vivo code during transcript analysis, referring to the ethos, philosophy or attitude of the engineering team in Company F. A number of references were made to the "culture" or "philosophy" of the team.

Despite both rating highly against the maturity model, Company F was seen to benefit greatly from Continuous Delivery, while Company A did not. (See Section 5.2). One notable

difference between Company A and Company F was the “culture” within the engineering team of delivering software continuously.

For example, although neither company has an official policy on fixing broken builds. Company F notes, “[Fixing builds] relies on people being good people… Everyone wants things to move smoothly, so usually the person who breaks it fixes it, but it’s not uncommon for other people to fix things.” Company F also noted culture when describing the benefits of Continuous Delivery, with Participant F1 stating “culturally, I think it’s more enjoyable to work in that kind of environment.”

Further contributing to the notion of engineering culture having an effect on Continuous Delivery; Company B, Company C and Company F – those companies considered to be truly benefiting from Continuous Delivery (see section 5.2) – all scored well on key practices related to collaboration (regularly reviewing configuration management processes; collaboration to manage risks and reduce cycle time; regularly meeting to improve the build management and continuous integration process). It is noted, too, that these collaborative practices are listed at the highest level of process maturity within the CDMM. Alternative CDMMs also refer specifically to culture (Praqma A/S 2015; Rehn et al. 2012).

This may suggest that “cultural” practices are key to truly benefit from the delivery process; a high CD rating without these practices may not be enough.

## 5.4 The Influence of Continuous Delivery on Business Processes

A number of examples arose in higher-rated companies of business processes changing to better leverage Continuous Delivery. Use of Continuous Delivery seems to enable practitioners to better exploit Agile-style iterative processes, with early feedback allowing iterations at the product or business level.

Participant F1 stated, “I think it affects how your product and business can work too. You can operate around … the idea of learning and feedback loops, because you can make changes quickly.” It is interesting to note here that the early and continuous feedback provided by Continuous Delivery may be best utilised if it is considered by the business, resulting in small, incremental batches of requirements for software change. Company F considers new development work on a weekly basis. As reported by Participant F1, software development is “influenced by [product] roadmap, plus feedback from quality or

iteration and decided on a week-by-week basis… the engineering or building philosophy is to try to optimise for learning and feedback along the way." This an excellent example of leveraging Continuous Deployment to create Bosch's (2012) concept of "R&D as an Experiment System".

In the past six months, Company A has adopted continuous deployment to pre-production systems, which it uses to gain product feedback before release. As a result, the company has introduced smaller work batches per development cycle and has modified the development process to allow for further iteration based on this feedback loop. Participant A indicates that the company has seen improvements from this and predicts further process change, stating, "we've definitely seen benefits from it [change to processes], but we're not there yet." The introduction of smaller batches of work is in keeping with Company F's weekly consideration of new development work and can be seen to work alongside CD's smaller work batches (see section 2.3.1); an example of the Lean principle of "optimising the whole" (see section 2.4.1).

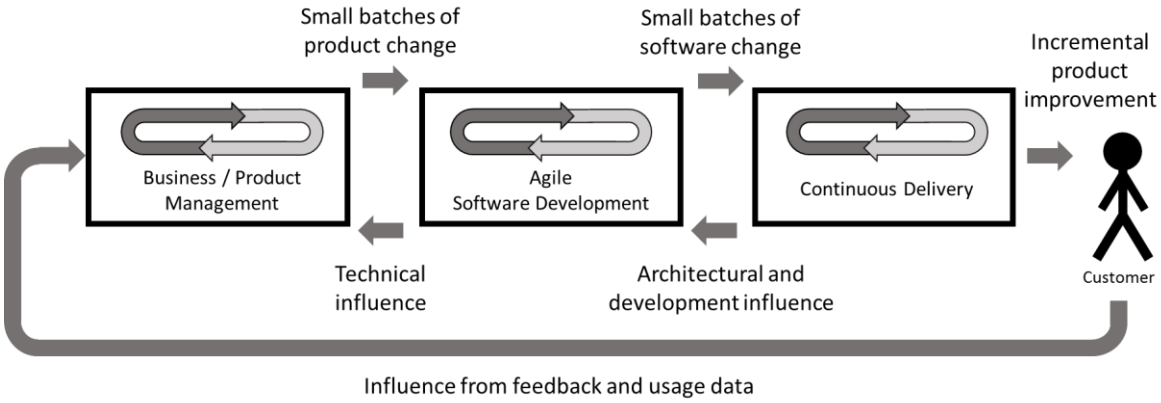


FIGURE 5.1 – The CD process in context. Based on research by Chen (2015b), Bosch (2012), Olsson et al. (2012), Humble and Farley (2010) and the findings of this research.

Combining the concept of smaller development work batches with research by Chen (2015b), Bosch (2012) and Olsson et al. (2012), Continuous Delivery can be placed within the cycle of product development and release. As seen in Figure 5.1, small batches of change flow from the business to the customer, while technical factors, architectural factors and customer feedback influence business and development processes.

Company B, which produces bespoke software, noted that many of its clients are accustomed to agreeing a set of fixed requirements at the beginning of a project, as opposed to a continuously-delivered, evolving project. Participant B stated "we'd agree a set of requirements up-front with the customer, because that's what they're comfortable with." In cases such as these, the company is still able to develop software continuously and incrementally, but the product does not benefit from constant feedback. Other clients of Company B agree to development using a time-based model, allowing for continuous delivery, feedback and interaction. This interesting objection to Continuous Delivery by customers is in accordance with reports in existing literature (Karvonen et al. 2015; Blank 2014).

## 5.5 Perception of Continuous Delivery

Participants were asked how important the practices discussed were to the company. All participants, regardless of their CD adoption rating, were unanimous in their agreement that the practices they had adopted were useful.

Those higher on the CD Rating tended to express a greater dependence on the practices. Participant C1 stated "we absolutely couldn't function without them" and Participant G echoed this sentiment: "[they're] something we can't live without". Participant A said the practices "increase confidence in releases and features and allow us to trail new features", citing "huge benefits." Both participants at Company F similarly advocated Continuous Delivery practices, with Participant F1 describing them as "crucially important to how our engineering team works" and Participant F2 answering, "I shudder to imagine a world where we'd have to do it all by hand again." Company B noted the benefits for a company of its size: "for a small company, it's pretty important that we can do these things once and then not have to do them again." Participant E, whose company rated lowest against the Continuous Delivery Maturity Model, also saw value in practices, including those that were not adopted by the company.

There were cases in which incorrect assumptions about particular key practices were made, however. Two companies expressed an aversion to frequent releases, with one company predicting that frequent releases would cause "a lot of trouble." Reduced risk through frequent, smaller releases is perhaps counterintuitive; since release is currently a risky activity, it may be natural to assume this risk will be increased as release frequency is

increased. However, smaller, incremental changes lead to reduced risk overall (Humble and Farley 2010; Claps et al. 2015).

Another company indicated that a change to the delivery process was undesirable, because the current process was “reliable”. Research suggests that the delivery process can become more than a reliable release mechanism, providing true business value (Olsson et al. 2012; Bosch 2012; Leppanen et al. 2015).

## 5.6 Continuous Delivery and Business Success

It has been proposed that Continuous Delivery can improve both IT performance and overall organisational performance (Puppet Labs 2015). It is interesting to consider this proposition in the context of this research, by using available data about participating companies; following Glaser’s principle that “all is data”.

Company B rated highly against the CDMM and was also found to have truly adopted Continuous Delivery (see Section 5.2). The company has won a major industry award for exceptional growth twice in the past two years and continues to grow in both revenue and employees.

Company F has attracted strong international investment and has grown to over one hundred employees, serving thousands of customers, in the four years since establishment. Participant F1 reports that the company has used Continuous Delivery since very early in its development. Given that the company has closely aligned various business processes to exploit Continuous Delivery (see Section 5.4), it is very probable that Continuous Delivery has played some role in enabling this growth.

While this does not establish causation, it lends weight to the arguments made by Chen (2015a); Humble and Farley (2010); and Puppet Labs (2015).

## 5.7 Summary

Rigorous qualitative data analysis revealed additional factors that influence Continuous Delivery adoption, addressing research question RQ2.

Findings show that culture or ethos within companies is important if the full benefits of CD are to be achieved identifying it as an influencing factor in CD adoption. Furthermore, changes to business processes can help companies to achieve CD's benefits, indicating business processes as a factor influencing CD adoption. Assumptions made about CD were found to be false in some cases, which led to a negative perception of some CD practices. Since the participants in this study were senior technical employees with influence over their organisation, perception of CD is an influencing factor in adoption.

Finally, although causation cannot be established, the fact that the two highest-rated companies in terms of CD adoption have both achieved exceptional business success is certainly noteworthy.

# 6 Conclusions and Future Work

## 6.1 Introduction

This research aimed to determine the extent of Continuous Delivery adoption in Irish SMEs, and sought to uncover factors influencing Continuous Delivery adoption. This chapter briefly considers how the research questions were answered, and evaluates the research findings in the context of existing literature. The chapter also recognises the limitations of this study; considers its contribution to the field of research; and proposes areas for future research.

## 6.2 Research Summary

### *6.2.1 Research Questions and Answers*

This research investigated Continuous Delivery in companies in Ireland of SME size. The research sought to answer two research questions:

> RQ1: To what extent have small to medium-sized companies in Ireland adopted practices that contribute to Continuous Delivery?

To address RQ1, a Continuous Delivery Maturity Model (CDMM) was produced which was heavily based on the original model by Humble and Farley (2010) and adapted to this research using a model created by Forrester Research (2013). Structured interview questions derived from this CDMM were used to gather quantitative data on each participating company, with interviews targeting high-level technical employees in companies within the target demographic. Analysis of this data showed that adoption of Continuous Delivery practices varies greatly among SMEs in Ireland (see Section 4.4.1).

> RQ2: What factors influence adoption of Continuous Delivery within small to-medium-sized companies in Ireland?

To address RQ2, influencing factors that were implied in existing literature were considered, and unknown factors were explored. A number of factors that influence Continuous Delivery adoption in Irish SMEs were identified.

- Number of Employees

The results presented in section 4.3.1 show a significant association between number of employees and adoption of Continuous Delivery practices, justifying the focus of this research on companies of a particular size and establishing total number of employees as a factor influencing adoption of CD practices. The research also showed a significant association between number of software employees and adoption of Continuous Delivery practices, identifying another influencing factor.

- Software Environment

As presented in section 4.5.3, the results did not show a significant association between the environment in which software is run and the company's Continuous Delivery rating. This echoes the recent findings of Puppet Labs (Puppet Labs 2015, p.18).

However, during rigorous analysis of qualitative data using grounded theory method, a theory emerged which suggested that Continuous Delivery may be difficult when developing software for mobile devices in particular. To explore this further, a second interview was conducted at a participating company, specifically investigating mobile development. Analysis showed that although the overall Continuous Delivery rating was comparable, there was a statistically significant difference in the adoption of practices within the key process area related to pre-production environments and application deployment.

Humble et al. (2015, p.168) and Puppet Labs (2015, p.18) propose that Continuous Delivery can be employed successfully in mobile development. The results of this research support this position, but indicate that aspects of Continuous Delivery are more difficult to achieve when developing software of this nature.

- Company Age

While statistical results did not show that company age is a significant predictor of adoption of Continuous Delivery practices, rigorous analysis of transcripts revealed that multiple participants encountered issues with the implementation of Continuous Delivery practices in legacy systems (see section 4.5.2).

- Software Development Culture

Those companies benefitting most from Continuous Delivery (see Section 5.2) reported a culture or ethos of cooperation around software delivery and scored highly on key practices

related to collaboration (see Section 5.3). Development culture affects the value gained from Continuous Delivery, indicating it as a factor influencing further adoption.

- Business Processes

Analysis of transcripts revealed an alignment of business processes with Continuous Delivery within some of the highest-rated companies. In the case of one company, changes were made to business processes at an early phase of Continuous Delivery adoption, to increase value to the business. (See Section 5.4). This shows that appropriate business processes can derive maximum value from Continuous Delivery; therefore business processes are indicated as a factor influencing adoption.

- Perception of Continuous Delivery

While the highest-rated companies reported very positive opinions of Continuous Delivery practices, there were a few instances of participants reacting negatively to particular practices, or holding a different opinion of a practice than is proposed by the literature. (See Section 5.5). Since interviews were conducted with senior technical staff, these negative perceptions or mistaken assumptions about particular Continuous Delivery practices will influence their adoption.

### *6.2.2 Emerging Themes*

Further themes emerged while conducting constant comparative analysis of interview transcripts.

- Nature of Continuous Delivery

It was noted that some companies scored highly when rated against the CDMM, but were prevented from gaining the true benefits of Continuous Delivery outlined in section 2.3. Companies were seen to be prevented from gaining full benefits due to the nature of the software they develop, or due to differences in software development culture. (See section 5.2). Companies that are unable to achieve the full benefits of Continuous Delivery are less likely to adopt further CD practices.

- Legacy Systems

As noted in section 6.2.1, company age was not an indicator of Continuous Delivery adoption, but in-depth analysis of transcripts revealed that companies had problems applying Continuous Delivery practices to legacy systems. The age of software systems is therefore proposed as a factor in Continuous Delivery adoption, as opposed to the age of the company itself.

This contradicts research conducted by Puppet Labs, which states "the type of system – … greenfield or legacy – is not significant. Continuous Delivery can be applied to any system, provided it is architected correctly" (Puppet Labs 2015, p.5). Recent research by Chen (2015b) and Bellomo et al. (2014) suggests a number of architectural requirements that must be met by software in order to use CD effectively. It is perhaps the case, then, that legacy systems encountered in this research were not architected in a way that allows for Continuous Delivery. However, given that Continuous Delivery itself is a modern concept, it would seem likely that legacy systems by their nature are less likely to have been architected with Continuous Delivery in mind.

## 6.3 Research Limitations

Collection of data via interviews has resulted in rich and accurate data that address both research questions. However, the sample size has been limited by time constraints on the research and difficulty in finding participants at the desired technical level, in willing companies. Additionally, convenience sampling was used, due to time and financial constraints.

The research used a Continuous Delivery Maturity Model to help address RQ1 (see section 3.7). The model was customised to fit this research, making it appropriate for software-producing SMEs using any software development methodology. Although this CDMM provided an excellent tool for rating adoption of key practices related to Continuous Delivery, the capabilities of this CDMM have not been fully established. This presents an opportunity for future research, as discussed in section 6.4.

Grounded theory method was used to add rigour to the analysis of interview transcripts in addressing RQ2; the analysis techniques of coding, constant comparison and memo-writing enabled theory to emerge from the data, and allowed testing of the emerging theory within the time constraints of the research (see section 4.5.3). However, research time constraints did limit the extent to which true Grounded Theory research could be employed. As a result,

theoretical saturation may not have been reached and coding was completed by a single researcher. This, again, represents an opportunity for future research, as discussed further in section 6.4.

## 6.4 Opportunities for Future Research

As presented in section 3.7, a number of Continuous Delivery Maturity Models were evaluated when selecting the most appropriate model for this research. The research used a tailored CDMM that was appropriate for SMEs using any software development methodology. The benefits of this CDMM are yet to be explored. Future research using the same approach may attempt to determine which of the key practices are most beneficial to business and adjust the weightings of key practices accordingly; perhaps gaining further insights. (See weightings used in Appendix 6 – Interview Answer Weightings).

The research has identified that software development culture is important to gain benefits from Continuous Delivery. Alternative CDMMs do include key process areas related to culture and there is precedent in Information systems for Maturity Models to be consolidated (Poeppelbuss et al. 2011); this may be considered as an avenue for further research and would be of academic and practical interest. Development culture may also be investigated to determine which specific cultural factors are most important in enabling Continuous Delivery's benefits to be realised. Humble et al. (2015, p.168) and Puppet Labs (2015, p.22) identify Westrum's (2014) work on information flow as culturally important for Continuous Delivery, but there is an opportunity for further research. Identification of these cultural factors would be of particular interest to Continuous Delivery practitioners.

Given that this research reveals difficulties in implementing Continuous Delivery practices when developing software for mobile devices, this area represents an excellent opportunity for further research. Certain key practices in the CDMM were found to be prohibitive on mobile devices, while others were impossible due to app store review policies (see section 4.5.3). Future research could investigate which practices are applicable to mobile software development and may identify further practices specific to the mobile platform, which would be valued by practitioners. Additionally, difficulties identified with Continuous Delivery in the area of mobile development may indicate that other areas of software development have similar constraining factors, representing further research opportunities.

A theory has emerged during this research that suggests that implementing Continuous Delivery practices in legacy systems is difficult. This theory requires further exploration, presenting a research opportunity. Organisations using legacy systems have to make trade-off decisions related to ongoing use or redevelopment of legacy systems (Bisbal et al. 1999). Research into the difficulty involved in implementing Continuous Delivery in these systems could provide another factor to inform this decision.

Continuous Delivery practices ultimately have to operate in the context of the overall organisation. The research found companies using business processes that particularly suited Continuous Delivery. With this in mind, this research echoes Chen's (2015a) call for research into business practices that are most effective in enabling Continuous Delivery adoption and maximising its benefits. Humble et al. (2015) and Puppet Labs (2015, p.15) both suggest lean management techniques in this area, but further academic work is required.

Finally, this research identified that the number of employees in a company was associated with a greater adoption of Continuous Delivery practices. Conversely, Ries (2011, p.191) observed that Continuous Deployment (of which Continuous Delivery is an integral part) was gaining popularity in startups. Future research may consider how Continuous Delivery is achieved at the smaller scale, with a limited amount of employees and technical staff. This would be of interest to all Continuous Delivery practitioners, including the startup community.

## 6.5 Contribution to the Field of Research

In addition to proving a method and Continuous Delivery Maturity Model for future studies to undertake studies of Continuous Delivery (see Appendix 4 – Maturity Model Used in this Research), the research contributes to the existing literature by answering the research questions (see section 6.2.1) and presenting emerging themes which are of interest to the field of research (see section 6.2.2).

It is the researcher's belief that this study represents the first attempt to investigate widespread Continuous Delivery adoption in Ireland.

## 6.6 Conclusion

This research establishes that there is a broad spectrum of adoption of Continuous Delivery among companies in Ireland of SME size.

The research presents statistical evidence that the total number of employees and the number of software employees in these companies are factors that are significantly associated with the adoption of Continuous Delivery practices.

Significant differences were found between the adoption of practices when developing web-based software and mobile software, showing that software platform is also a factor in the viability of Continuous Delivery adoption.

Finally, the research identified culture, business processes and perception of Continuous Delivery as factors influencing adoption of the practice.

Based on these findings and existing literature, companies using or adopting Continuous Delivery should consider not only the technical aspects associated with the practice on their particular software platform, but also organisational constraints in the form of culture; business process; and opinion that may be required to fully realise the benefits of Continuous Delivery adoption.

## References

Akerele, O., Ramachandran, M. & Dixon, M., 2013. System dynamics modeling of agile continuous delivery process. In *Proceedings - AGILE 2013*. Nashville, TN: IEEE Computer Society, pp. 60–63.

Allen, A.O., 1990. *Probability, Statistics, and Queueing Theory: With Computer Science Applications*, Gulf Professional Publishing.

Babbie, E., 2012. *The Practice of Social Research* 13th ed., Wadsworth: Cengage Learning.

Beck, K., 2000. *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional.

Beck, K., 1998. Extreme Programming: A Humanistic Discipline of Software Development. In E. Astesiano, ed. *Fundamental Approaches to Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg. Available at: http://link.springer.com/10.1007/BFb0053578.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D., 2001. Principles behind the Agile Manifesto. *www.agilemanifesto.org*. Available at: http://agilemanifesto.org/principles.html [Accessed December 11, 2014].

Becker, J. & Niehaves, B., 2007. Epistemological perspectives on IS research: a framework for analysing and systematizing epistemological assumptions. *Information Systems Journal*, 17(2), pp.197–214. Available at: http://doi.wiley.com/10.1111/j.1365-2575.2007.00234.x [Accessed February 9, 2015].

Bekk Consulting AS, 2013. A Maturity Model for Continuous Delivery. *http://www.bekk.no*. Available at: http://bekkopen.github.io/maturity-model/ [Accessed March 15, 2015].

Bellomo, S., Ernst, N., Nord, R. & Kazman, R., 2014. Toward Design Decisions to Enable Deployability: Empirical Study of Three Projects Reaching for the Continuous Delivery Holy Grail. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, pp. 702–707. Available at:

http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6903628 [Accessed January 15, 2015].

Bisbal, J., Lawless, D. & Grimson, J., 1999. Legacy information systems: issues and directions. *IEEE Software*, 16(5), pp.103–111. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=795108 [Accessed August 16, 2015].

Blank, S., 2014. Why Continuous Deployment May Mean Continuous Customer Disappointment. *Harvard Business Review*. Available at: https://hbr.org/2014/01/why-continuous-deployment-may-mean-continuous-customer-disappointment [Accessed August 9, 2015].

Booch, G., 1991. *Object Oriented Design: With Applications*, Benjamin/Cummings Pub.

Booch, G., 1994. *Object-oriented Analysis and Design with Applications*, Benjamin/Cummings Publishing Company.

Bosch, J., 2012. Building Products as Innovation Experiment Systems. In M. Cusumano, B. Iyer, & N. Venkatraman, eds. *Software Business, ICSOB 2012*. Lecture Notes in Business Information Processing. Berlin: Speinger-Verlag Berlin, pp. 27–39.

Bryman, A., 2006. Integrating quantitative and qualitative research: how is it done? *Qualitative Research*, 6(1), pp.97–113. Available at: http://qrj.sagepub.com/content/6/1/97.abstract [Accessed July 11, 2014].

Van der Burg, S. & Dolstra, E., 2010. Automated Deployment of a Heterogeneous Service-Oriented System. In *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, pp. 183–190. Available at: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5598096 [Accessed February 7, 2015].

Central Statistics Office, 2014. *Business in Ireland*, Dublin: Stationery Office.

Charmaz, K., 2014. *Constructing Grounded Theory*, SAGE Publications.

Charmaz, K., 2006. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*, London: SAGE Publications.

Chen, L., 2015a. Continuous Delivery: Huge Benefits, But Challenges Too. *IEEE Software*, 32(2), pp.50–54. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7006384 [Accessed January 18, 2015].

Chen, L., 2015b. Towards Architecting for Continuous Delivery. *2015 12th Working IEEE/IFIP Conference on Software Architecture*, pp.131–134. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7158514.

Claps, G.G., Berntsson Svensson, R. & Aurum, A., 2015. On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software Technology*, 57, pp.21–31. Available at: http://www.sciencedirect.com/science/article/pii/S0950584914001694 [Accessed January 9, 2015].

Cockburn, A., 2006. *Agile Software Development: The Cooperative Game*, Pearson Education.

Cois, C.A., Yankel, J. & Connell, A., 2014. Modern DevOps: Optimizing software development through effective system interactions. In *2014 IEEE International Professional Communication Conference (IPCC)*. IEEE, pp. 1–7. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7020388 [Accessed February 6, 2015].

Corbetta, P., 2003. *Social Research: Theory, Methods and Techniques*, SAGE Publications.

Coupaye, T. & Estublier, J., 2000. Foundations of enterprise software deployment. *Proceedings of the Fourth European Conference on Software Maintenance and Reengineering*. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=827313 [Accessed December 10, 2014].

Dobson, P.J., 2002. Critical realism and information systems research: why bother with philosophy? *Information Research: an international electronic journal*, (2), p.124. Available at: http://elib.tcd.ie/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsdoj&AN=3d63ebcb72209196f4e22d9b55c2037b&site=eds-live.

Dyck, A., Penners, R. & Lichter, H., 2015. Towards Definitions for Release Engineering and DevOps. *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, pp.3–3. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7169442.

European Commission, 2005. *The New SME Definition, User Guide and Model Declaration*, Available at: http://bookshop.europa.eu/en/the-new-sme-definition-pbNB6004773/.

Fagerholm, F., Guinea, A.S., Mäenpää, H. & Münch, J., 2014. Building blocks for continuous experimentation. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering - RCoSE 2014*. New York, New York, USA: ACM Press, pp. 26–35. Available at: http://dl.acm.org/citation.cfm?id=2593812.2593816 [Accessed January 25, 2015].

Feitelson, D.G., Frachtenberg, E. & Beck, K.L., 2013. Development and Deployment at Facebook. *IEEE Internet Computing*, 17(4), pp.8–17. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6449236 [Accessed January 11, 2015].

Ferreira, N.N.V. & Langerman, J.J., 2014. Proving That the Release Management Process Can Enhance Throughput in Software Development Projects. *2014 9th International Conference on Computer Science & Education*, (Iccse), pp.419–424. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6926496.

Fitzgerald, B. & Stol, K.-J., 2014. Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering - RCoSE 2014*. New York, New York, USA: ACM Press, pp. 1–9. Available at: http://dl.acm.org/citation.cfm?id=2593812.2593813 [Accessed January 25, 2015].

Forrester Research, 2013. *Continuous Delivery : A Maturity Assessment Model*, Cambridge, MA. Available at: http://info.thoughtworks.com/Continuous-Delivery-Maturity-Model.html.

Forrester Research, 2014. *The New Software Imperative : Fast Delivery With Quality Eight DevOps Practices Are The Key*,

Fowler, M., 2013. Continuous Delivery. *martinfowler.com*. Available at: http://martinfowler.com/bliki/ContinuousDelivery.html [Accessed January 11, 2015].

Fowler, M., 2009. FlaccidScrum. *http://www.martinfowler.com/*. Available at: http://www.martinfowler.com/bliki/FlaccidScrum.html [Accessed February 1, 2015].

Fowler, M., 2014. Microservices. *martinfowler.com*. Available at: http://martinfowler.com/articles/microservices.html [Accessed November 19, 2014].

Gfader, P., 2012. Use Scrum and Continuous Delivery to build the right thing. *Scrum.org*, (August).

Glaser, B.G. & Strauss, A.L., 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine de Gruyter.

Hammond, J.S., Rymer, J.R., Gilpin, M., Knoll, A. & Rose, S., 2011. *Five Ways To Streamline Release Management*,

Haugset, B. & Hanssen, G.K., 2008. Automated Acceptance Testing: A Literature Review and an Industrial Case Study. *Agile 2008 Conference*, pp.27–38.

Heath, H. & Cowley, S., 2004. Developing a grounded theory approach: a comparison of Glaser and Strauss. *International Journal of Nursing Studies*, 41(2), pp.141–150. Available at: http://www.sciencedirect.com/science/article/pii/S0020748903001135 [Accessed July 9, 2014].

Hossain, E., Babar, M.A. & Paik, H., 2009. Using Scrum in Global Software Development: A Systematic Literature Review. In *2009 Fourth IEEE International Conference on Global Software Engineering*. IEEE, pp. 175–184. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5196931 [Accessed August 4, 2014].

Humble, J., 2014. The Case for Continuous Delivery. *ThoughtWorks.com*. Available at: http://www.thoughtworks.com/insights/blog/case-continuous-delivery [Accessed January 25, 2015].

Humble, J. & Farley, D., 2010. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Boston: Addison-Wesley Professional.

Humble, J., O'Reilly, B. & Molesky, J., 2015. *Lean Enterprise: How High Performance Organizations Innovate at Scale*, Sebastopol, CA: O'Reilly Media.

Humble, J., Read, C. & North, D., 2006. The deployment production line. In *AGILE Conference, 2006*. pp. 113–118. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1667569&isnumber=34909 [Accessed January 18, 2015].

Humble, J. & Russell, R., 2009. *The Agile Maturity Model Applied to Building and Releasing Software*, Available at: http://info.thoughtworks.com/agile-maturity-model-applied-building-and-releasing-software.html.

Johnson, P. & Clark, M., 2006. Mapping the terrain: an overview of business and management research methodologies. In *Business and Management Research Methodologies*. London: Sage.

Karvonen, T., Lwakatare, L., Sauvola, T., Bosch, J., Olsson, H., Kuvaja, P. & Oivo, M., 2015. Hitting the Target: Practices for Moving Toward Innovation Experiment Systems. In J. M. Fernandes, R. J. Machado, & K. Wnuk, eds. *Software Business SE - 10*. Lecture Notes in Business Information Processing. Springer International Publishing, pp. 117–131. Available at: http://dx.doi.org/10.1007/978-3-319-19593-3_10.

Kelemen, M.L. & Rumens, N., 2008. *An Introduction to Critical Management Research*, SAGE Publications.

Khazanchi, D. & Munkvold, B.E., 2003. On the rhetoric and relevance of IS research paradigms: a conceptual framework and some propositions. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*. IEEE, p. 10 pp. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1174717 [Accessed February 8, 2015].

Krusche, S., Alperowitz, L., Bruegge, B. & Wagner, M.O., 2014. Rugby: an agile process model based on continuous delivery. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering - RCoSE 2014*. New York, New York, USA: ACM Press, pp. 42–50. Available at: http://dl.acm.org/citation.cfm?id=2593812.2593818 [Accessed January 21, 2015].

Leppanen, M., Kilamo, T. & Mikkonen, T., 2015. Towards Post-Agile Development Practices through Productized Development Infrastructure. In *2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering*. IEEE, pp. 34–40. Available at: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=7167171 [Accessed August 17, 2015].

Memon, A.M. & Xie, Q., 2004. Empirical evaluation of the fault-detection effectiveness of smoke regression test cases for GUI-based software. *IEEE International Conference on Software Maintenance, ICSM*, pp.8–17.

Mettler, T., Rohner, P. & Winter, R., 2010. Towards a classification of maturity models in information systems. In *Management of the interconnected world*. Springer, pp. 333–340.

Minick, E., 2014. Continuous Delivery Maturity Model - IBM UrbanCode. *IBM developerWorks*. Available at: https://developer.ibm.com/urbancode/docs/continuous-delivery-maturity-model/ [Accessed March 1, 2015].

Neely, S. & Stolt, S., 2013. Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy). In *2013 Agile Conference*. IEEE, pp. 121–128. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6612887 [Accessed January 26, 2015].

Olsson, H.H., Alahyari, H. & Bosch, J., 2012. Climbing the “Stairway to Heaven” -- A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, pp. 392–399. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6328180 [Accessed December 16, 2014].

Paulk, M.C., Curtis, B., Chrissis, M.B. & Weber, C. V, 1993. Capability Maturity Model for Software, Version 1.1. *Software, IEEE*, 10(4), pp.18–27.

Peirce, C.S. & Houser, N., 1998. *The Essential Peirce: Selected Philosophical Writings, Volume 2*, Indiana University Press.

Poeppelbuss, J., Niehaves, B., Simons, A. & Becker, J., 2011. Maturity Models in Information Systems Research: Literature Search and Analysis. *Communications of the Association for Information Systems*, 29(1), pp.506–532.

Poppendieck, M. & Poppendieck, T., 2009. *Leading Lean Software Development: Results Are not the Point*, Boston: Pearson Education.

Poppendieck, M. & Poppendieck, T., 2003. *Lean Software Development: An Agile Toolkit*, Upper Saddle River, NJ: Addison-Wesley Professional.

Praqma A/S, 2015. Continuous Delivery Maturity Model. Available at: http://www.praqma.com/sites/default/files/img/cdmaturity_paper.pdf [Accessed March 1, 2015].

Puppet Labs, 2013. *2013 State of DevOps Report*,

Puppet Labs, 2015. *2015 State of DevOps Report*, Portland.

Rehn, A., Palmborg, T. & Boström, P., 2012. *The Continuous Delivery Maturity Model*, Stockholm. Available at: http://www.diabol.se/whitepapers/The Continuous Delivery Maturity Model.pdf [Accessed March 1, 2015].

Reichertz, J., 2009. Abduction: The Logic of Discovery of Grounded Theory. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 11(1). Available at: http://www.qualitative-research.net/index.php/fqs/article/view/1412/2902 [Accessed June 21, 2015].

Ries, E., 2011. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses* First., Crown Business.

Saunders, M., Lewis, P. & Thornhill, A., 2012. *Research Methods for Business Students* 6th ed., Harlow: Pearson Education Limited.

Saunders, M., Lewis, P. & Thornhill, A., 2009. *Research Methods for Business Students*, Harlow: Pearson Education Limited.

Schwaber, K. & Sutherland, J., 2013. The scrum guide. *Scrum.org*, 1, p.16. Available at: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf.

Strauss, A. & Corbin, J., 2008. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, London: Sage.

Suddaby, R., 2006. What grounded theory is not. *Academy of Management Journal*, 49(4), pp.633–642. Available at: http://search.ebscohost.com.library.myebs.de/login.aspx?direct=true&db=bth&AN=22083020&site=ehost-live.

Tashakkori, A. & Teddlie, C., 1998. *Mixed methodology: combining qualitative and quantitative approaches*, London: Sage.

Thönes, J., 2015. Microservices. *Software, IEEE*, 32(1), pp.113–116.

Velasquez, N.F., Kim, G., Kersten, N. & Humble, J., 2014. *2014 State of DevOps Report*,

Westrum, R., 2014. The study of information flow: A personal journey. *Safety Science*, 67, pp.58–63. Available at: http://www.sciencedirect.com/science/article/pii/S0925753514000174 [Accessed August 15, 2015].

Wong, W.E., Horgan, J.R., London, S. & Agrawal, H., 1997. A study of effective regression testing in practice. *Proceedings The Eighth International Symposium on Software Reliability Engineering*, pp.264–274.

Xebia Labs, 2015. *Introducing Continuous Delivery in the Enterprise*, Boston. Available at: https://xebialabs.com/assets/files/whitepapers/introducing-continuous-delivery-in-the-enterprise.pdf [Accessed March 20, 2015].

# Appendices

## Appendix 1 – Ethics Application and Supporting Documentation

**School of Computer Science and Statistics**
**Research Ethical Application Form**

**Part A**

Project Title: Continuous Delivery: A study of Software Release Practice

Name of Lead Researcher (student in case of project work): Grahame Todd

Name of Supervisor: Paula Roberts

TCD E-mail: toddgr@tcd.ie Contact Tel No.: 0879479690

Course Name and Code (if applicable): MSc Management of Information Systems

Estimated start date of survey/research: March/April 2015

I confirm that I will (where relevant):

- Familiarize myself with the Data Protection Act and the College Good Research Practice guidelines http://www.tcd.ie/info_compliance/dp/legislation.php;
- Tell participants that any recordings, e.g. audio/video/photographs, will not be identifiable unless prior written permission has been given. I will obtain permission for specific reuse (in papers, talks, etc.)
- Provide participants with an information sheet (or web-page for web-based experiments) that describes the main procedures (a copy of the information sheet must be included with this application)
- Obtain informed consent for participation (a copy of the informed consent form must be included with this application)
- Should the research be observational, ask participants for their consent to be observed
- Tell participants that their participation is voluntary
- Tell participants that they may withdraw at any time and for any reason without penalty
- Give participants the option of omitting questions they do not wish to answer if a questionnaire is used
- Tell participants that their data will be treated with full confidentiality and that, if published, it will not be identified as theirs
- On request, debrief participants at the end of their participation (i.e. give them a brief explanation of the study)
- Verify that participants are 18 years or older and competent to supply consent.
- If the study involves participants viewing video displays then I will verify that they understand that if they or anyone in their family has a history of epilepsy then the participant is proceeding at their own risk
- Declare any potential conflict of interest to participants.
- Inform participants that in the extremely unlikely event that illicit activity is reported to me during the study I will be obliged to report it to appropriate authorities.
- Act in accordance with the information provided (i.e. if I tell participants I will not do something, then I will not do it).

Signed: Grahame Todd Date: 05/03/2015
Lead Researcher/student in case of project work

**Part B**

| *Please answer the following questions.* | | *Yes/No* |
|---|---|---|
| Has this research application or any application of a similar nature connected to this research project been refused ethical approval by another review committee of the College (or at the institutions of any collaborators)? | | No |
| Will your project involve photographing participants or electronic audio or video recordings? | | Yes |
| Will your project deliberately involve misleading participants in any way? | | No |
| Is there a risk of participants experiencing either physical or psychological distress or discomfort? If yes, give details on a separate sheet and state what you will tell them to do if they should experience any such problems (e.g. who they can contact for help). | | No |
| Does your study involve any of the following? | Children (under 18 years of age) | No |
| | People with intellectual or communication difficulties | No |
| | Patients | No |

SCSS Research Ethics Application Form September 2011

## Information Sheet for Participants

**BACKGROUND OF RESEARCH:**

The research aims to investigate the software release and development practices of technology businesses in Ireland and compare them to a model of "Continuous Delivery" – a practice that aims to keep software in a state in which it can be released at any time.

The concept of Continuous Delivery first appeared in 2010 in a book by Jez Humble and David Farley, called "Continuous Delivery: Reliable Software Releases Through Build, Test and Deployment Automation".

Continuous Delivery is enabled through a number of practices and components, which contribute to the entire software development process, including: software build automation; configuration management; pre-production environment use and maintenance; testing and test automation; and automated software deployment.

Among other benefits, it has been claimed and demonstrated that adoption of Continuous Delivery can: reduce cycle times in iterative software development; provide feedback into the software development and release process; and increase overall software quality.

Given that Continuous Delivery is a relatively new concept, the body of research is limited. This research aims to add to this, in addition to offering an insight into the release and development practices of software-producing businesses operating in Ireland.

**PROCEDURES:**

An interview of approximately fifty minutes to one hour in length will be conducted with participants. Audio recordings of the interview will be made so that they can later be reviewed by the researcher, but these recordings will not be made identifiable unless prior permission is sought by the researcher and granted by the participant.

All questions in this interview are optional and you are free to opt out of answering any question in the interview. Please indicate at any time if you would like to decline any question, or if you would like to end the interview. You may end the interview at any time, for any reason, without penalty.

Since the study is investigative, there is no single "correct" answer to individual questions.

**RISKS / BENEFITS:**

No risks to the participant are anticipated. The participant may gain some knowledge of Continuous Delivery, depending on their current familiarity with the practice.

**ILLICIT ACTIVITY:**

In the unlikely event that illicit activity is reported, the researcher will be obliged to report it to appropriate authorities.

**DEBRIEFING:**

A full explanation of the study is available at the participant's request after the interview has been completed. The researcher can be contacted using the details provided, should more information be required after the interview has concluded.

**ANONYMITY:**

The interview will be recorded, but recordings will not be made identifiable unless prior permission is given – the researcher will obtain permission for any specific reuse in papers, talks etc. No audio or video recordings will be made available to anyone other than the researcher / research team, without the participant's consent.

Answers given and data derived from them will be treated with full confidentiality and if published, data will not be identifiable as that of any individual or organisation.

The researcher will obtain permission for any direct quotations used, to ensure their contextual appropriateness.

**PARTICIPANT SELECTION:**

This study uses convenience sampling and participants were selected by the researcher. The research aims to gather information in interview form, from senior technical employees who have knowledge of their company's software release practices.

**POTENTIAL CONFLICTS OF INTEREST:**

The researcher's colleagues may be asked to participate in this research.

Thank you in advance for your participation in the interview.

## Participant Consent Form

**BACKGROUND OF RESEARCH:**

The research aims to investigate the software release and development practices of technology businesses in Ireland and compare them to a model of "Continuous Delivery" – a practice that aims to keep software in a state in which it can be released at any time.

The concept of Continuous Delivery first appeared in 2010 in a book by Jez Humble and David Farley, called "Continuous Delivery: Reliable Software Releases Through Build, Test and Deployment Automation".

Continuous Delivery is enabled through a number of practices and components, which contribute to the entire software development process, including: software build automation; configuration management; pre-production environment use and maintenance; testing and test automation; and automated software deployment.

Among other benefits, it has been claimed and demonstrated that adoption of Continuous Delivery can: reduce cycle times in iterative software development; provide feedback into the software development and release process; and increase overall software quality.

Given that Continuous Delivery is a relatively new concept, the body of research is limited. This research aims to add to this, in addition to offering an insight into the release and development practices of software-producing businesses operating in Ireland.

**PROCEDURES OF THIS STUDY:**

An interview of approximately fifty minutes to one hour in length will be conducted with participants. Audio recordings of the interview will be made so that they can later be reviewed by the researcher, but these recordings will not be made identifiable unless prior permission is sought by the researcher and granted by the participant and organisation.

All questions in this interview are optional and you are free to opt out of answering any question in the interview. Please indicate at any time if you would like to decline any question, or if you would like to end the interview. You may end the interview at any time, for any reason, without penalty.

**PUBLICATION:**

The researcher will obtain permission for specific reuse in papers, talks etc. Answers given and data derived from them will be treated with full confidentiality. If published, data will not be identifiable as that of any individual or organisation.

Individual results will be aggregated anonymously and research reported on aggregate results.

**DECLARATION:**

- I am 18 years or older and am competent to provide consent.

- I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.
- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.
- I understand that if I make illicit activities known, these will be reported to appropriate authorities.
- I understand that I may stop electronic recordings at any time, and that I may at any time, even subsequent to my participation have such recordings destroyed (except in situations such as above).
- I understand that, subject to the constraints above, no recordings will be replayed in any public forum or made available to any audience other than the current researchers/research team.
- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.
- I understand that I may refuse to answer any question and that I may withdraw at any time without penalty.
- I understand that my participation is fully anonymous and that no personal details about me will be recorded.
- I have received a copy of this agreement.

**PARTICIPANT'S NAME: _____**

**PARTICIPANT'S SIGNATURE:**

**_____**

**Date: _____**

**Statement of investigator's responsibility:** I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participant understands my explanation and has freely given informed consent.

**RESEARCHER'S CONTACT DETAILS:**

**Researcher's E-mail Address:** toddgr@tcd.ie
**Researcher's Contact Tel No.:** +353 (0) 879479690

**RESEARCHER'S SIGNATURE:**

**_____**

**Date: _____**

## Organisation Information Sheet

**BACKGROUND OF RESEARCH:**

The research aims to investigate the software release and development practices of technology businesses in Ireland and compare them to a model of "Continuous Delivery" – a practice that aims to keep software in a state in which it can be released at any time.

The concept of Continuous Delivery first appeared in 2010 in a book by Jez Humble and David Farley, called "Continuous Delivery: Reliable Software Releases Through Build, Test and Deployment Automation".

Continuous Delivery is enabled through a number of practices and components, which contribute to the entire software development process, including: software build automation; configuration management; pre-production environment use and maintenance; testing and test automation; and automated software deployment.

Among other benefits, it has been claimed and demonstrated that adoption of Continuous Delivery can: reduce cycle times in iterative software development; provide feedback into the software development and release process; and increase overall software quality.

Given that Continuous Delivery is a relatively new concept, the body of research is limited. This research aims to add to this, in addition to offering an insight into the release and development practices of software-producing businesses operating in Ireland.

**PROCEDURES:**

In order to investigate the release and development practices of technology businesses, the study aims to gather data from senior technical employees who have knowledge of these practices.

An interview of approximately fifty minutes to one hour in length will be conducted with participants. Audio recordings of the interview will be made so that they can later be reviewed by the researcher, but these recordings will not be made identifiable unless prior permission is sought by the researcher and granted by the participant.

All questions in this interview are optional and the participant is free to opt out of answering any question in the interview. The participant may end the interview at any time, for any reason, without penalty.

Since the study is investigative, there is no single "correct" answer to individual questions.

**RISKS / BENEFITS:**

No risks to the participant are anticipated. The participant may gain some knowledge of Continuous Delivery, depending on their current familiarity with the practice.

**ILLICIT ACTIVITY:**

In the unlikely event that illicit activity is reported, the researcher will be obliged to report it to appropriate authorities.

**DEBRIEFING:**

A full explanation of the study is available on request after the interview has been completed. The researcher can be contacted using the details provided, should more information be required after the interview has concluded.

**ANONYMITY:**

The interview will be recorded, but recordings will not be made identifiable unless prior permission is given – the researcher will obtain permission for any specific reuse in papers, talks etc. No audio or video recordings will be made available to anyone other than the researcher / research team, without the consent of the participant and organisation.

Answers given and data derived from them will be treated with full confidentiality and if published, data will not be identifiable as that of any individual or organisation.

The researcher will obtain permission for any direct quotations used, to ensure their contextual appropriateness.

**PARTICIPANT SELECTION:**

This study uses convenience sampling and participants were selected by the researcher. The research aims to gather information in interview form, from senior technical employees who have knowledge of their company's software release practices.

**POTENTIAL CONFLICTS OF INTEREST:**

The researcher's colleagues may be asked to participate in this research.

## Organisation Consent Form

**BACKGROUND OF RESEARCH:**

The research aims to investigate the software release and development practices of technology businesses in Ireland and compare them to a model of "Continuous Delivery" – a practice that aims to keep software in a state in which it can be released at any time.

The concept of Continuous Delivery first appeared in 2010 in a book by Jez Humble and David Farley, called "Continuous Delivery: Reliable Software Releases Through Build, Test and Deployment Automation".

Continuous Delivery is enabled through a number of practices and components, which contribute to the entire software development process, including: software build automation; configuration management; pre-production environment use and maintenance; testing and test automation; and automated software deployment.

Among other benefits, it has been claimed and demonstrated that adoption of Continuous Delivery can: reduce cycle times in iterative software development; provide feedback into the software development and release process; and increase overall software quality.

Given that Continuous Delivery is a relatively new concept, the body of research is limited. This research aims to add to this, in addition to offering an insight into the release and development practices of software-producing businesses operating in Ireland.

**PROCEDURES OF THIS STUDY:**

In order to investigate the release and development practices of technology businesses, the study aims to gather data from senior technical employees who have knowledge of these practices.

An interview of approximately fifty minutes to one hour in length will be conducted with participants. Audio recordings of the interview will be made so that they can later be reviewed by the researcher, but these recordings will not be made identifiable unless prior permission is sought by the researcher and granted by both the organisation and the participant.

All questions in this interview are optional. The participant is free to opt out of answering any question in the interview. The participant may end the interview at any time, for any reason, without penalty.

**RESEARCH PUBLICATION:**

The researcher will obtain permission for specific reuse in papers, talks etc. Answers given and data derived from them will be treated with full confidentiality. If published, data will not be identifiable as that of any individual or organisation.

Individual results will be aggregated anonymously and research reported on aggregate results.

**DECLARATION:**

- I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.
- I agree that the data collected is used for scientific purposes and I have no objection that the data is published in scientific publications in a way that does not reveal the identity of the organisation.
- I understand that if illicit activities are made known, these will be reported to appropriate authorities.
- I understand that the participant may stop electronic recordings at any time, and that I may at any time, even after participation, have such recordings destroyed (except in situations such as above).
- I understand that, subject to the constraints above, no recordings will be replayed in any public forum or made available to any audience other than the current researchers/research team.
- I freely and voluntarily allow data provided in this interview to be part of this research study, though without prejudice to my legal and ethical rights.
- I understand that the participant may refuse to answer any question and may withdraw at any time without penalty.
- I understand that the organisation's participation is fully anonymous.
- I have received a copy of this agreement.

**SIGNED:**

**_____**

**(PRINT NAME): _____**

**ON BEHALF OF ORGANISATION:**

**_____**

**Date: _____**

**Statement of investigator's responsibility:** I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the organisation's representative understands my explanation and has freely given informed consent.

**RESEARCHERS CONTACT DETAILS:**

**Researcher's E-mail Address:** toddgr@tcd.ie
**Researcher's Contact Tel No.:** +353 (0) 879479690

**RESEARCHER'S SIGNATURE:**

**_____**

**Date: _____**

# Appendix 2 – Interview Questions

### Company Questions

"Can you give me an overview of your company's business?"

"How long has the company been in business?"

"What is your personal role in the company?"

"What size is the company in terms of turnover?"

"And what size is the company in number of employees?"

"Can you give an overview of the software produced?"

- "Is the software a hosted application?"
- [If hosted] "Where is it hosted (locally or remotely; self-hosted; cloud-hosted; hybrid)?"

"How many employees are involved in software development, or support software development in some way (including technical management, IT Operations, technical project or product managers)?"

### Software Build Management and Continuous Integration

"How is your software built? Is it a manual or automated process?"

"Are the binaries, reports (or textual output) produced by the build process managed in some way?"

"Are builds created regularly? Is this process automated?"

"Are tests run regularly? Is this process automated?"

"Are you able to accurately recreate previous software builds?"

- "In this case, would the build be recreated from source control?"
- "Is this process automated?"

"Do you use source control?"

- [If using source control and automated building] “Do you perform a software build on every commit (or change in source control)?”
- [If using source control and automated testing] “Are automated test triggered on every commit (or change in source control)?”

“Do you use the same build script for every build that takes place?”

“Do you gather any build metrics?” [If prompted what these might be; “For example, time taken to produce the build, tests passed and time taken for each, build outcome for each build?”]

- [If build metrics gathered…] “Are these metrics made visible to the development teams?”
- [If build metrics gathered…] “Are there any actions taken on these metrics, or policies around them? For example, is there a policy that states that programmers must act immediately to fix a broken build?”

“Are there any regular meetings to discuss improvement of the build process?” [If so…] “How often are these held?”

*“Can you describe you overall build process, including any processes followed and documentation completed? When development has completed, what is the process for producing the new build of your software?”*

**Environments and Deployment**

“What different software environments does a software build go through before reaching the production environment?”

“How are these environments created or provisioned? Is provisioning automated?”

“Is a separate binary produced for each environment?”

“How is the software deployed to these environments?”

- [If automated…] “Is deployment automated to all environments?”

[Possible further question on creation of environments, if not covered by initial question in this section: “How are new environments created or provisioned?”]

“Are the configuration changes needed for each environment version controlled?”

“Is the deployment process the same for all environments?”

“How do you manage deployments to multiple dependent services at once (or orchestrated deployments)?”

“Do you test release processes?”

“Do you test rollback processes?”

[If provisioning is automated…] “Is provisioning fully automated to all environments?”

*“On the way to production, can you outline the different environments that a new release might progress through at various stages? Also what process is followed to migrate potentially-releasable software from one of these environments to another?”*

**Release Management and Compliance**

“How often do you release to production?”

“Are releases are reliable once deployed?”

“Do you have the ability to trace the origin of a release back to the original requirements / bug report / request for change? If so, how is this done?” [Determine whether traceability is comprehensive].

“Can you describe any release / change management processes that you current have in place? Is this process strictly enforced?”

“Do you have any exceptional regulatory compliance measures that you are required to meet? How difficult is it for you to meet these as a company?”

“Do you continually monitor application health?”

- [If monitored] “Is this proactive? Do you gather metrics on its health? Or are you notified on error?”

“Do you continually monitor environment health?”

- [If monitored] “Again, is this proactive? Do you gather metrics on its health? Or are you notified on error?”

“Do you monitor cycle time or ‘the time tasks take to progress from requirements to implementation’?”

“Do delivery / development teams and operations teams collaborate regularly to proactively manage and improve upon the release processes?” [If so…] “How often does this take place?”

**Testing**

“Is software testing a manual process, an automated process, or a combination of both?”

“Are the environments used for software testing similar to the production environment? Are these available early in the software project’s lifecycle?”

[If not answered by the first question in this section] “Do you use automated unit and acceptance testing?”

“Is testing an integral part of the development process?”

“Do you track software quality metrics?”

“Do you test software against non-functional requirements?” [If asked what these might be “Performance testing, security testing and usability testing are common non-functional requirements.”]

“How often do you have to perform a rollback of a software release? (Do you track these?)”

“How common is it to find a production defect? What is the process to fix these; are they addressed immediately?”

*"Can you outline any testing that a potential software change goes through before it is released to production?"*

**Data Management**

"Are database migrations versioned?"

"Are database changes versioned with the main application?"

"Are database changes performed using scripts?"

"Are database changes performed manually, or via an automated process?"

"Is database performance monitored?"

- [If so…] "Is database performance optimised as a result of this?

"Do you test database upgrades and rollbacks on every deployment?" [Determine if either one of these is tested on every deployment.]

"Do you review the database deployment process itself regularly, considering recent releases?" [If so…] "How often does this take place?"

*"How do you manage changes to data that are integral to software version changes? For example, if a change to a software service also requires a change to the database, how this is handled at the time of release?"*

**Configuration Management**

"Do you use source control (or version control) as part of the software development process?"

- [If not…] "How do you manage different versions of the software?"
- [If so…] "How often do developers check-in or commit changes to source control?"

- [If so…] “What is kept under version control, apart from code itself?” [If examples requested: “For example, are build configurations, deploy scripts or data migrations versioned?”]

“How do you manage the use of external libraries within the development process?”

“How do you manage dependencies between separate, dependent software components within the company’s own software?”

[If change management used (“Release Management and Compliance” section)…] “Does your change management process mandate particular procedures for version control use?”

“How frequently do developers commit or merge to mainline (also known as ‘master’ or ‘trunk’?”

“Do you have a particular policy around source code branching? If so, can you describe the strategy?”

“Is there a regular review of the source control and configuration management processes?” [If so…] “How often does this take place?”

*“How valuable would you say the practices we have discussed here today are to you as a company? What benefits do you gain from them?”*

*Further exploratory questions covering the above already-covered topics, guided by participant responses.*

## Appendix 3 – The Humble and Farley Maturity Model

| Practice | Build management and continuous integration | Environments and deployment | Release management and compliance | Testing | Data management | Configuration management |
|---|---|---|---|---|---|---|
| **Level 3 - Optimizing:** Focus on process improvement | Teams regularly meet to discuss integration problems and resolve them with automation, faster feedback, and better visibility. | All environments managed effectively. Provisioning fully automated. Virtualization used if applicable. | Operations and delivery teams regularly collaborate to manage risks and reduce cycle time. | Production rollbacks rare. Defects found and fixed immediately. | Release to release feedback loop of database performance and deployment process. | Regular validation that CM policy supports effective collaboration, rapid development, and auditable change management processes. |
| **Level 2 - Quantitatively managed:** Process measured and controlled | Build metrics gathered, made visible, and acted on. Builds are not left broken. | Orchestrated deployments managed. Release and rollback processes tested. | Environment and application health monitored and proactively managed. Cycle time monitored. | Quality metrics and trends tracked. Non functional requirements defined and measured. | Database upgrades and rollbacks tested with every deployment. Database performance monitored and optimized. | Developers check in to mainline at least once a day. Branching only used for releases. |
| **Level 1 - Consistent:** Automated processes applied across whole application lifecycle | Automated build and test cycle every time a change is committed. Dependencies managed. Re-use of scripts and tools. | Fully automated, self-service push-button process for deploying software. Same process to deploy to every environment. | Change management and approvals processes defined and enforced. Regulatory and compliance conditions met. | Automated unit and acceptance tests, the latter written with testers. Testing part of development process. | Database changes performed automatically as part of deployment process. | Libraries and dependencies managed. Version control usage policies determined by change management process. |
| **Level 0 – Repeatable:** Process documented and partly automated | Regular automated build and testing. Any build can be re-created from source control using automated process. | Automated deployment to some environments. Creation of new environments is cheap. All configuration externalized / versioned | Painful and infrequent, but reliable, releases. Limited traceability from requirements to release. | Automated tests written as part of story development. | Changes to databases done with automated scripts versioned with application. | Version control in use for everything required to recreate software: source code, configuration, build and deploy scripts, data migrations. |
| **Level -1 – Regressive:** processes unrepeatable, poorly controlled, and reactive | Manual processes for building software. No management of artifacts and reports. | Manual process for deploying software. Environment-specific binaries. Environments provisioned manually. | Infrequent and unreliable releases. | Manual testing after development. | Data migrations unversioned and performed manually. | Version control either not used, or check-ins happen infrequently. |

The Humble and Farley Continuous Delivery Maturity Model (Humble and Farley 2010, p.419). Key process areas are listed vertically; “Build Management and Continuous Integration” and “Data Management” are examples of key process areas in this model. Maturity levels are listed horizontally. Key practices are listed in the body of the model. For example, “defects found and fixed immediately” is an example of a key practice at maturity level 3 of the “Testing” key process area.

## Appendix 4 – Maturity Model Used in this Research

| Key process areas: | Build Management and CI | Environments and Deployment | Release Management and Compliance | Testing | Data Management | Configuration Management |
|---|---|---|---|---|---|---|
| **Level 3** | Teams regularly meet to discuss improvement of the build process. | Provisioning fully automated. | Regular collaboration to manage risks and reduce cycle time. | Production rollbacks are rare. Defects found and fixed immediately. | Release to release feedback loop of database performance and deployment process. | Regular review of CM policy |
| **Level 2** | Build metrics gathered, made visible and acted upon. Builds are not left broken. | Orchestrated deployments managed. Release and rollback processes tested. | Frequent releases. Environment and application health monitored and proactively managed. Cycle time monitored. | Quality metrics and trends tracked. Non-functional requirements defined and measured. | Database upgrades and rollbacks tested with every deployment. Database performance monitored and optimised. | Developers check in to mainline at least once a day. Branching only used for releases. |
| **Level 1** | Automated build and test cycle every time a change is committed. Re-use of build scripts and tools. | Fully automated, push button process for deploying software to all environments. Same process to deploy to every environment. | Change management and approval processes defined and enforced. Regulatory and compliance measures met. Full traceability from requirements to release. | Automated unit and acceptance tests. Testing an integral part of development process. | Database changes performed automatically as part of deployment process. | External libraries and internal dependencies managed. Version control usage policies determined by change management process. |
| **Level 0** | Regular automated build and testing. Any build can be recreated from source control using automated process. | Automated deployment to some environments. Cheap creation of new environments. All configuration versioned. | Infrequent, but reliable releases. Limited traceability from requirements to release. | Some automated testing. Production-like environments available for testing. | Changes to databases done with automated scripts, versioned with application. | Version control in use for everything required to create software. |
| **Level -1** | Manual processes for building software. No management of artifacts and reports. | Manual process for deploying software. Environment-specific binaries. Environments provisioned manually. | Infrequent and unreliable releases. | Manual testing after development. | Data migrations unversioned and performed manually. | Version control either not used, or check-ins happen infrequently. |

# Appendix 5 – Results of Statistical Analysis

## Mobile vs Web Platform Development at Company F – Paired Samples T-Test

The following analysis was excluded from the body of this research for brevity. Results were produced using SPSS.

**Case Summaries**

| Practice No. | PlatformEnvAndDep | MobileEnvAndDep |
|---|---|---|
| 15 | 2.00 | .00 |
| 16 | .00 | .00 |
| 17 | 2.00 | 1.00 |
| 18 | 2.00 | .00 |
| 19 | 4.00 | 4.00 |
| 20 | 2.00 | 2.00 |
| 21 | .00 | .00 |
| 22 | 2.00 | 1.00 |
| 23 | 2.00 | .00 |
| 24 | .00 | .00 |
| 25 | .00 | .00 |
| 26 | .00 | .00 |
| Total N | 12 | 12 |

**Paired Samples Test**

| | | Paired Differences | | | |
|---|---|---|---|---|---|
| | | 95% Confidence Interval of the Difference | | | |
| | | Upper | t | df | Sig. (2-tailed) |
| Pair 1 | PlatformEnvAndDep - MobileEnvAndDep | 1.23064 | 2.602 | 11 | .025 |

## Chi-Squared Tests

Contingency tables for all chi-squared tests are available in the body of the research. See the List of Tables and Diagrams for further information.

## Appendix 6 – Interview Answer Weightings

| Key Process Area: Build Management and CI | | | |
|---|---|---|---|
| **Practice No.** | **Practice** | **Weighting** | **Notes** |
| 1 | Regularly meet to improve build and CI processes | 2 | Bimonthly or more frequently considered “regular” |
| 2 | Build metrics gathered | 2 | |
| 3 | Build metrics made visible | 2 | |
| 4 | Broken build acted upon | 2 | |
| 5 | Other build metrics acted upon | 2 | |
| 6 | Automated build on every commit | 4 | Scoring on this practice prevents scoring on Practice 9 |
| 7 | Automated testing on every commit | 4 | Scoring on this practice prevents scoring on Practice 10 |
| 8 | Re-use of build scripts and tools | 2 | |
| 9 | Regular automated build | 2 | Scoring on this practice prevents scoring on Practice 6 |
| 10 | Regular automated testing | 2 | Scoring on this practice prevents scoring on Practice 7 |
| 11 | Any build can be recreated from source | 2 | |
| 12 | Build recreation is via automated process | 2 | |
| 13 | Manual build processes | 0 | Maturity level -1 |
| 14 | No management of binaries and reports | 0 | Maturity level -1 |
| | | | |
| | **Key Process Area Total** | **24** | |

**Key Process Area: Environments and Deployment**

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| 15 | Provisioning fully automated | 2 | |
| 16 | Orchestrated deployments managed | 2 | |
| 17 | Release processes tested | 2 | |
| 18 | Rollback processes tested | 2 | |
| 19 | Automated process for deploying software to all environments | 4 | Scoring on this practice prevents scoring on Practice 21 |
| 20 | Same process to deploy to every environment | 2 | |
| 21 | Automated deployment to some environments | 2 | Scoring on this practice prevents scoring on Practice 19 |
| 22 | Cheap creation of new environments | 2 | |
| 23 | All configuration externalised and versioned | 2 | |
| 24 | Manual processes for deploying | 0 | Maturity level -1 |
| 25 | Environment-specific binaries | 0 | Maturity level -1 |
| 26 | Manual provisioning | 0 | Maturity level -1 |
| | | | |
| | **Key Process Area Total** | **18** | |

**Key Process Area: Release Management and Compliance**

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| 27 | Regular collaboration to manage risks and reduce cycle time | 2 | Bimonthly or more frequently considered “regular” |
| 28 | Environment health monitored | 2 | |

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| 29 | Application health monitored | 2 | |
| 30 | Environment health proactively managed | 2 | |
| 31 | Application health proactively managed | 2 | |
| 32 | Cycle time monitored | 2 | |
| 33 | Change management / approval processes defined | 2 | |
| 34 | Change management / approval processes enforced | 2 | |
| 35 | Regulatory and compliance conditions NOT met easily | -2 | |
| 36 | Frequent releases | 4 | Fortnightly or more frequently = “frequent”. Scoring for this practice prevents scoring on 39 |
| 37 | Full traceability from requirements to release | 4 | Scoring on this practice prevents scoring on Practice 38 |
| 38 | Limited traceability from requirements to release | 2 | Scoring on this practice prevents scoring on Practice 37 |
| 39 | Infrequent, but reliable releases | 2 | Less frequently than fortnightly = “Infrequent”. Scoring for this practice prevents scoring on Practice 36 |
| 40 | Infrequent and unreliable releases. | 0 | Maturity level -1 |
| | | | |
| | **Key Process Area Total** | **24** | |

**Key Process Area: Testing**

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| 41 | Production rollbacks are rare | 2 | |
| 42 | Defects found and fixed immediately | 2 | |
| 43 | Software quality metrics and trends tracked | 2 | |

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| 44 | Non-functional requirements defined and measured | 2 | |
| 45 | Automated unit and acceptance tests | 4 | Scoring on this practice prevents scoring on Practice 49 |
| 46 | Testing is an integral part of development process | 2 | |
| 48 | Production-like environments available for testing | 2 | |
| 49 | Some automated testing | 2 | Scoring on this practice prevents scoring on Practice 45 |
| 50 | Manual testing after development | 0 | Maturity level -1 |
| | | | |
| | **Key Process Area Total** | **16** | |

**Key Process Area: Data Management**

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| 51 | Feedback loop of database performance and deployment process | 2 | Bimonthly or more frequently review results in scoring 1 |
| 52 | Database upgrades tested with every deployment | 2 | |
| 53 | Database rollbacks tested with every deployment | 2 | |
| 54 | Database performance monitored | 2 | |
| 55 | Database performance optimised | 2 | |
| 56 | Database changes performed automatically during deployment | 2 | |
| 57 | Databases changes performed with automated scripts | 2 | |
| 58 | Databases changes versioned with application | 2 | |
| 59 | Data migrations not versioned and performed manually | 0 | Maturity level -1 |

| Practice No. | Practice | Weighting | Notes |
|---|---|---|---|
| | | | |
| | **Key Process Area Total** | **16** | |

**Key Process Area: Configuration Management**

| Practice No. | Practice | Weighting | |
|---|---|---|---|
| 60 | Regular CM policy review | 2 | Bimonthly or more frequently considered “regular” |
| 61 | Developers commit or check-in to mainline at least once a day | 2 | |
| 62 | Branching only used for releases | 2 | |
| 63 | External libraries managed | 2 | |
| 64 | Internal dependencies managed | 2 | |
| 65 | Version control determined by change management process | 2 | |
| 66 | Version control in use for everything required to create software | 2 | |
| 67 | Version control either not used, or check-ins happen infrequently | 0 | |
| | | | |
| | **Key Process Area Total** | **14** | |

| **TOTAL (All Key Process Areas)** | **112** | |
|---|---|---|