An Empirical Case Study of Technical Debt Management: A Software Services Provider Perspective

David O’Keeffe

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 2017

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work, and has not been submitted as an exercise for a degree at this or any other university. I further declare that this research has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics.

Signed: David o'Keeffe

David O'Keeffe
1st September 2017

Permission to lend and/or copy

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

Signed David o'Keeffe ______________________

David O’Keeffe

1st September 2017

Acknowledgements

I would like to specially thank my family and my partner, Nicola, for their patience and words of encouragement over the past two years.

I would also like to thank my supervisor, Diana Wilson, for her advice and guidance whilst undertaking this dissertation.

Finally, I would like to thank the participants that took part in this research as without them this dissertation would not have been possible.

Abstract

Financial debt plays an important and positive role in our economy under normal conditions. In software development, the “Technical Debt” (TD) metaphor is gaining traction amongst software practitioners and researchers in recent years. The objective of this study is to investigate how senior software practitioners with over ten years’ experience understand TD and explicitly conduct technical debt management (TDM) in the context of an Irish software services provider. In this qualitative study, one Irish software development provider is studied to gather empirical evidence on TD understanding and how TDM is conducted. An exploratory case study method is used for data collection and analysis of the case organisation by conducting interviews with 14 software practitioners and examining 34 case organisation documents. The main findings suggest that understanding and attitudes towards TD and TDM are generally positive. Organisational factors that influence TDM include risk appetite, culture, client relationship and management commitment. Six main TDM techniques are used for eight TDM activities. Finally, a framework for TDM is proposed based on the research findings. There is a requirement for identifying methods to overcome the identified TDM challenges. TD needs to be managed and controlled effectively to sustain software development and to ultimately achieve business objectives. Future studies should incorporate empirical studies to validate TDM techniques in genuine software development projects for specific industries.

Keywords: Technical Debt Management, Software Engineering, Organisational Factors Software Debt, Technical Debt Techniques, Technical Debt Challenges

Table of Contents

List of Tables

List of Figures

Abbreviations

LEO	Lyons Electronic Office
TD	Technical Debt
TDM	Technical Debt Management
SDLC	Software Development Life Cycle
IT	Information Technology
ICT	Information Communications Technology
XP	Extreme Programming
CMM	Capability Maturity Model
ASA	Automated Static Analysis
CAQDAS	Computer-Assisted Qualitative Data Analysis Software
T&M	Time and Materials
DSCR	Debt Service Coverage Ratio
TDD	Test-Driven Development
KT	Knowledge Management
VOIP	Voice Over Internet Protocol

Chapter 1: Introduction

This research examines the understanding and management of technical debt (TD) in a real software development setting. It aims to explore how TD is being managed by software development practitioners, the organisational factors that influence TDM and any challenges that it presents.

1.1 Background to the Study

The first computer to be used for commercial business applications in 1951 (Lavington et al., 2009), the "LEO I", was initially used to calculate the costs of a large catering and food manufacturers' weekly bakery distribution run. The functionality of LEO I evolved to include payroll and inventory calculations shortly after its introduction and eventually progressed to conducting scientific computations. continue to evolve. The phenomenon of software evolution has since become an important topic.

Lehman (1980) identified a set of observations in the evolution of proprietary software. The laws are referred to as Lehman's Laws. The laws have continued to develop and there are now eight in total (Lehman, 1996; Liguo et al., 2013). The laws describe a balance between forces progressing new software developments and those that obstruct progress. Two of these laws which are particularly relevant to TDM are defined below.

1. The law of continuing change - Any software system used in the real-world must change or become less and less useful in that environment
2. The law of increasing complexity - As time flows forwards, entropy increases. That is, as a program evolves, its structure will become more complex. Just as in physics, this effect can, through great cost, be negated in the short term.

(Lehman, 1980)

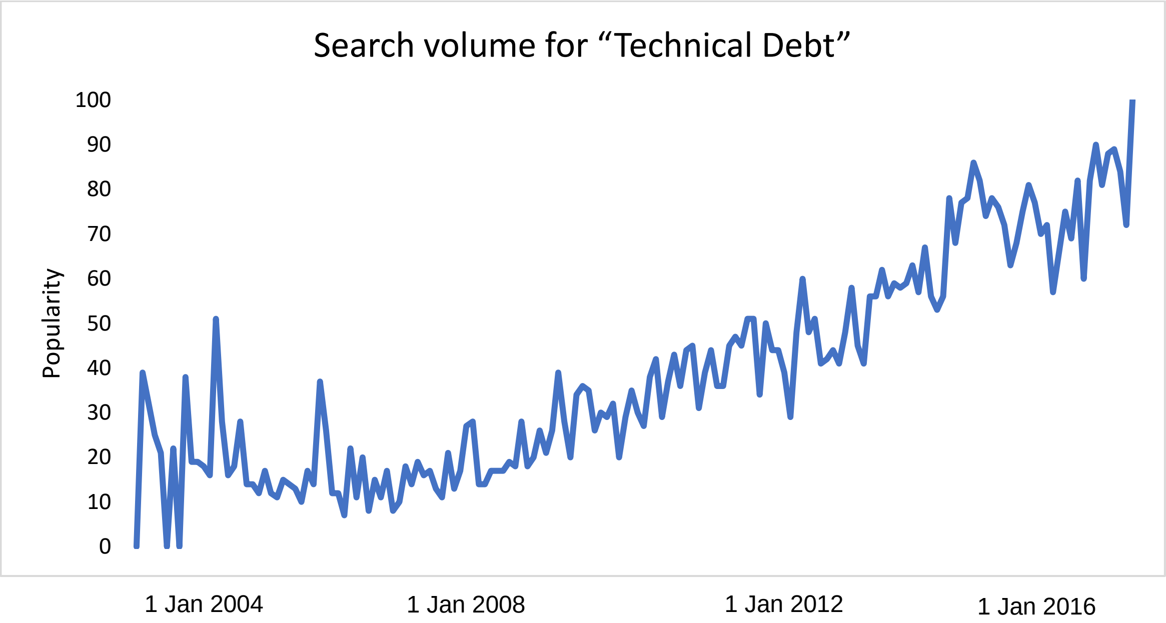
Software development and support projects are constrained by skills, budget and deadlines. Compromises are often made to manage these constraints throughout the software development lifecycle (SDLC). A shortcut or a workaround can give organisations a benefit in the short-term with a more expedient software release to the customer and possibly an advantage over competitors in time-to-market (Kruchten et al., 2012; Yli-Huumo et al., 2015). These shortcuts incur a "technical debt" in the software that should be repaid to restore the quality of the software system in the future to avoid “interest” in the form of decreasing productivity of the development team and maintainability (Seaman et al., 2012).

Competitive markets often force organisations to deliver software to short schedules and shorter delivery cycles. This continuous pressure creates a cycle of TD accrual which is often not repaid due to new customer requirements and maintenance and support required for the existing system. Delivering perfect code is not often achievable due to the above constraints. Doing so may risk delays and customer dissatisfaction. This could ultimately have implications for the software service providers’ commercial interests (Yli-Huumo et al., 2016). Due to this, it is necessary for organisations to develop procedures and processes to contain and control the TD. Technical Debt Management (TDM) consists of activities, processes, techniques, and tools that can be used to identify, measure, prevent and reduce TD in software systems (ibid).

The Information & Communication Technology (ICT) sector in Ireland directly employs over 105,000 people. Computer services are responsible for 22% of our national exports. It is one of the fastest growing sectors of the Irish economy, with employment up 40% since 2010 (collinsmcnicholas.ie, 2016). Software service providers have been a popular choice for Irish businesses in recent years. Firms are increasingly choosing to outsource their Information Technology (IT) support, development and advisory requirements to software service providers. A recent survey shows 51 percent of Irish companies are increasing software spending and are now five percent above the global average (Taylor, 2017). In 2016, the Irish government department of public expenditure and reform (Per.gov.ie, 2016) had purchase orders totalling just under €17 million for shared software services. This figure consists of purchase orders of €20,000 and over and does not include purchase orders under this figure. Due to this trend, the role of the software service providers in the management of TD is an increasingly important one (Babu, 2016).

1.2 Significance and Rationale of the Research

There is an increasing dependence on information systems in both the society and the economy with software evolution success and maintenance becoming more critical (West, 2015). Central to software evolution success is the controlling and management of TD. Interest in the TD concept has become more prevalent in recent years as illustrated in Figure 1. Currently, the term is in peak popularity. However, there is a lack of empirical evidence about how to manage this TD in real-life software development settings (Li et al., 2015).



Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity for the term. A value of 50 means that the term is half as popular. Likewise, a score of 0 means the term was less than 1% as popular as the peak (Google Trends, 2017).

FIGURE 1 - Search volume for the term "technical debt" since 2004 to present

The research objective of this work is to investigate the understanding of TD and how TDM is conducted by senior software practitioners in an Irish software company. This investigation aims to:

1. Examine the understanding of TD in a real setting
2. Explore the attitudes of senior software practitioners towards TD and its management
3. Identify the organisational factors that influence the TDM activities
4. Discover the techniques used to conduct TDM activities
5. Identify the challenges of explicit TDM

It is hoped that this experimental research can bring a deeper insight into the use of explicit TDM by software practitioners and will ultimately lead to greater efficiencies across software delivery. This may have an impact on cost of the software development process and may result in efficiencies and ultimately cost savings for public and private organisations.

1.3 Research Question

The following primary research question will be examined in this study:

"How do senior software practitioners with over ten years' experience understand TD and explicitly conduct TDM in the context of an Irish software services provider?"

Four secondary questions can be derived:

RQ1. How do senior software practitioners view the TD concept and its management?

RQ2. What organisational factors influence the management of Type II TD?

RQ3. What methods and techniques are used for explicit TDM activities?

RQ4. What challenges exist in the management of TD?

1.4 The Scope of the Research

This empirical qualitative research focuses on the exploration of how TDM is perceived and conducted in a real setting. Senior software practitioners with ten or more years' software development experience were chosen for the semi-structured interviews. The reason for this is due to observation by Taksande (2011) that more concrete and detailed experiences with TD could be gathered and studied form participants who had more than ten years of experience in an IT setting. The research also focuses mainly on intentionally accrued source code TD as opposed to TD which was accrued inadvertently. This will be discussed further in section 2.2.2.

1.5 Beneficiaries of this research

This investigation provides an insight into how TDM is conducted in an Irish software services provider across multiple software development and support projects. Thus, the primary beneficiaries are expected to be software practitioners who work as part of software development and support teams to deliver software products and services to customers. Chief information officers would benefit from this research as they are responsible for their organisations information systems TD. It is also pertinent to researchers investigating the topic of TD and its related management activities.

1.6 Chapter Structure

The dissertation is structured as follows:

Chapter 1: Introduction

This chapter introduces the context and rationale for choosing the research question. It outlines the relevant background to the research question and why the chosen focus is important. It also outlines the scope of the study and who is likely to benefit from the findings.

Chapter 2: Literature Review

This chapter reviews important and relevant literature relating to the research question and provides a frame of reference to existing TD literature, stressing the lack of empirical studies and definitions. The chapter explores the TD definition, the TD phenomenon, TD taxonomy, factors influencing TDM, techniques for conducting TDM and challenges that it presents. It also explores the theoretical background to the research question.

Chapter 3: Research Methodology

This chapter provides a brief overview of the research philosophies, methodologies and strategies available. It explains the reason for the chosen research methodology as well as the merits and limitations of choosing such an approach.

Chapter 4: Findings and Analysis

This chapter presents the research findings and analyses the data that was collected from the interviews and documents.

Chapter 5: Conclusions and Future Work

This chapter concludes the dissertation by discussing the findings of the research and interpreting the findings in the context of the existing literature. It also determines whether the research findings have answered the research question and contains recommendations for potential future research areas in the field.

Chapter 2: Literature Review

2.1 Introduction

This chapter presents a comprehensive review of the literature relevant to TD and its management. This chapter will highlight the key themes and trends to emerge from previous recent studies on the above areas of TD. The literature review includes journal articles, conference proceedings, blogs, books and edited volumes. The research goes in-depth to focus on the TD understanding, organisational factors that impact TDM, techniques for TDM and challenges that TDM presents. Relevant literature was identified by searching databases (Table 1) for terms such as “Technical Debt” AND “Management” excluding articles that were not available in the English language (102 articles for the search string ‘t:(technical debt) (management) l: eng)’.

TABLE 1 – Searched Databases

ID	Database	Searched	Selected
DB1	IEEE Xplore Digital Library	YES	YES
DB3	ScienceDirect	YES	YES
DB4	Scopus	YES	YES
DB5	SpringerLink	YES	YES
DB6	ACM Digital Library	YES	YES
DB7	Web of Science	NO	NO

This literature review examines the existing research under the following topics:

- What is Technical Debt?
- Technical Debt Management (TDM)
- Attitudes towards TDM
- TDM Organisational factors
- TDM Techniques
- Challenges presented by TDM

2.2 What is Technical Debt?

The purpose of this section is to explain the term “Technical Debt” (TD) through concepts defined by other authors and through examples in existing literature.

2.2.1 Technical Debt Definition

The definition of TD has been interpreted differently, refined and presented by various researchers and practitioners (Kruchten et al, 2013). Despite this, there is still no agreed consensus in both technical (McConnell, 2007; Fowler, 2009; Martin, 2009) and scientific (Brown et al., 2010; Codabux and Williams, 2013; Izurieta et al., 2012) literature regarding what could be considered a TD case due to the original metaphors spectrum of interpretations. Martin (2009) interprets the TD metaphor as trade-offs that the technical and business teams must consider to be able to meet time constraints and customer expectations. TD should refer only to cases where poor designs are defined for strategic reasons but correctly implemented. In contradiction to McConnell (2007), Martin proposed dirty code should not be considered TD. Siebra et al (2016) agree that TD should be associated only with intentional decisions happening in the code base, with messy code not counted as TD (See Figure 2).

Others believe that old technologies in legacy software should also be classified as TD (Norton, 2009; Fowler, 2009). Guo et al. (2014) hold a broad view of the TD phenomenon, having defined TD as any immature, incomplete or inappropriate activity that affects the subsequent development and maintenance of software. Theodoropoulos et al (2011, p.45) expanded this definition with the inclusion of ‘any gap in the technology infrastructure’. The TD metaphor has limits however. Rooney (2010) argues that it is not sufficient to describe TD which is accrued using modern development approaches such as Extreme Programming (XP). Debt accumulated this way is often paid off shortly after it has been incurred. The XP approach is rarely used in real software development settings however. Allman (2012) believes people who decide to accrue TD are usually not the ones who repay the debt. This may encourage people to take more TD to accelerate software development. Schmid (2013) points out that there is no standard TD unit of measurement and that the interest amount needing to be repaid is dependent on the future development that is effected by the TD.

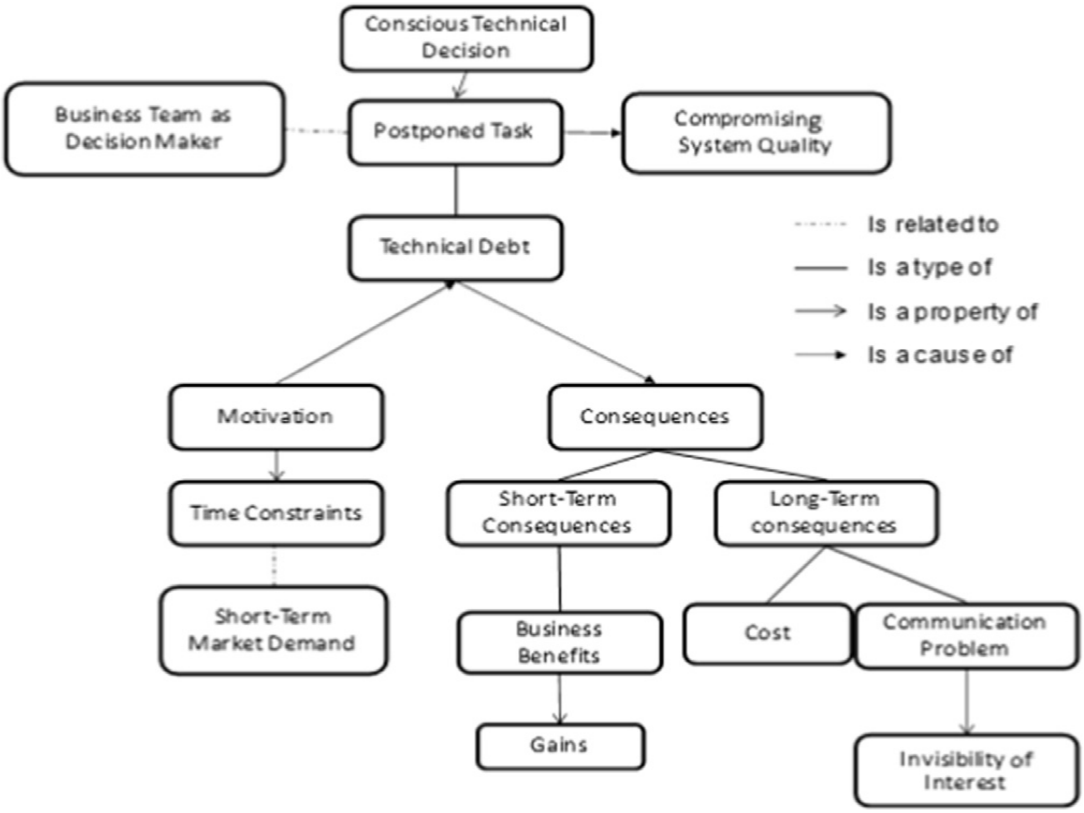


FIGURE 2 – Conceptual Model of TD

(Siebra, 2016).

In recent literature, researchers are attempting to define TD in software engineering sub-area contexts. Alves et al (2014; 2016) summarised fifteen identified TD types in a mapping study findings which include service debt, build debt and process debt. Ernst (2012) examined TD in software requirements gathering. TD in this area is defined as the distance between the system implementation and the actual state of the world (ibid). Li et al (2015) focused on architectural TD. More recently, Maldonado et al (2015) examined self-admitted TD sub-areas by analysing 33,000 developer comments in genuine source code.

Presently, the metaphor is still developing. Siebra (2016) describes TD as a method to manage and communicate long-term consequences that some technical decisions may cause. Heinz (2016) compares working on a technically indebted software system to running through mud. One consequence is low speed, because the mud has high friction. The second consequence is that mud is unstable, making injury likely due to falling. These consequences are metaphors for changing systems that have high TD. Due to TD, nearly every aspect of the system is more difficult to develop, slower, and more dangerous as there is a higher failure risk in production and problematic system maintenance.

Although the TD metaphor is intuitive, the differing meaning interpretations enforces the argument that the term is not well defined at present. Different interpretations and uses of TD concepts is facilitating over-extension. Schmid (2013) approached TD metaphor limits and the problems that arise when over-extending its applicability. Schmid (ibid.) proposes ways of handling this weakness in TD definition, based on his own perspective and experience, rather than from a formal theory. In a study conducted by Ernst et al (2015), it was found that TD is commonly understood at an abstract level to convey urgency about accumulating software costs. However, software practitioners do not agree on what project elements constitute TD. The most recent annual seminar on managing TD developed the following working TD definition:

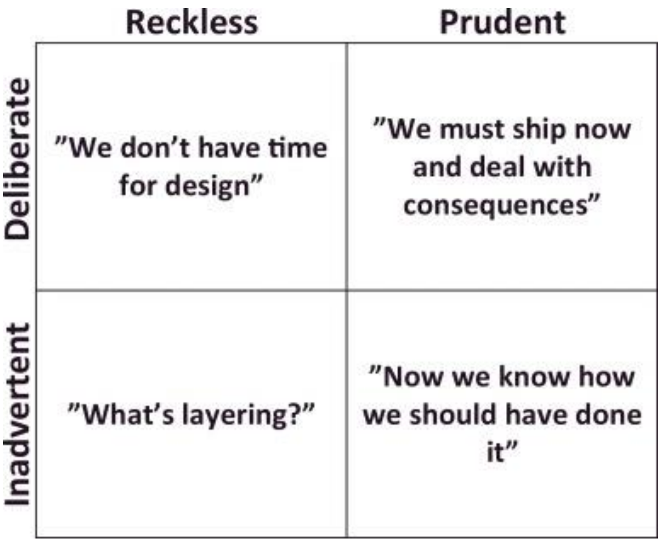
"In software-intensive systems, technical debt is the collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability that impacts internal system qualities, primarily maintainability and evolvability." (Nord, 2016)

The above TD definition restricts TD to phenomena closely connected to source code, such as code design and architectural debt. This leaves out other important TD types that are at least partially comparable to financial debt but are outside the scope of this research, e.g. People Debt, Process Debt, and Infrastructure Debt (Avgeriou et al., 2016). Overall, current TD conceptualizations differ. There is no clear and universally accepted definition. This is highlighted by the varying explicit and implicit TD definitions of in academic literature (Tom et al., 2013). The TD metaphor interpretations lack clarity, definition and represent a barrier to efforts to model and conduct TDM. Tom et al (2013) cited a need to rigorously define and validate the TD concept in both the practitioner community as well as academia. A literature critique in TD classification is outlined in the Section 2.2.2.

2.2.2 *Technical Debt Taxonomy*

Early TD research focused on establishing the foundations by conceptualising the phenomenon and classification (Guo et al, 2014). The literature broadly agrees that there are two primary TD categories, although the categories naming convention vary. McConnell (2007) describes them as intentional and unintentional whilst Fowler (2009), positions them as deliberate versus inadvertent. Most debt categories identified in Fowler's TD quadrant (Figure 3) can be mapped to the debt 'types' in McConnell's TD taxonomy (Figure 4). For example, Fowlers reckless-inadvertent debt is equivalent to McConnell's unintentional debt

(Type I); similarly, reckless-deliberate debt is the same as intentional short-term debt (Type II.A) and deliberate-prudent debt maps to intentional long-term debt (Type II.B). However, Fowlers TD quadrant captures a TD type that does not appear in McConnell's taxonomy: prudent-inadvertent debt. Prudent-inadvertent debt is TD that a project accrues when it develops in a manner that reflects the team's present knowledge of the problem, but is recognized as TD after the team has acquired more knowledge about the problem. This categorisation was identified as the most common TD categorisation in one study and may be explained by agile methodology popularity and the failure to implement it well around that time (Ernst, 2015). These categories will be referenced throughout this paper as the TD accrual categories, with this research focusing on deliberate\intentional TD accrual.



	Reckless	Prudent
Deliberate	"We don't have time for design"	"We must ship now and deal with consequences"
Inadvertent	"What's layering?"	"Now we know how we should have done it"

FIGURE 3 - Technical Debt Quadrant (Fowler, 2009)

McConnell (2007), in his blog, expanded Cunningham's metaphor into the following TD taxonomy:

i. *Unintentional Debt* - Debt incurred unintentionally due to low quality work (Non-strategic)

ii. *Intentional Debt* - Debt incurred intentionally (strategic)
   a. *Short-term Intentional Debt* - debt usually incurred reactively, for tactical reasons
      i. *Focused Short-Term Intentional Debt* - Individually identifiable shortcuts (like a car loan)
      ii. *Unfocused Short-Term Intentional Debt* - Numerous tiny shortcuts (like credit card debt)
   b. *Long-term Intentional Debt* - Debt usually incurred proactively, for strategic reasons

FIGURE 4 – McConnell's TD taxonomy

TD is often seen as a negative concept in software development (Lim et al., 2012; Yli-Huumo et al., 2014). However, TD can have both positive and negative impacts on a software project. TD that is intentionally incurred to achieve short-term benefit, can be constructive (Allman, 2012a) with the caveat that the TD cost is visible and controlled. Software developers view creating shortcuts and non-scalable solutions as increasing the complexity within the code base (Yli-Huumo et al., 2014). When the code base starts to accumulate too much TD that is not fixed subsequently, the development becomes more challenging. This is because the shortcuts are not designed to work well with other parts of the code base. Complexities in the code base can decrease system quality and may result in reduced productivity when attempting to implement new solutions and features (Yli-Huumo et al., 2015). Type II and Type II.B are forms of TD usually encountered at the architectural level. TD is intentional, deliberate and prudent (Fowler, 2009). For example, in order to overcome some difficulty, market entry or partner release. As stated in the above section, this research will focus on Type II TD.

2.3 Technical Debt Management (TDM)

2.3.1 What is TDM?

As mentioned in section 1.1, Lehman (1980) adapted the second law of thermodynamics measure of entropy to software development which generated the software entropy concept. This is the phenomenon that as a software-reliant system is modified, its disorder, or entropy, tends to increase. As the software grows, so does TD. One method to measure this increasing complexity is McCabe’s (1976) Cyclomatic Complexity metric. This metric is based on the number of linearly independent paths for a source code section. It aims to quantify the complexity of a method or method in a single number (ibid). Thus, the management of this code complexity is recognised as being an important task (Agile Alliance, n.d.). TDM entails identifying the sources of extra costs in software maintenance and analysing when it is profitable to invest effort into improving a software system (Tom et al, 2013). TDM is becoming the most influential driver of software engineering advancement according to Nord (2016). Sustaining innovation pace while ensuring high software quality therefore involves establishing an explicit TDM strategy as a core software engineering practice throughout the SDLC (Nord, 2016). Striving for zero TD in an industrial context is acknowledged as being unrealistic, and maybe detrimental. Investment to reduce TD to zero would be unproductive (See Figure 5). A balance must be struck between TD an overengineering (Ebert, 2007). Establishing TDM techniques are therefore necessary to guide the TD repayment and the refactoring opportunities ranking. These are the provision of design decisions that will improve the quality of software.

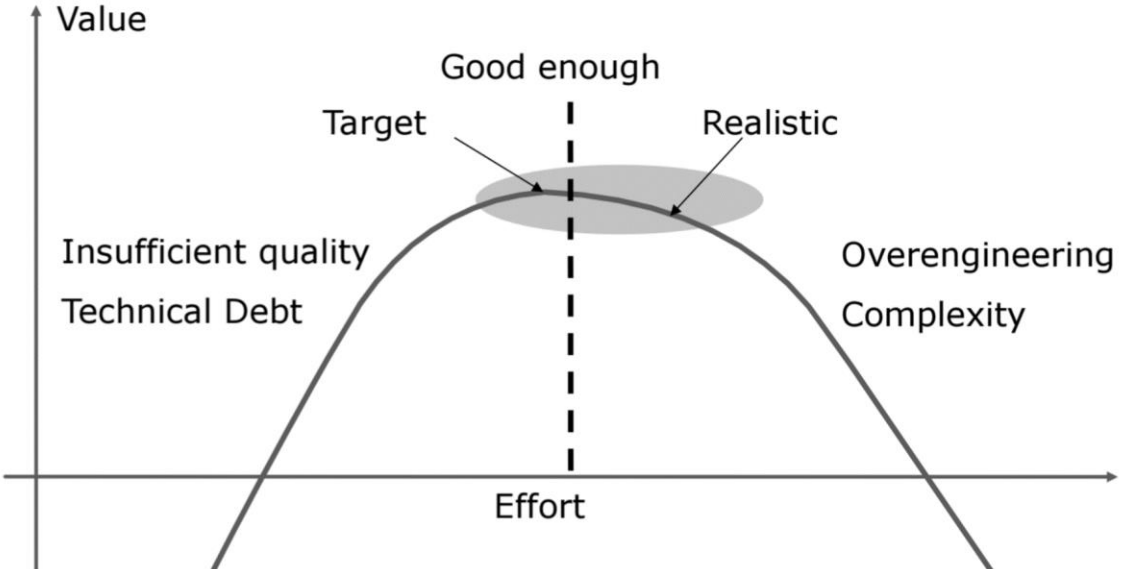


FIGURE 5 – Reduce Accidents and Control Essence (RACE)

Source: Ebert (2007)

The Managing Technical Debt workshop series, which begun in 2011, has been a major contributor to TDM research. Avgeriou et al. (2016) identified a gap in TDM research in relation to development and organisational context. They cited the need for studying what aspects of context affect how TD is best managed. TD maturity was recognised as an area that required further research. The need for a maturity model that depicts the levels at which TD could be managed was also cited (ibid). Yli-Huumo et al. (2016) subsequently adapted the capability maturity model (CMM) (Paulk et al., 1993) to TDM activities.

2.3.1.1 TDM Activities

Eight TDM activities were identified by Li et al (2015) in a systematic mapping study finding to understand the current state of TDM. This finding was later corroborated by Alves et al (2016). The TDM activities are the following: (1) **TD identification** detects TD caused by intentional or unintentional technical decisions in a software system through specific techniques, such as static code analysis; (2) **TD measurement** quantifies the benefit and known TD cost in a software system through known technique, or estimates the overall TD level in a system (3) **TD prioritization** ranks identified TD according to certain predefined rules to support deciding which TD items should be repaid first and which TD items can be tolerated until later releases (4) **TD monitoring** watches the cost changes and unresolved TD benefit overtime; (5) **TD repayment** resolves or mitigates TD in a software system by techniques such as reengineering and refactoring; (6) **TD representation / documentation** provides a way to represent and codify TD in a uniform manner addressing the particular stakeholder concerns; and (7) **TD communication** makes identified TD visible to stakeholders so that it can be discussed and further managed. The eighth activity identified, (8) **TD Prevention**, aims to prevent potential TD from being incurred. The literature relating to each TDM activity is outlined in Table 2 (ibid). Alves et al (2016) found that the various TDM activities received different levels of attention, with TD repayment, identification, and measurement receiving the most attention. TD representation/documentation received the least attention.

TABLE 2 - TDM activities and the number of related studies

TDM activity	**No. of studies**	**%**
TD repayment	59	63
TD identification	51	54
TD measurement	49	52
TD monitoring	19	20
TD prioritization	17	18
TD communication	17	18
TD prevention	9	10
TD representation/documentation	4	4

Source: (Li et al., 2015)

TDM approaches address ad-hoc cases or concrete TD types (Codabux, 2013; Letouzey, 2012; 2016). Depending on the TD type, different approaches for management may be required (Guo, 2014). Few approaches analyse TDM from a holistic perspective. Fernadez-Sanchez (2015) suggests that a model focused on a single aspect is too specific to support decision making. In summary, the current TDM understanding includes some limited techniques for managing TD. There is a requirement for more empirical evidence from realistic software development settings (Li et al. 2015; Alves et al., 2016).

2.3.2 Current Empirical TDM Research

Li et al (2015) identified that there are many studies on individual TDM activities and various components. Few empirical studies on the topic and even less that focus on holistic TDM in a real-life software development setting with (Yli-Hummo et al., 2016; Alves et al., 2016) being the only empirical studies to-date. Guo et al (2011) proposed a framework to facilitate TDM in an explicit way from a cost management perspective. The framework centres on a TD list which contains TD items. Each item represents a task that was left undone, but that runs a risk of causing future problems if not completed (Seaman et al, 2012). An item has a defined property set such as the definition, location, interest, principal, responsibility. There are two issues with this method is that in practice. 1) TD items may not be independent of each other and 2) TD instances can be described as individual items which may not always be the case (ibid). It is unknown if this approach is used in real-settings.

A study investigating how TDM decision-making was conducted by Klinger et al (2011). Although the study was conducted from a software delivery enterprise perspective, a small sample size of four architects was used. The findings concluded that decisions for TDM were largely ad-hoc and informal. A significant gap between technical and non-technical stakeholders exists in terms of TD communication and general discussion regarding TD (ibid).

There has been a larger number of research studies and development of tools to assist with TDM. Automated source code analysis tools such as SonorQube and DebtFlag have been developed to assist with one or more of the requirements for TDM. SonorQube have been applied in a few studies to both identify and measure TD in source code repositories with only one study regarding DebtFlag (Holvitie et al., 2013) which is a tool that focuses on identifying, documenting and tracking TD. Interestingly, the DebtFlag paper has three citations in total since publication, which are all self-citations by the papers author (IEEE Xplore Digital Library, 2017). The need for empirical studies investigating how TDM is conducted in organisations is evident (Li et al., 2015). Recently, the trend has been that the average TDM study rigor and relevance has increased. The TD research community is aware of this gap and is working to improve it (Fernadez-Sanchez et al, 2017).

2.3.3 Theoretical Framework for TDM

Tom (2013) rebranded McConnell's TD types and created a theoretical framework for TD. The aim was to provide a holistic TD view. The framework illustrated in Figure 6 comprises a set of TD dimensions, attributes, precedents and outcomes.

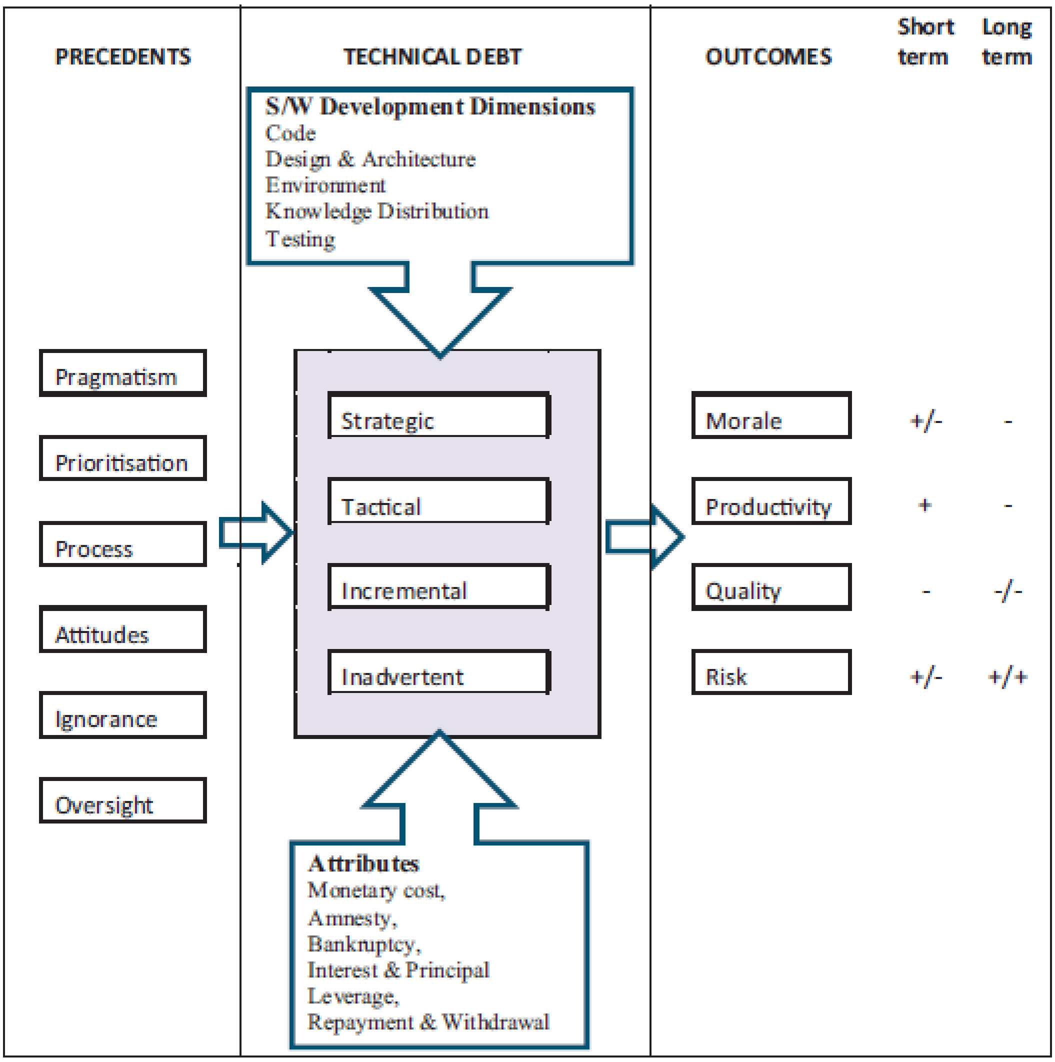


FIGURE 6 - Theoretical Framework of Technical Debt

Source: Theoretical framework of technical debt (Tom, 2013).

Tom (2013) found that TD has a negative impact on morale, productivity, quality and risk. This means that TD is not only a financial metaphor but can also be associated with real monetary cost. Software developer time and particularly outsourced development time, is expensive. When this developer time is ineffectively utilised, the development teams' velocity is slowed as they are forced to spend time fixing regressions rather that developing new features. This is illustrated as a positive feedback loop of lowered velocity, higher deadline pressure, and greater TD (Figure 7). This figure should make sense to developer and client alike (Agile Alliance, n.d.).

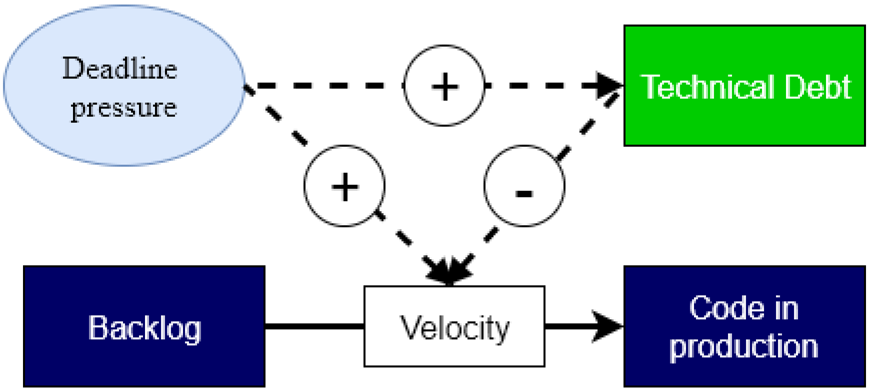


FIGURE 7 - Positive Feedback Loop

(Adapted from Agile Alliance, n.d.).

Toms' study also concluded that common sentiment exists regarding an increased focus on explicitly associating costs with TD. When a decision is made, either directly or indirectly to accrue TD, there is an increased focus on recording the debt and the costs can be attributed to those decisions. It is also conceded that there is an existing challenge in quantifying TD in any of its forms. There is no way of proving what developers intuitively know to be significant TD especially with the many confounding variables in place. A developer could have completed a software development task quicker but would not know if the code would have had more or less bugs, what those bugs would be, and if they would have high impact. One of the precedents in Figure 6 is attitudes, which will be discussed in the following section.

2.4 Attitudes Towards TDM

Organisations differ in their philosophies about TD usefulness like financial debt. Some endeavour to avoid accruing any debt, while others view debt as something to be leveraged and use it wisely (Ramakrishnan, 2013).

McConnell (2013) stated that attitudes towards debt can be religious or aesthetic. Business staff tend to be optimistic about debt whereas technical staff were more inclined to be pessimistic about debt. This sentiment based on perspective is seconded by Lim et al (2012). However, he also stated that these attitudes are often *"not conscious"* (2013, p.7). Tom et al (2013) found that the practitioners' attitudes play a significant role in contributing to TDM. Tom et al. (2013, p.1504) describes the attitudes towards TD as *"general apathy"* towards TD issues as a factor in TD accrual. These attitudes influence individual decisions to create TD and can contribute to carelessness. Carelessness can result in inadvertent TD which does not provide leverage, unlike TD that is driven by other factors (ibid).

2.4.1 Reification Method

McConnell (2007) hypothesised that the main value of the TD term is reifying a concept that is otherwise too intangible to be handled well. Einar (2015) states that TD is a symptom of an underlying lack of appropriate abstractions, which in turn stems from insufficient modelling of the problem domain. He compares TD to necessary communication that has not taken place: discussions and decisions to resolve ambiguity and make informed trade-offs have been swept under the rug. TD is the reification of this lack of code resolution. This is consistent with Conway's Law (1968) where it is posited that organisations which design systems are constrained to produce designs which are copies of the organisations communication structures.

2.4.2 Inevitability

Research conducted by Tom et al (2013) indicates that the practitioners focus should not be to repay and avoid all TD but to instead focus on its adequate management. Where there are limited resources available, some TD is inevitable and that it is possible, and necessary, to accrue TD (ibid).

2.4.3 Explaining TD

Even though the TD metaphor simplifies the concepts that the term encapsulates, it may not be sufficient for junior developers to grasp, let alone non-technical stakeholders or clients (Reddy, 2016). There is also the risk that the TD term is overextended and misused if clients lack TD understanding. Tom et al (2013) suggests that there is a need for research to focus on how to translate the TD metaphor into literal facts that are used operationally to manage TD in the software domain.

2.4.4 Pragmatism in TDM

Creating a minimum viable product in a short timeframe is an example of pragmatism given by Brazier (2007) who notes that the small companies' futures in niche markets often rely on being first to market. TD long term consequences are not visible to the customer until it is too late. The software company producing the software might not exist for much longer unless it can deliver to the customer in the short term (ibid). Conversely, Falessi et al (2013) identified time-to-market as an important consideration. Fernadez-Sanchez et al (2015) discusses the paradox of how time-to-market is rarely considered by current TDM tools and methods although it is the most referenced TD cause. In summary, a pragmatic trade-off between software release characteristics and TD must be made to effectively manage TD (Ramasubbu et al., 2013).

2.4.5 TDM is increasing in importance

There is support for TD attributes such as amnesty and leverage. When TD is managed, and controlled, positive outcomes occur. TD accrual allows accomplishment of something that is otherwise unattainable – a fundamental TD principle (Tom et al, 2013). Ernst et al (2015) identified TDM as being a reactive activity as it needs to cause significant pain on multiple fronts before it is addressed and stresses the ongoing explicit TDM importance. Ramasubbu et al. (2015) recommend introducing a TD policy which is based both on the organisation business context and the and on the technological environment in which the firm operates.

2.5 Organisational Factors and TDM

Ramasubbu et al (2015) also highlight influential factors of an operating business environment on a software team’s TD policy as an area in which the empirical evidence is suggestive, but significant questions remain unanswered. Questions such as why are more technically capable software teams more debt adverse and the quality of business reasoning for accruing TD. TD policy must provide a framework that supports new behaviours leading to justifying TD accrual (Brenner, 2017).

Organisational dimensions impact on IS success continue to be research from various perspectives (Miller and Doyle, 1987, Grover, 1993, King et al. 1992; Tallon et al, 2000; Ang et al, 2001). Snipes et al (2012) describe the decision to fix or defer TD items as dependent on the amount of TD that has been accumulated as unintentional TD. They found that the cost factors that influence the TD decision are: severity, existence of a workaround, fix urgency required by a customer, effort to implement the fix, proposed fix risk, testing scope required. These factors are listed in decreasing order of importance with severity being the most influential factor. Other factor categories such as organisational factors need to be empirically evaluated from a TDM perspective to assess their validity and to gain insights on how they can be improved. Research use different terminology to describe organisational dimensions such as contexts, variables and factors. Further examination to determine how practitioners consider TDM decisions from a software services provider perspective is also warranted. This study attempts to assess the organisational factors that influence decision-making for TDM from a practitioner's standpoint.

2.5.1 Management Commitment

Ward Cunningham (1992) coined the TD term to convey the extent of deterioration in the software structure in the form of metaphors so that managers from non-technical backgrounds can relate to the problem (Sharma et al., 2015). Top management support is theorised as the involvement and top-level management participation in the organisations IT activities (Jarvenpa et al., 1991). Management commitment to TD repayment is important (Ramakrishnan, 2013).

2.5.2 Organisational Culture

The American Heritage Dictionary (2015) defines "culture" as *"the totality of socially transmitted behaviour patterns, arts, beliefs, institutions, and all other products of human work and thought characteristic of a community or population."* Bower (2003) describes it simply as *"the way we do things around here".* Organisational culture is one of the important determinants of IS success that IS not deeply discussed by researchers. Wiegers (1996) discusses the essential components of both healthy and unhealthy software engineering cultures via individual, management and organisational behaviours. There is a void in empirical evidence as to how organisational culture influences TDM.

2.5.3 Management IT Knowledge

Data collected by Lim et al (2012) revealed that TD often resulted from decisions made by high-level technical and non-technical managers. Software development knowledge influences decisions to incur TD. Teams with technical leadership who had *"just get stuff done"* attitudes were more likely to incur debt than those with team leaders who focused on the software purpose (ibid).

2.5.4 Developer Mindset

Yli-Hummo et al (2016) suggested developer mindset is an influential factor for how and when TDM is conducted. Attitudes towards TDM can sometimes be negative as developers and managers focus is to deliver new features which leads to quick solution usage. By all software development team members contributing to TDM it can facilitate the TDM activities to support each other such as documentation feeding into prioritisation. Additionally, the risk appetite at an individual, team or organisation level can influence decisions relating to TD in several ways. Taking the decision to accrue even small amounts of TD can have side-effects. 'Broken Windows' is a theory about crime developed by Wilson et al, (1982). It states that when low level crimes such as vandalism are ignored, more serious crimes occur. With software development, a similar pattern can emerge (Lash, 2014). TD can impact on risk both positively or negatively in the short-term. Therefore, the appetite for risk could influence the TD creation and impact on project risk (Tom et al, 2013).

2.5.5 Trust and Client Relationship

The literature categories trust in software outsourcing relationships into two parts 1) achieving trust initially and 2) maintaining trust (Oza et al., 2006; Babar et al., 2007). A much-cited paper studying the role of trust in outsourced IS developed projects found that projects proceed through virtuous and vicious cycles involving trust, structure and performance (Sabherwal, 1999). Heintz (2016) states that while unmanaged TD is a problem, it can also be a symptom of an imbalance in the enterprise. He argues that technical approaches to addressing the TD problem are unlikely to achieve a long-term solution. Cultural transformations are difficult, in part, because even after we successfully identify what must change, moving an organisation as one toward that goal can be even more challenging. In this instance, because of a "virtuous cycle," the effort to manage TD can provide enterprise leaders the leverage they need. If the enterprise can hold accountable the organisational elements on whose behalf IT now issues TD, political power within the enterprise can be rebalanced, which would then facilitate the TD control. IT can then assume the strategic role so necessary for enterprise success (ibid)

Other research has focused on trust as a dynamic in IS outsourcing relationships (Heiskanen et al., 2008). An empirical study by Babar et al (2007) identified that cultural understanding, communication strategies, contract conformance, and timely delivery are vital factors in maintaining gained client trust. Ova et al (2006) found that the critical factors identified for maintaining trust in an established outsourcing relationship include transparency, demonstrability, honesty, processes followed and commitment. The findings also suggest that trust is fragile in outsourcing relationships. The relationship between product value and intrinsic software quality is important. Ernst (2012) argues in a position paper that if the product is not delivering value to the client, then high intrinsic quality (i.e. low TD) is irrelevant. If intrinsic quality is low (significant amount of debt) then the chances of delivering valuable software in the future is likewise low.

2.6 TDM Techniques

Literature investigating TDM strategies, methods and techniques was non-existent in 2010 with no papers published in that year. 2011 saw a few papers being published proposing the use of cost-benefit analysis and the portfolio approach for TDM including Gat and Heinzs (2011) TD Assessment which uses a combination of crude techniques to formulate a plan for TD reduction. It was not until 2012, when 16 papers were published exploring TDM techniques, that the topic became more popular (Alves. et al, 2016). TDM approaches should be refined significantly, while more empirical studies are needed to show evidence of different TDM technique usage. Many TDM approaches are only mentioned in the related primary studies. The authors have not discussed or investigated how to use those approaches in actual cases. Ernst et al (2015) found that there are few explicit systematic TDM practices while some TDM occurs in existing processes more often. Thus, these approaches may not be practical enough to be used in real projects (Li et al, 2015). The current literature for various TDM techniques discussed in the next section.

There is some overlap between TDM activities and the techniques used to carry out these activities. Case studies can also be used to explore existing TDM notions of value in practice (Avgeriou et al., 2016). Exploratory case studies could also be used to identify “grassroots” approaches to TDM (ibid, p.131). For example, environments without an explicit strategy to manage TD, but due to necessity, techniques have developed to deal with the most important TDM issues (ibid.). Alves (2016) summarised the various techniques and approaches for each TDM activity across fifty-eight studies which mention, propose, or use one or more TDM approaches, TD types, and publication years. The number of studies on approaches for different TDM activities vary. The approaches for TD identification, measurement and repayment were mentioned, used, or proposed the most frequently in the selected studies. Approaches for TD representation / documentation received the least attention. The most cited management strategies are the portfolio approach and cost-benefit analysis. These approaches have not been evaluated in a real-setting and the number of papers identified is small at ten and eleven respectively (Alves, 2016).

2.6.1 Automated Source Code Analysis

Source code is any collection of computer instructions written using a human-readable programming language (The Linux Information Project, 2004). Source code analysis is a specific reverse engineering technique that extracts information from a program from its source code (Tonella et al., 2007). Automated Static Analysis (ASA) tools scan source code for violations of recommended programming practices which may decrease software program quality. These violations should be removed through refactoring to avoid future problems (Boolgerd et al., 2009). The most referenced tool for conducting ASA are SonorQube and DebtFlag (See 2.3.2), which are defined as continuous inspection engines to evaluate and manage TD. SQUALE is a quality and analysis model used in ASA. Letouzey developed the method for evaluating (2012; 2012a) for managing TD. Fontana et al (2016) and Guaman et al (2017), conducted research on the use of SQALE and tools for analysis and code TD identification through static analysis. These, however, were not empirical studies. The SQALE method has been used to perform software source code assessments of various sizes in different application domains and programming languages (ibid).

2.6.2 TD Items and TD List

As mentioned in Section 2.3.2, The technique of communicating TD using TD items has been proposed by several researchers. A TD item is defined as a unit of TD in a software system (Guo et al., 2011; 2011a; Holvitie et al., 2013, Zazworka et al., 2013). A TD item example is a "God class*",* which is an object that knows too much or does too much (Riel, 1996, p.80). The literature varies in what information should be captured in a TD item. ID, location, author, type and description are TD item data components that have been identified by a least four studies. Date/time, principal and interest amount have been identified as data components in three studies. Other components are found in two studies or less, they include interest probability, context and correlation. TD in a software system comprises of a number of TD items. A TD items group can then be presented in the form of a TD List. A TD list keeps all identified TD items and makes them visible to stakeholders (Guo et al., 2011, Holvitie et al., 2013). The TD list can be used in the TD prioritisation activity.

2.6.3 Cost-Benefit Analysis

This technique evaluates whether the expected interest is high enough to justify the debt payment. TD items may be prioritised for repayment based on the cost versus benefit (Figure 9). Snipes et al (2012) proposed a cost-benefit analysis model based on cost factors with the below variables.

- P – Debt Principal
- I_W – Portion of interest for defining a workaround

- I_c - Portion of interest for customer support for the workaround
- I_p – Patch cost
- I_{pr} – Probability of customer requesting a patch for the defect
- I_{fr} – Probability the deferred defect is eventually fixed (principal repaid)

The cost-benefit ratio *p* is given the following equation (ibid) outlined in Figure 8.

$$\rho = \frac{P}{I_w + I_c + I_p * I_{pr} + P * I_{fr}}$$

FIGURE 8 – Cost-Benefit Equation

Some special situations must be considered in cost-benefit analysis. When a system is retired, its TD is removed (McConnell, 2007; Tom et al., 2013) and results in Tom et al (2013) labels a debt amnesty. Another being the constraint that an organisation does not have adequate resources to fix all identified TD items (Letouzey et al., 2012a).

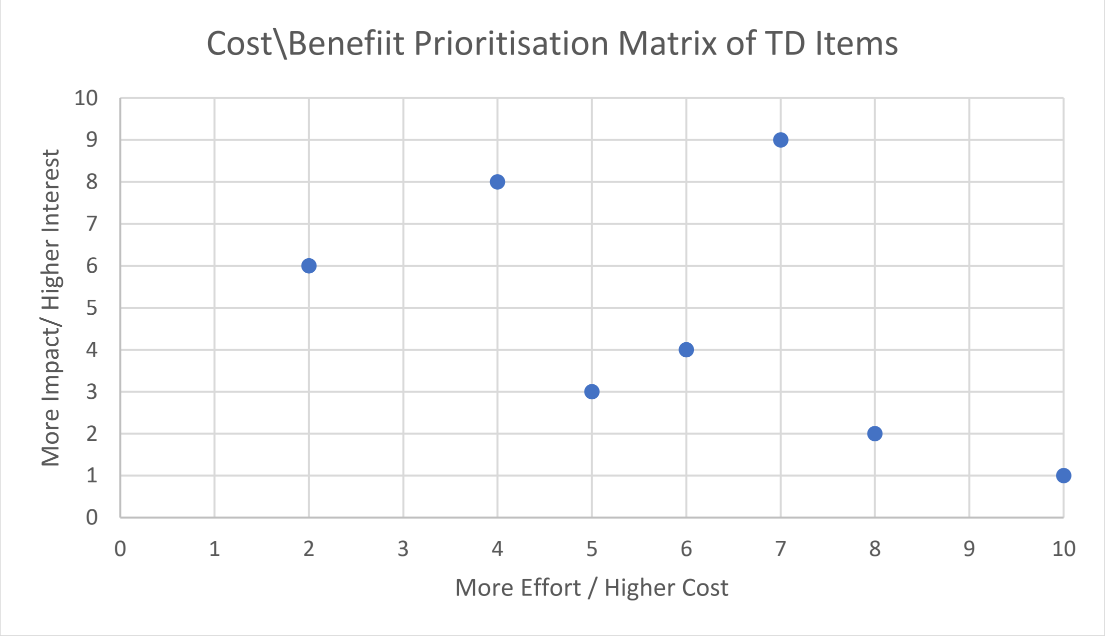
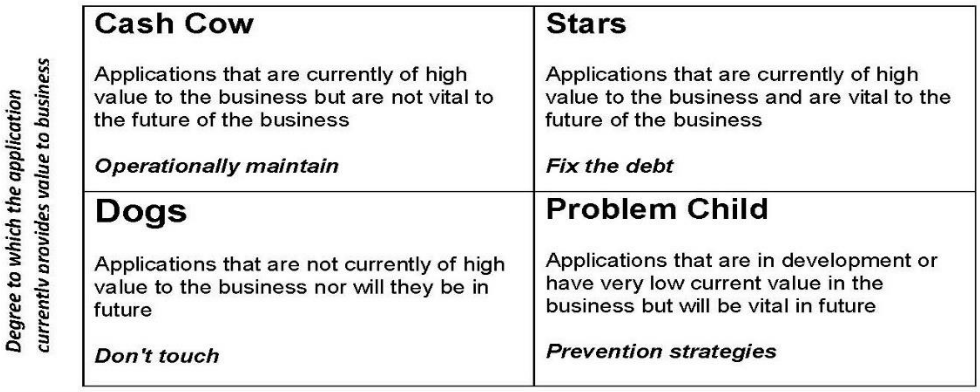


FIGURE 9 – Cost-benefit prioritisation matrix

2.6.4 Portfolio Approach

A portfolio management approach for TD prioritisation has been Guo et al (2011) which is an adaption of the investment portfolio model used in the finance domain. Garnett (2013) suggests TD portfolio prioritization as a technique. Based on the degree of business value that the application currently provides against the degree of business value that the application fulfils for the future business needs, the TD Prioritization Quadrant was proposed (Figure 10). The quadrant appears to be an adaption of the Boston box and McFarlan's Application Matrix (McFarlan, 1984; Ward, 1987, Kaplan Financial 2012). Fontana (2015) proposed a Code Smell Intensity Index for TD prioritisation that have been identified.


Degree to which the application currently provides value to business

Cash Cow Applications that are currently of high value to the business but are not vital to the future of the business ***Operationally maintain***	**Stars** Applications that are currently of high value to the business and are vital to the future of the business ***Fix the debt***
Dogs Applications that are not currently of high value to the business nor will they be in future ***Don't touch***	**Problem Child** Applications that are in development or have very low current value in the business but will be vital in future ***Prevention strategies***

FIGURE 10 – TD Prioritization Quadrant

2.6.5 TD Register and Backlog

As seen in Table 4.1, the TDM activity of TD documentation\representation has not been studied comprehensively. Where it has been studied, TD is being documented and logged via TD backlogs which usually make up a TD list for review. Documentation can be valued differently by software practitioners. Developers may only document issues that they consider important (Lethbridge et al, 2003). Yli-Hummo (2016) found different development teams documented TD in several different ways. TD items could be documented in a general product backlog or have their own dedicated backlog or not be documented at all and remain tacit knowledge amongst the development team. The lack of standard TD documentation practices brings its own issues in terms of knowledge management implications and development team velocity (See section 2.7.2). It is important that TD Documentation is included in the overall TDM Strategy (Klinger et al., 2011). The TD documentation activity has ramifications for other TDM activities also. This is evident in creating a systematic repayment strategy where the documentation and storing of all TD issues is required (Lim et al., 2012). Documentation in software development can also improve understanding and communication between stakeholders which could be applied to TD documentation and TD communication (Das et al., 2007; Forward et al., 2002).

2.6.6 Human Estimation and Expert Opinion

A systematic mapping study by Fernadez-Sanchez (2015) found that most authors suggest the need to use expert knowledge to add TD information that cannot be estimated in another way. Cai et al (2014) aim to automate refactoring recommendations to software architects which can then be analysed and selected based on software architects' opinion.

2.7 TDM Challenges

TDM has been described as challenging to implement in practice (Yli-Huumo et al., 2016). The reasons for this are far ranging. For example, managers and developers struggle to identify and estimate the various TD types present in a software system. Furthermore, predicting how the TD will change and what impact that will have in the future is difficult (Li et al., 2015). Power (2015) found seven current challenges for TDM: (1) Agreeing what TD is; (2) quantifying TD; (3) Visualising TD; (4) Tracking TD over time; (5) Non-repayment of TD for an extended period; (6) Mapping TD as defect root cause; and (7) Understanding the delay cost. Yli-Huumo et al (2016) found that a lack of tools is a major challenge. Ernst et al (2015) found that although a wide range of tools were used, tool usage per percentage of answers was small. The study conducted by Ernst et al (2015) used the Likert scale which could be argued is one dimensional and assume that respondents can accurately map their responses to a question into the options allowed (Jamieson, 2004).

2.7.1 Lack of TDM Tools

Yli-Huumo et al (2016) found that TDM is frequently conducted as a human activity. He identified that most TDM activities were done with rough estimations and based on hunches. As well as the excess time consumed by not using automated tools, without tools and models based on data, there is the risk that the decisions made are not always the correct ones. However, it is conceded that the development of a complete support tool for TDM brings practical challenges (Falessi, 2013). Alves (2016) identified 29 tools for managing TD in some capacity. Four were specifically for TDM whilst the rest were being leveraged from other areas and process for software development. Furthermore, the identified TDM tools support two of the TDM activities. A tool not mentioned by Alves et al (2016) is 'TD-Manager' developed by Foganholi et al (2015) which centres around an integrated TD catalogue for producing a list TD item list.

2.7.2 Knowledge Management

The absence of well-written technical documentation on software systems indicating a high amount of documentation debt can be challenging for software development teams (Slinker, 2008). A large amount of tacit knowledge retained by individuals in a software development team is a risk for stakeholders as TDM relies on TD information being documented for monitoring and communication activities (Tom et al., 2013).

2.7.3 TDM Investment Justification

Practitioners may find difficulty in justifying the real need for TDM and its benefits as TDM can be time-consuming and can create additional work in the development processes (Yli-Huumo et al., 2016). More research on TDM is required to produce evidence on TDM benefits and drawbacks. This will give confidence in development teams to allocate appropriate time and resources to TDM.

2.7.4 Difficulties in business and economic value transformation

TDM outcomes should be reviewed in terms of its economic consequences (Falessi et al., 2013). The challenge is that decisions have different consequences depending on when it is made. For example, refactoring source code cost for the current release is different from the cost of refactoring for a future release (ibid).

2.8 Conclusion

This chapter has examined and reviewed the existing literature in relation to the research topic. This chapter has discussed the current definition and TD understanding, what attitudes exist towards TD and its management, the factors that influence TDM, techniques for TDM and the TDM challenges that currently exist. The next chapter describes the research methodology used for this study and the rationale for its selection.

Chapter 3: Research Methodology

3.1 Introduction

This chapter describes the research method that frames the execution of this investigation. It provides justification as to why this is deemed to be the most appropriate research methodology to be followed. The chapter contains clarification on the data type being used, how data was collected and details the steps followed to conduct data analysis. A detailed explanation of how the research was conducted and the way the data were collected is provided. In addition, the main strengths and weaknesses associated with the chosen approach are detailed. Research can be defined as "an activity that involves finding out in a more or less systematic way, things you did not know" (Walliman, 2011 p. 7). Methodology can be described as "the philosophical framework within which the research is conducted or the foundation upon which the research is based" (Brown, 2006). The chapter concludes with a summary of the research methodology chosen.

3.2 Research Purpose

The research aims to explore how TD is defined and managed in an industrial setting by attempting to answer the questions outlined in section 3.3. TD impacts on all aspects of IT project development and has received an increased focus in recent years. This study aims to examine both the understanding and TD attitudes, organisational factors that influence TDM and the techniques employed by software practitioners for TDM. Is there a common TD understanding? Is one organisational factor viewed as more important than others? What techniques are used to conduct TDM? In addition, challenges that obstruct effective TDM are also studied.

3.3 Research Questions

The research questions (RQ) below will be addressed through the methodology outlined in Section 3.4 based on the collected data interpretation which is detailed in Section 3.5.

RQ1. How do senior software practitioners view the TD concept and its management?
RQ2. What organisational factors influence the management of Type II TD?
RQ3. What methods and techniques are used for explicit TDM activities?
RQ4. What challenges exist in the management of TD?

3.4 Research Methodologies

Research methodology has been described as the overall approach to a problem which could be put into practice in a research process, from the theoretical underpinning to the collection and analysis of data (Remenyi et al., 2003). Collis and Hussey (2014) simply defined methodology as the overall approach to the entire process of the research study. Based on these definitions, research methodology focuses on the problems that are the subject of investigation in the research study and varies depending on the subjects being studied. Saunders et al (2007) illustrates the research methodology components in the form of a “Research Onion” (Figure 11). This onion depicts the research methodology stages that need to be developed as layers. These layers are Research philosophy, approach, strategy, choice, time horizon, and techniques. Each layer describes a more detailed research process stage. The below illustration provides a framework for which a research methodology can be designed. The framework success is primarily based on its adaptability for most research methodologies and contexts (Bryman, 2012). Other definitions and layer classifications exist but those identified by Saunders et al (2009) are preferred for the purposes of this research as it provides an unambiguous overall framework for the complete research process.

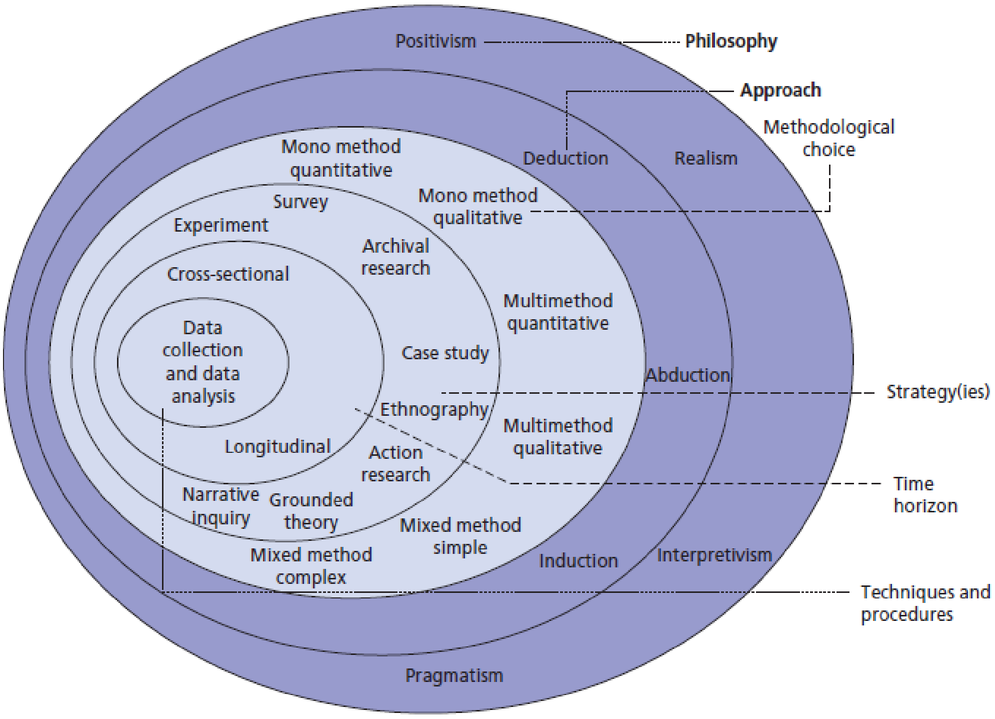


Figure 11– The Research ‘Onion’

(Saunders at al., 2007)

3.4.1 Philosophies

A research philosophy refers to the set of beliefs concerning the nature of the reality being investigated (Bryman, 2012). Yin (2003) describes a research philosophy as how the data within the research study is collected, analysed, interpreted and presented. It is the underlying definition of the nature of knowledge. Assumptions created by a research philosophy produce assumptions about the researcher's world view and the justification for how the research will be approached (Flick, 2011; Saunders, 2007). Depending on the research goals and the method used to achieve these goals, research philosophies can vary (Goddard et al., 2004). The research philosophy choice should be defined by the knowledge type being researched (May, 2011). Comprehension of the available research philosophies and the philosophy used for this study, can therefore assist in explaining the assumptions present in the research process and how this fits with the chosen research methodology. No philosophy is inherently better than another but instead is dependent on researchers' preferences (Podsakoff et al., 2012). The philosophy serves as the justification for the research methodology. The research methodology should be informed by the nature of the phenomena being investigated. The below paragraphs discuss the four most common philosophies that are used amongst IS researchers along with their applicability to this research study.

3..4.1.1 Positivism

The positivist philosophy believes that reality is stable and can be observed and described from an objective perspective (Levin, 1988), i.e. without interfering with the phenomena under investigation. The philosophy has had a successful association with the physical and natural sciences. Positivism is commonly related with observations and experiments to collect numeric data (Easter-by-Smith et al, 2006). Researchers are interested in collecting general information and data from a large sample instead of focusing on research details. There has been much discourse on the positivist philosophy suitability for social science research (Hirschheim, 1985) with many researchers preferring a more pluralistic attitude towards IS research methodologies (Kuhn, 1970; Bjørn-Andersen, 1985; Remenyi et al., 2003). This study is focused on IS and therefore, people and technology interaction. The social sciences as opposed to the physical sciences. For this reason, positivism was discounted as a possible philosophy to be applied for this research, *3..4.1.2 Interpretivism*

The interpretivist perspective is appropriate for business and management research such as organisational behaviour. Business situations are complex whilst also being unique to a set of circumstances and group of individuals. Although this unique perspective raises the research findings generalisability issue which aims to capture the complexity of social situations. However, interpretivism does not view generalisability as insignificant due to the evolving nature of business organisations. Present organisational circumstances may not be applicable in the near future. Due to this aim of this study to explore subjective meanings expressed by the TD phenomenon and the way that it is conducted, interpretivism was chosen for this study (Saunders et al., 2007).

3..4.1.3 Realism

The realism philosophy relies on the idea that there is a reality which may not correspond to the human mind, independent of an individuals' worldly perceptions (Saunders et al, 2007; Bryman et al, 2015). Realism shares many similarities with Positivism whilst differing on the idea that humans are influenced by social factors and therefore cannot be treated the same way as natural sciences. Realism can be further broken down into direct and critical realism which are not relevant to this study.

3..4.1.4 Pragmatism

Core to the pragmatic philosophy is the research question itself and the therefore the research may apply multiple approaches to answer the question (Creswell, 2003). Another key pragmatism characteristic is the assumption that no other traditional philosophy meets the research requirements and these other philosophies may constrain the research process. Pragmatists recognise that research methods have limitations and often use a mixed method approach and work with both quantitative and qualitative data collection, using complementary analysis techniques (Saunders et al. 2007). A pragmatist might begin to collect data by conducting face-to-face interviews and use the findings to inform the questions of a survey.

3.4.2 Research Approach

Mingers (2001) states there are three available approaches to research; deductive, inductive and abductive. Deduction is when instances are deduced to follow from general laws or assumed premises. Induction, is an approach where general laws are induced from certain instances. Abduction, is where evidence is used to alter the probabilities associated with a hypothesis and therefore confirm or disprove it.

3.4.2.1 Deduction

Cooper et al (2013) describe deduction as the process by which we test whether the hypothesis can explain the fact (Figure 12). The deductive approach involves theory development. The theory is then tested through specifically designed research strategies. Deductive reasoning starts general and finishes more specific (top-down approach). Definitions, interpretations and the initial statement originations often limit the usefulness of deductive reasoning in research (ibid).

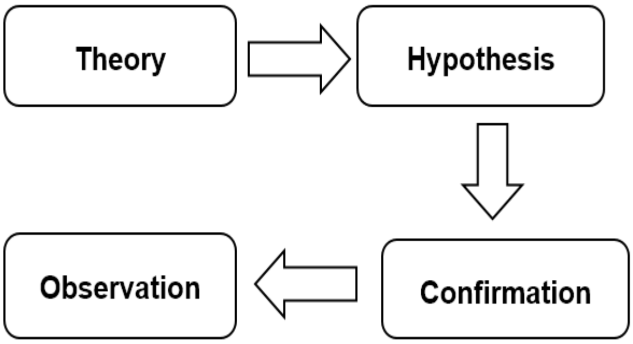


FIGURE 12 – Deductive Process

3.4.2.2 Induction

Cooper et al (2013) describe induction as asking “Why is this?” when observing a fact. A tentative explanation or hypothesis is proposed to answer this question. The hypothesis goal is to explain the event of condition which prompted the question. The inductive research process is illustrated in Figure 13. The inductive approach begins with specific observations and progresses to wider generalisations and theories (bottom-up approach). Due to the exploratory nature of this research (Robson, 2002), data collection, organisation and analysis will be guided by an inductive approach. The collection, analysis and continual data re-examination process will determine the research findings.

The inductive approach can provide valuable insights but has limitations as it is based on incomplete observations. Therefore, it is unknown if a TDM technique has not been experienced because it does not exist or because it does exist but it has not been witnessed.

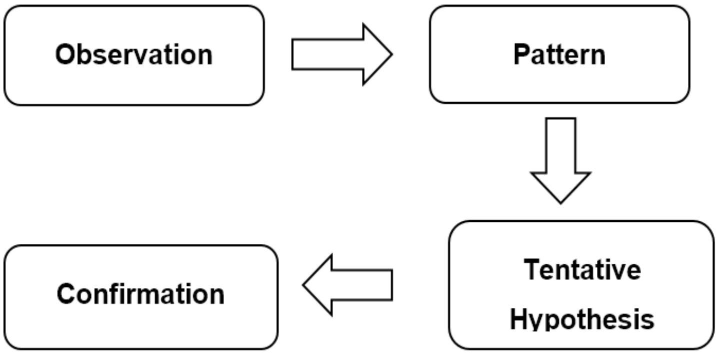


FIGURE 13 - Inductive Process

3.4.3 Methodological Choice

In general, there are three research methods. They are quantitative, qualitative, and the mixed-methods approach (Creswell, 2003). The qualitative research method is framed in terms of using words while the quantitative research method is framed using numbers. Hence, the mixed research method incorporates elements of both the qualitative and the quantitative methods. Quantitative and qualitative research represent different strategies which are distinct in terms of the role of theory, epistemological issues and ontological concerns (Bryman, 2015). The debate regarding the respective benefits of quantitative versus qualitative research has been the focus of a much dispute over the past several decades (e.g., Bryman, 1984; Hammersley, 1992; Tashakkori et al., 1998; Kelle, 2001; Creswell, 2003; Johnson et al., 2004)

Case studies do not imply evidence usage and they can be done using three approaches mentioned previously (Eisenhardt, 1989; Yin, 1989). While quantitative data often appears in case studies, qualitative data usually predominates (Patton et al., 2003). For the above reasons, a qualitative research method will be used in this study.

3.4.4 Research Strategy

Saunders et al. (2012) outline several strategies that are available when conducting business and management research. The following strategies were considered regarding its suitability in addressing the research questions.

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

Empirical and theoretical research are the two main academic research categories (Remenyi et al., 2003). Theoretical research involves finding a result predicted by a theory, while empirical research focuses on data gathering and analysis. This research paper will apply an empirical research method.

The chosen research strategy used for this research is a case study. A case study is 'an empirical inquiry that investigates a contemporary phenomenon with its real-life context, especially when the boundaries between phenomenon and context are not clear' and it 'relies on multiple sources of evidence' (Yin., 2003, p. 13). Case study research investigates pre-defined phenomena but does not involve explicit control or variables manipulation. The focus is on in-depth phenomenon understanding and its context (Cavaye, 1996). A case study is used to contribute to individual, group and organisational knowledge and has been a common research strategy in social sciences. Case studies as a research methodology are becoming popular in software engineering (Runeson et al., 2009). As software development is carried out by individuals, groups and organisations within a social context it is a multidisciplinary area where the case study strategy is relevant. Case studies typically combine data collection techniques such as interviews, observation, questionnaires, and document analysis (Yin, 2003).

The case study is ideal for exploring new processes or behaviours or ones that are little understood (Hartley, 1994) such as the subject of TD and TDM. The major difference between the case study and other qualitative designs such as grounded theory and ethnography (Glaser et al.,1967; Corbin et al., 2008) is that the case study is applicable to the use of theory or conceptual categories that guide the research and data analysis. In contrast, grounded theory or ethnography presupposes that theoretical perspectives are grounded in and emerge from first hand data.

As this study's objective is to study TD and its management in-depth in a real software development setting, a single-case study approach was selected. Case studies can be categorised as holistic case studies or embedded cases studies (Yin, 2003) (Figure 14). This research studies the case as a whole, making it a holistic case study.

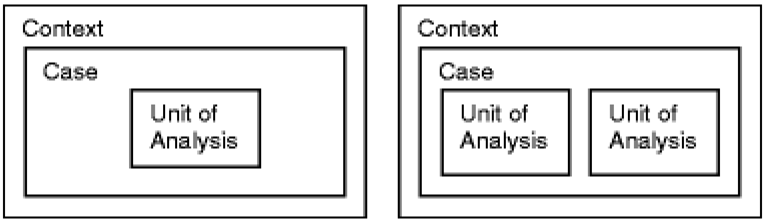


FIGURE 14 - Holistic Case Study (Left) and Embedded Case Study (Right)

(Benbasat et al., 1987).

The rationale for this is that this study aims to study software practitioners from one organisation together. An embedded case study would have studied multiple units of analysis within a single case study. There are several case studies types; explanatory, exploratory, or descriptive. The case study type selected for this research was guided by the overall study purpose. As this research aims to explore what is currently happening and seek insights into a case, the exploratory type was selected. Previous similar influential studies such as (Tom et al., 2013; Yli-Huumo et al., 2016) have also followed this case study type.

There is a severe danger that without a theoretical framework, the researcher may provide description without meaning (Hartley, 1994). Gummesson (1988) says that a lack of preunderstanding will cause the researcher to spend considerable time gathering basic information. This preunderstanding may arise from general knowledge such as theories, models, and concepts or from specific knowledge of institutional conditions and social patterns. The key is not to require researchers to have split but dual personalities: 'Those who are able to balance on a razor's edge using their pre-understanding without being its slave' (ibid., p. 58). Yin (2003) and Stake (1995) use different terms to describe a variety of case studies. Yin (2003) categorizes case studies as explanatory, exploratory, or descriptive and differentiates between single, holistic case studies and multiple-case studies. Stake (1995) identifies case studies as intrinsic, instrumental, or collective. The distinction between intrinsic and instrumental case studies is the degree to which the focus is on the unique or the generalizable aspects of the researched case (Hartley, 2004). A case study cannot be defined through its research methods, but in terms of its theoretical orientation and interest in individual cases (Stake, 1995; Hartley, 2004).

Yin (2003) suggest that case study design should be considered when:

- The focus of the study is to answer ‘how and ‘why’ questions;
- You cannot manipulate the behaviour of those involved in the study;
- You want to cover contextual conditions because you believe they are relevant to the phenomenon under study; or
- The boundaries are not clear between the phenomenon and context.

The single case study method was chosen for this study to understand TD phenomenon aspects in-depth. Although this eventual understanding will encompass important contextual conditions because they are pertinent to the phenomenon being studied (Yin, 2009). The case study strategy follows an in-depth look at one organisation, community or person (Bryman, 2004). This study endeavours to capture the local situation in greater detail and with respect to more variables than is possible with surveys. In this instance, the case is an IT consultancy and software services company based in Dublin. Case study research is associated with description and with theory development. As such, TDM exploration is an area where existing knowledge is limited and is well suited to this strategy (Cavaye, 1996).

Another reason for the case study strategy selection was that the research topic was TDM but the case could not be considered without the context, the consultancy business format, and more specifically the software development and support settings. It was in this setting where the TDM activities were developed and conducted. It would have not been feasible for this research to elicit accurate software practitioners’ accounts of conducting TDM without considering the context where it occurred.

3.4.5 Time Horizons

The Time Horizon is the time framework within which the project is intended for completion (Saunders et al., 2007; Bryman, 2012). Two time horizons type are specified within the research onion (Figure 11), the cross sectional and the longitudinal (Bryman, 2012).

3.4.5.1 Cross-sectional

Cross-sectional studies are a snapshot where data is collected at a certain point in time (Flick, 2011). This time-horizon is used when the investigation is concerned with certain phenomenon research at a specific time. As this research is concerned with how TDM is conducted for a case organisation during the summer of 2017, this time horizon was chosen.

3.4.5.2 Longitudinal

A longitudinal time horizon refers to the repeated observations of the same variables over an extended period. This time horizon is used where change over time is being examined (Goddard, 2004). Thus, the researcher gains some control over the variables being investigated. Due to research time constraints, a longitudinal time horizon was not selected for this study. However, an interesting opportunity for future research would be to investigate how TDM maturity is evolving over an extended time frame.

3.5 Data Collection

Two data types that can be collected in research are primary and secondary data. As this research is using a case study research strategy, appropriate data collection methods were considered. Yin (2003, pp. 83-96) states that there are six possible data collection methods for case studies. They are archival records, documents, direct observation, interviews, physical artefacts and participant-observation.

The steps that were taken for conduction this case study were identified by Yin (2003) and are as follows:

1. Determine and define the research questions.
2. Select the cases and determine data gathering and analysis techniques.
3. Prepare to collect the data.
4. Collect data in the field.
5. Evaluate and analyse the data.
6. Prepare the report.

To select the data collection technique, several considerations were taken into account such as population, sampling, questions, content, bias and administration. This study uses a combination of semi-structured interviews and document data sources. Saunders (2012) outlined three interview types; (1) Structured or Intensive which is an interview with a standardised question set; (2) Unstructured is an informal conversation in which participants speak freely and; (3) Semi-Structured interview which contains a combination of open-ended and specific questions, with the latter used to guide the conversations. The use of semi-structured interviews can allow for unexpected information types. Further evidence was gathered using the documentation and textual analysis. These research instruments are described in more detail in Section 3.5.3.

Depending on the methodological approach used both the data collected and the analysis technique applied can vary (Bryman, 2012). The process used at the data collection and analysis stage has a significant contribution to the reliability and validity of this study, outlined in Section 3.8 (Saunders et al., 2007).

3.5.1 Population

A sample represents a segment of a larger population (Bryman, 2012). This case study uses a single population of four hundred software practitioners based in Ireland and employed by the studied organisation to provide software services to its clients. The population makes no differentiation between sub-contracted and permanent staff and excludes staff who deal primarily with computer hardware as the focus of this study relates to software.

3.5.2 Sampling

Sampling is the process of selecting units from the population. Studying the sample may allow for results generalisation to be applied to the larger population from which they were selected (Trochim*,* 2006*).* For quantitative research, the samples size and how it is selected can be used to verify research results reliability. With qualitative research, sample characteristics are also important and smaller samples are typically used. This is due to phenomena only needing to be noticed once to be registered on the analytical map (Ritchie et al., 2003). There is a point of decreasing return when increasing the sample size no longer contributes to new evidence. Guest et al (2006) suggest that data saturation can occur within the first twelve interviews and subsequent interviews are unlikely to uncover many more new phenomena.

For this study, 15 software practitioners were approached to participate in the study. Non-probability sampling is more frequently used when adopting a case study strategy (Saunders et al., 2012) and was therefore selected for this study. Within this sampling approach, the purposeful sampling technique was chosen to select cases that will best enable to answer the research question (Neuman, 2000; Creswell, 2003*)* but availability and willingness to participate were also factors *(*Spradley, 1979; Bernard, 2002). Purposeful sampling involves identifying and selecting individuals that are knowledgeable or experienced in the research area (Creswell, 2003). This research requires the participation of individuals directly involved with conducting TDM activities. The sample for this research is limited and therefore random sampling was not an option.

Semi-structured Interviews: n= 14
Documents: n= 34

3.5.3 Research Instruments

3.5.3.1 Semi-structured Interviews

Semi-structured interviews were selected for data collection for this study as this is the most popular primary data collection method with exploratory case studies (Brown, 2006). A semi-structured interview is a qualitative interview that is defined by a pre-set question guide that aims to provide in-depth findings through informal discussions with participants (Collis and Hussey, 2003). Semi-structured interviews allow for probing responses further and to identify deeper patterns and meanings. Additionally, the data are expected to provide broad results. The interview method will allow for further insight into potential considerations methods and techniques, which would be difficult using surveys or other quantitative analysis methods. Silverman (2007) points out that the way the interviewer interacts with the interviewed person impacts the collected data. Therefore, special precautions were taken during the interview to avoid influencing interviewees, they were:

- Conducting overt encouragement to not imply approval or disapproval of what the participant said as that could bias the research findings.
- Asking the same questions in different ways to compare the responses so to avoid the participant giving answers that they think the researcher would prefer to hear.

Interview questions were developed and based on questions from previous research case studies conducted by Yli-Huumo et al (2016) and Snipes et al (2012) and have been adapted for the purposes of this study. The entire interview questionnaire was refined several times, e.g. merging or separating questions and adjusting question order, before conducting the first interview.

3.5.3.2 Document and Texts

Document analysis is a systematic procedure for reviewing or evaluating both printed and electronic document material. Document analysis, like other investigative methods in qualitative research, requires that data be examined and interpreted to extract meaning, gain understanding, and develop empirical knowledge (Rapley, 2007; Corbin, 2008). Documents contain text, tables and figures that have been recorded without a researcher's intervention. Atkinson et al (1997 p. 47) refer to documents as 'social facts' which are produced, shared, and used in socially organised ways. The analytic procedure involves finding, selecting, appraising and synthesising document data. Document analysis produces data such as text excerpts or quotations which are organised into major themes, categories, and case examples specifically through content analysis (Labuschagne, 2003). A detailed document analysis process was planned and adhered to throughout the study (See Appendix H).

Document analysis is often combined with other qualitative research methods for triangulation purposes - 'the combination of methodologies in the study of the same phenomenon' (Denzin, 1970, p. 291). The qualitative researcher is expected to draw upon multiple sources of evidence; that is, to seek convergence and corroboration using different data sources and methods. Apart from documents, such sources include participant or non-participant observation, physical artefacts and as seen in this study, interviews (Yin, 1989). Triangulation was used in this study is described in section 3.5.4.

The advantages of documents are that they can be reviewed repeatedly, they are unobtrusive, not created for the case study, broad in coverage (long span of time) and cover many events and sub-settings. There are disadvantages, however. Documents may contain insufficient detail as they are created for some purpose other than research and are therefore created independent of a research agenda. In addition, there can be biased selectivity if collection is incomplete, retrievability can be low, there may be reporting bias via reflection of researcher bias and access to documents may be blocked (Bowen, 2009).

3.5.4 Triangulation

Having its origin in navigation and military strategy, the term 'triangulation' is used to refer to the observation of the research issue from at least two different points (Flick, 2011). Triangulation can allow for greater accuracy as researchers 'can improve the accuracy of their judgments by collecting different kinds of data bearing on the same phenomenon' (Jick et al., 1979, p.602). Denzin (1988) proposes four triangulation types: multiple methods, multiple data sources, multiple investigators, or multiple theories to confirm emerging findings. Multiple methods of data collection were used for triangulation in this research: interviews and documents. Participant interview responses were checked against data documents relevant to the research questions (Merriam et al., 2015). In addition, selected participants had distinct perspectives. For example, different teams, practices and roles, working with differing technologies, methodologies and in different areas of the SDLC. Documents were also gathered and analysed. These included source code analysis reports, project initiation documents, IT strategy documents support knowledge wikis. This was used as another data source to complement and cross check the findings from the interviews.

Overall, the combination of documents and qualitative interviewing in a single study may help researchers to understand participants' diverse experiences, deep insight, differing perspectives and actual behaviours through triangulation and thus increase credibility in the research findings (Atkinson et al., 2002).

3.6 Data coding and analysis

Following the data collection from the semi-structured interviews and organisational documentation, the final step of the process was to interpret the data. This research used the Miles and Huberman (1994) approach to data coding for data analysis. This approach is based on a framework which is applicable to a variety of qualitative studies (Figure 15). All data was analysed using a coding system. This research used QDA Miner Lite, a computer-assisted qualitative data analysis software (CAQDAS), for data analysis included data coding, sorting and retrieval (Provalis Research, 2016).

A code is a term that represents a theme, theory, idea, characteristic, or data dimension. Codes are used to 'pull together and categorize a series of otherwise discrete events, statements, and observation which they identify in the data' (Charmaz, 1983, p.112). Coding is the action of identifying text in a document that exemplifies ideas or concepts and connecting it to a node that represents that idea or concept (Basit, 2003). Codes should be precise, enable a large part of the data material to be consumed under it and

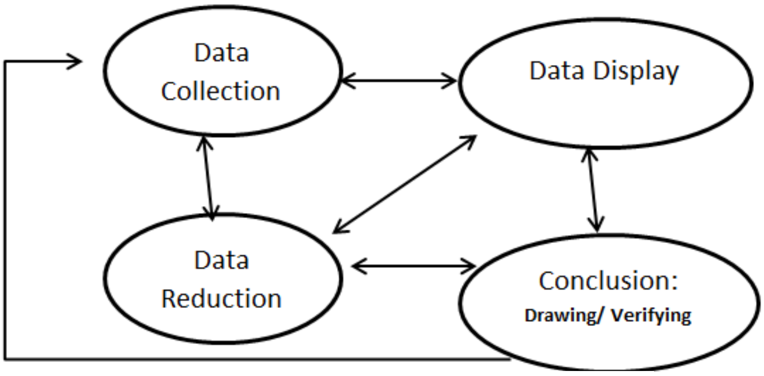
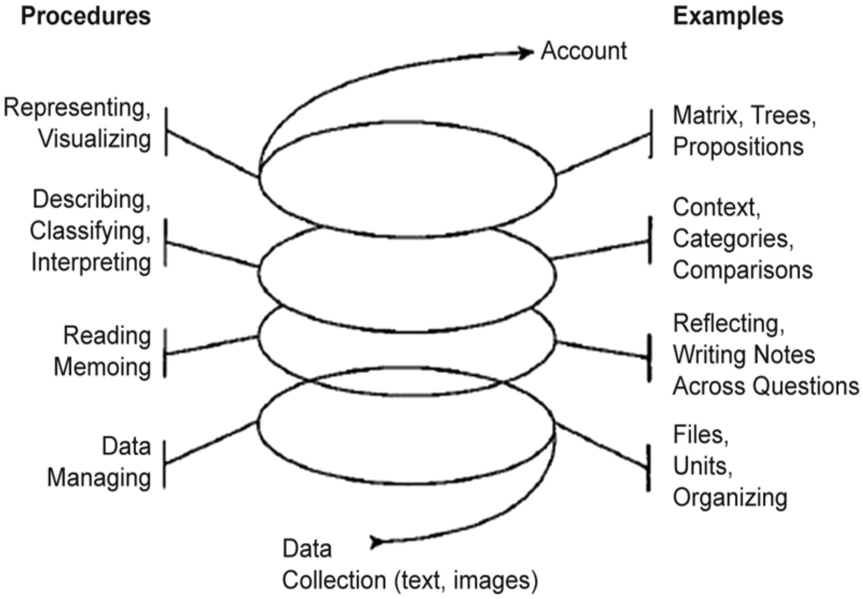


FIGURE 15 - Miles and Huberman approach to data coding for data analysis

relevant to the research question (Kelle, 2001). Multiple codes can be assigned to the same text segment in a document. As the coding took place, coding categories were revised. During coding, memos were taken which were used to suggest new interpretations, as well as connections with other data. Creswell (2007) describes the data analysis spiral which is applicable to a variety of qualitative studies. The procedures illustrated in Figure 16 were also used by this study.


Procedures
Examples
Account
Representing, Visualizing
Matrix, Trees, Propositions
Describing, Classifying, Interpreting
Context, Categories, Comparisons
Reading Memoing
Reflecting, Writing Notes Across Questions
Data Managing
Files, Units, Organizing
Data Collection (text, images)

Source: Creswell (2007)

FIGURE 16 – The Data Analysis Spiral

Case study findings are significant since they present the participants' perspectives without any complicated statistical procedures and as such the results are more comprehensible for the intended readers, that is, software practitioners. The first step in deciding how to analyse the collected data is to define a unit of analysis (Trochim, 2006). The unit of analysis is the major entity that is being analysed in the study (Kenny, 1996). In case studies, there is holistic or embedded design (Yin, 1989) as mentioned in section 3.4.4. A holistic design examines the global nature of the phenomenon, whereas an embedded design also pays attention to subunit(s). This study used a holistic design to analyse the units. The unit of analysis chosen for this study is an organisation. The group consisting of individuals that have experience with conducting TDM in the organisation being studied.

3.7 Ethical Considerations

Data collected for academic research will have ethical implications. Issues can arise in relation to informed consent, confidentiality, privacy, copyright, intellectual property rights and corporate permission. Ethical approval is required before any studies involving human participants can commence. An application for ethics approval was submitted to the School of Computer Science and Statistics (SCSS) Research and Ethics Committee on the 19th of April 2017 for approval to carry out the research. Following the research supervisor review and submission approval, permission to proceed with gathering data was received on the 8th June 2017 from the Ethics Committee. The ethical application included a research proposal form outlining the details of the research and ensured any conflicts of interests were identified. Each interview participant was provided with access to all relevant documentation (See Appendix F).

3.8 Validity and Reliability

Quality is an important issue in research. Validity refers to the quality of the conclusions that this researched has reached based on the data collected. It is the ‘best available approximation to the truth of a given proposition, inference, or conclusion’ (Trochim, 2006). In quantitative research, validity can be subdivided into four types. They are conclusion, internal, construct and external. Each type addresses a specific methodological question. Validity questions are cumulative, the question that each validity type addresses assumes a positive answer to the former one. The terms ‘Reliability’ and ‘Validity’ are essential criterion for quality in quantitative studies. However, as this is a qualitative research study, a different set of standards will be used for judging the research quality. The terms credibility, confirmability, dependability and transferability were used as an alternative to the more traditional quantitatively-oriented criteria for quality (Lincoln et al., 1985).

TABLE 3 – Criteria for Judging Qualitative Research

Traditional Criteria for Judging Quantitative Research	**Alternative Criteria for Judging Qualitative Research**
Internal Validity	Credibility
External Validity	Transferability (aka. Applicability)
Reliability	Dependability (aka. Consistency)
Objectivity	Confirmability (aka. Neutrality)

Lincoln et al (1985, p. 300) use ‘dependability’, which corresponds to ‘reliability’ notion in quantitative research (Table 3). Clont (1992) and Seale (1999) endorse the concept of dependability with consistency or reliability in qualitative research. Data consistency is achieved when the research steps are verified through interrogation of raw data, products of data reduction products, and memos (Campbell, 1996). When judging qualitative work, Strauss et al (1990 p. 250) suggest that the ‘usual canons of ‘good science’…require redefinition in order to fit the realities of qualitative research’.

3.8.1 Credibility

Credibility involves establishing that the research results are credible from the perspective research participants. This research aims to understand the TD phenomena and its management from software practitioner’s perspective, the participants are the only ones who can legitimately judge the results credibility (Trochim, 2006)

3.8.2 Transferability

Transferability refers to the extent to which the study’s results can be generalized or transferred to other contexts or settings. This study has attempted to enhance transferability by describing the research context and the assumptions that were central to the research. Despite this, transferability is predominantly the responsibility of the one doing the generalising. Therefore, the person who wishes to transfer this study’s results to a different context is responsible for judging the sensibility of the transfer (Trochim, 2006).

3.8.3 Dependability

There is a need for this research to account for the ever-changing context within which research occurred. This study was responsible for describing any changes that occurred in the studied setting and how these changes affected the method the research approached the study (Trochim, 2006).

3.8.4 Conformability

Several strategies were adopted during this research for enhancing confirmability.

- The procedures for checking and rechecking the data throughout the study were documented.
- Negative instances that contradict prior observations were searched for and described.

- A post-study audit was conducted by examining the data collection and analysis procedures with the purpose of identifying research bias or distortion.

Some qualitative researchers have argued that validity is not applicable to qualitative research. However, there is agreement that there is a need for some qualifying check on their research. Creswell (2003) suggests that a study's validity is affected by the researcher's perception of validity in the study and their paradigm assumption choice (Lincoln et al., 1985; Seale, 1999; Mishler, 2000; Stenbacka, 2001; Davies et al., 2002). Maxwell (2005) posits that the degree to which an account is believed to be generalisable is a factor that distinguishes qualitative and quantitative research approaches.

3.9 Methodology Limitations

Every study has limitations (Leedy et al., 2005) or 'potential weaknesses or problems with the study identified by the researcher' (Creswell, 2007, p.198). A limitation is an uncontrollable threat to the study's internal validity.

3.9.1 Researcher Bias

Although measures were taken to remove bias, the researchers influence cannot be eliminated in a case study. This could affect the collected data reliability (Robson, 2002). As this study used a single case study strategy at a single software services organisation, the research findings applicability is limited to other software service providers. The research findings may not be applicable to organisations who do not outsource their software implementation and support services in any capacity, although this is becoming more uncommon.

3.9.2 The Hawthorne Effect

As this research gathered data via participant interviews, the Hawthorne Effect is a limitation. The effect concerns research participation, the consequence is observation awareness and thus, possible impact on behaviour (Chiesa et al., 2008). This has the potential to impact the research results, particularly during the face-to-face interviews. The researcher should acknowledge and understands the influence they may have on the interviewee. The research should monitor their actions, reactions and bias to be a reflective researcher (McCormick et al., 1988).

3.9.3 Ecological Fallacy

Measures were taken to avoid researcher fallacies such as the ecological fallacy where by the conclusions drawn from an analysis conducted at a group level may not apply at the individual level (Robinson, 1950). The exception fallacy was avoided by analyses at the individual level not being applied unconditionally to the organisation level. The analysis conducted for this study was done at the organisational level and at which generalizations should be made.

3.10 Lessons learnt

The research process poses several obstacles which overcome.

3.10.1 Geographical Constraints

One of the semi-structured interviews could not be conducted face-to-face as the participant was in South America during the interviewing period. For this participant, the voice-over-internet tool ‘Skype for Business’ was used to carry out this interview. The advantage of this was the clarity of the audio recording and the similarity of the conversation to normal skype conversations that the participant was used to conducting daily. This allowed the participant to be comfortable with their responses which resulted in detailed descriptions of their experiences with TD and TDM.

3.10.2 Interview Scheduling and Transcribing

As interviews were mostly conducted on-site in the studied organisations offices, there was difficulty in securing time slots to conduct the interview with participants due to participants’ busy schedules. Often, interviews were rescheduled and delayed to facilitate a suitable time for participants. To simplify the interview administration, participants were sent the relevant ethical forms to review with the interview invite. Transcribing interviews was time-consuming. After transcribing three participant interviews without an assisting software tool, a transcribing tool was sued to assist with the transcribing of audio files (Provalis Research, 2016).

3.11 Conclusion

In summary, this study can be defined as an interpretive exploratory case study, as the study's objective is to discover the TD understanding and how TDM is conducted without a prior hypothesis (Robson, 2002).

TABLE 4 – Adopted Research Methodology Summary

Philosophy	Interpretivism
Approach \ Epistemology	Induction
Methodological Choice	Multi-method qualitative
Research Strategy	Exploratory Case Study
Time Horizon	Cross-sectional
Data Collection Method	Semi-structured Interviews & Document Analysis

The following chapter graphically presents the data collected and outlined in the context of each research question. Data analysis is conducted with the goal of building a basis for conclusions to be drawn.

Chapter 4: Findings and Analysis

4.1 Introduction

This chapter describes findings pertaining to each research question after data collected from semi-structured interviews and organisation documentation was analysed within the context of the TD topic. The first section of this chapter presents some background information on the data collection and analysis procedures and the profile of the research participants. In the second section, semi-structured research results are presented. This is followed by results of the document and textual analysis. Both groups of data are analysed and presented in individual sections corresponding the relevant research question.

4.2 Profile of the Case Study Organisation

The semi-structured interviews and document and textual analysis were conducted on an Irish IT consulting organisation, headquartered in Dublin. The organisation provides software services to clients predominately based in Ireland and the U.K.

4.3 Interviews

4.3.1 Data collection

4.3.1.1 Selection of prospective participants

The prospective participants were experienced in software development in a software services provider capacity. Participants were actively dealing with issues of TD and had recognition and knowledge of the TD concept. The prospective participants were recruited through enquiries to various team leads as to the use of TDM within their respective teams and other referrals from these team leads. This increased confidence in the opportunity to collect high quality and rich information from the interviews. It was also recommended by a previous study conducted by Taksande (2011) that the quality of results would be better served if participants had over ten years' experience. Therefore, as part of the selection criteria, all participants were required to have at least ten years software development experience to participate in the study.

4.3.1.2 Interview Requests and Scheduling

On receipt of ethical approval and after appropriate participants had been identified, 15 invites were sent out via the organisations email system to the selected consultants. Both the prospective participant information sheet and the informed consent form were included in the invitation to participant along with some main important points to note. The participants were asked to read this material prior to the interview date and informed that they would be required to sign the informed consent form before the interview could commence. 14 of the invites were accepted with the remaining invite going unanswered as the consultant had subsequently left the organisation. Interviews were then scheduled with each of the participants separately. The interviews were conducted over the course of approximately one month. The first interview took place on the 12th June 2017 and the last on the 7th July 2017. The participants came from 11 different teams across the organisation with 15 different projects being covered. The summary of interview information is displayed in Table 5 below with more detailed participant attributes detailed in Appendix A and B.

TABLE 5 - Interview Information

Participant ID	Interview Medium	Interview Date and Time	Interview Length (Minutes)	Interview Location
P1	Face-to-face	12th June 2017 14:00	24.09	Client Site
P2	Face-to-face	16th June 2017 10:00	33.59	Dublin Office
P3	Face-to-face	16th June 2017 14:00	29.52	Dublin Office
P4	Face-to-face	16th June 2017 17:00	56.42	Dublin Office
P5	Face-to-face	28th June 2017 09:30	37.07	Dublin Office
P6	Face-to-face	28th June 2017 13:00	34.1	Dublin Office
P7	Face-to-face	28th June 2017 13:45	27.57	Dublin Office
P8	Face-to-face	28th June 2017 14:30	26.23	Dublin Office
P9	Face-to-face	28th June 2017 15:30	27.12	Dublin Office
P10	Face-to-face	5th July 2017 15:00	36.14	Client Site
P11	Skype for Business *	5th July 2017 18:00	52.56	Off-Site
P12	Face-to-face	6th July 2017 15:30	36.23	Client Site
P13	Face-to-face	7th July 2017 13:45	26.3	Dublin Office
P14	Face-to-face	7th July 2017 14:30	34.07	Dublin Office
Total			481.01	
Average			34.4	

* One interview was carried out via the “Skype for Business” application, the Microsoft voice over internet protocol application (VOIP), due to geographical constraints.

4.3.1.3 Interview procedures conducted

All participants signed a printed copy of the informed consent form before the interview commenced. Interview audio was recorded electronically on an encrypted laptop for all interviews. The electronic audio recordings were stored on the same laptop. The semi-structured interviews covered the participant background, TD understanding and their perceived attitudes towards TD and its management, influencing factors for TDM and TDM techniques and challenges faced. All participants were allowed the opportunity to share any other information that they felt was appropriate. All interviews except one were conducted face-to-face. One interview was conducted via a Skype call due to geographical separation between the researcher and the interviewee. A total of 481 minutes of audio recorded data were collected. Interviews had an average duration of approx. 34 minutes.

4.3.2 Interview Data Analysis

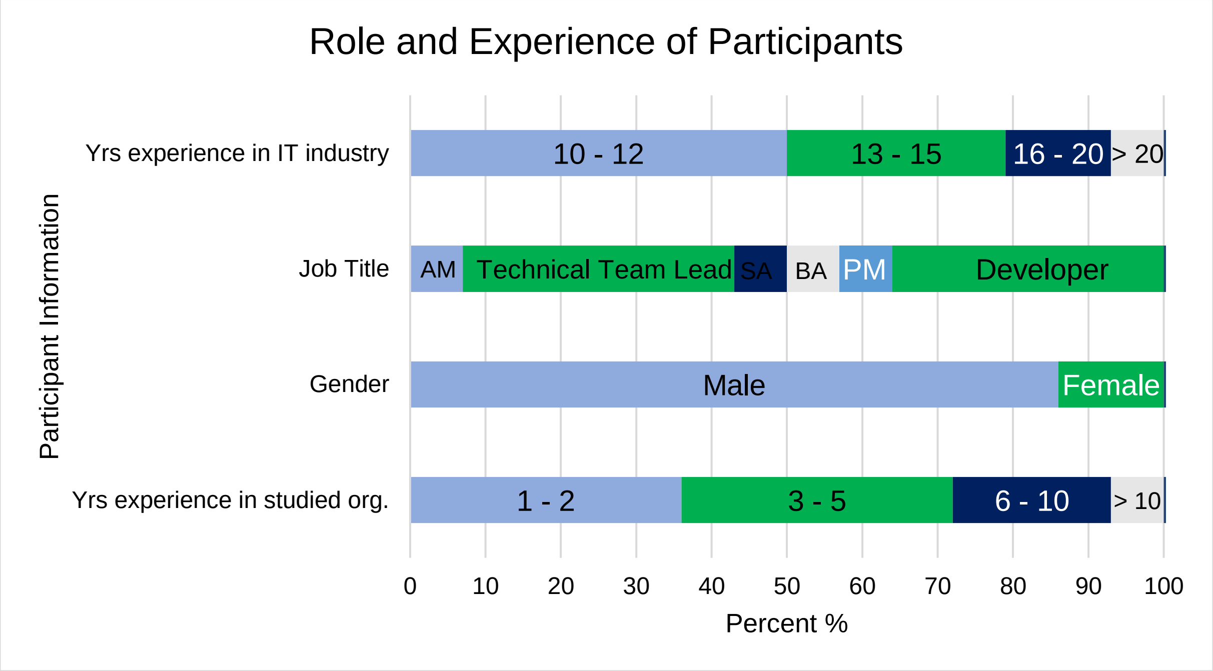
Interview audio recordings were transcribed to text documents as soon as possible individually after completion. An online transcription software tool was used to assist transcription of the audio recording (Transcribe, n.d.). The audio transcriptions resulted in a total of 66,891 clear text words (66,383 after obfuscation of personal and client data were complete) (See Appendix E). As soon as the transcription was completed a copy of the transcribed interview was provided to the participant to review. All interview information was stored in the case study database. This was used to record how and when evidence was collected and any other surrounding circumstances. A chain of evidence was maintained throughout the case study. This was done to increase reliability of the information in the case study, to allow an external observer to follow the derivation of any evidence and to trace the steps in either direction. Interview transcripts were coded using the qualitative analysis using the Miles and Huberman (1994) method with the assistance of the free qualitative analysis software tool, ‘QDA Miner Lite’ (Provalis Research, 2016). Codes were assigned inductively. Coding was conducted iteratively and constant comparison took place by existing codes to be testing against the unanalysed dataset (Silverman, 2000). The code book was saved in drafts on completion of each iteration of coding. Codes were then assigned to the relevant research question (See Table 6).

TABLE 6 - Example of the Data Coding Process

Interview Transcript Text	**Categories**
“Developers' mindset and then an **organisational culture**. A **developer's mind-set** is important”	Developer Mindset, Organisational Culture
“You know how committed they are to doing the right thing. Do they want to get just throw something in. The same for scaling up to the company wide mentality of just throwing something in. You see some companies that don't bother with unit testing. **That is obviously going to impact on how a team performs”.**	Developer Mindset, Organisational Culture
“They want to get things done properly and get things done in the right way and I think most developers that I have met and worked with down the years and are worth half their salt are like that. **They want to do the right thing and they want to do it the right way**, **they don't want to just throw something together and not be bothered**. I'd imagine that most people working with [the studied organisation] are like that. You really need that mindset to be good in IT anyway”.	Developer Mindset
“Developers need for **self-actualisation through perfect code.”**	Developer Mindset
“I have worked with people who would **have almost a sort of a rain man type approach with their code**. Indenting obsessed for example”.	Developer Mindset

The classification of the participants provides the context for the analysis of the interviews. The responses of each participant might be affected by their personal circumstances, knowledge or experience. Therefore, the below participant classification illustrated in Figure 17 aims to provide a base for comparison among the different participants (See Appendix A).

FIGURE 17 - Role and Experience of Participants



AM – Account Manager
SA – Solution Architect
BA – IT Business Analyst
PM – Project Manager

NOTE: One participant (P2) identified as being both a technical team lead and a solution architect. Technical Team Lead was chosen as the role title for data analysis purposes.

4.3.3 Interview Findings

4.3.3.1 TD Understanding and Attitudes towards TDM (RQ1)

4.3.3.1.1 TD Understanding

All participants were asked the following interview question: How would you describe the term technical debt? The responses are summarised in Figure 18 and the complete table of descriptions can be found in Appendix C.

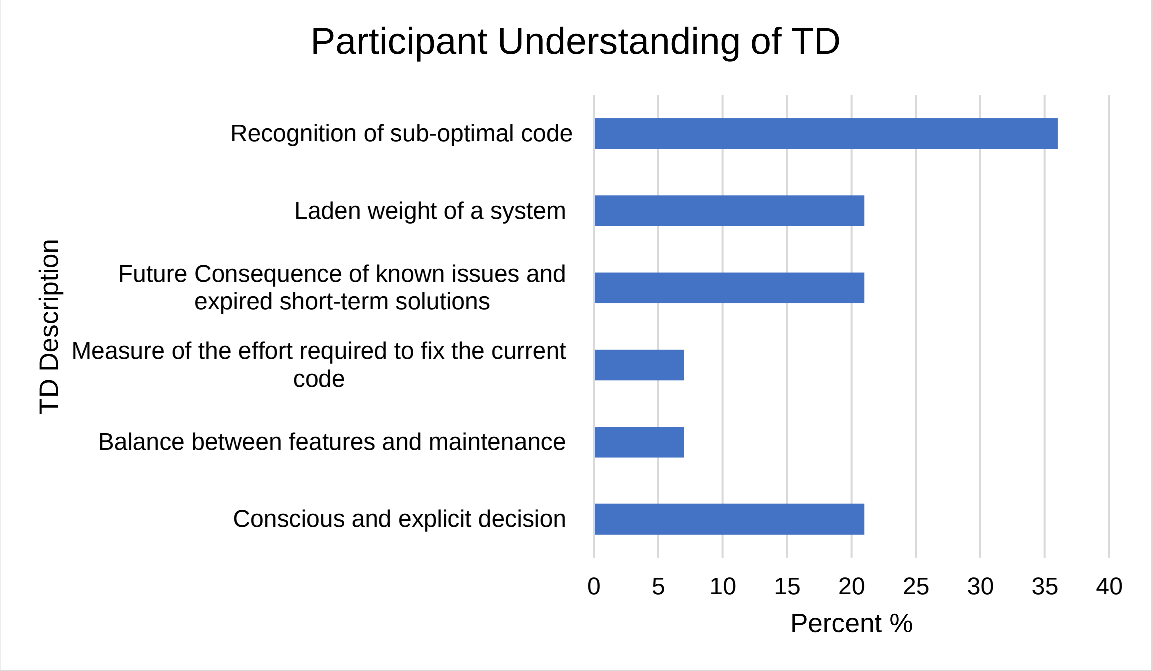


FIGURE 18 – Participants' Understanding of TD

50% of participants offered an analogy to describe the TD phenomenon. The full list can be found in Appendix D.

- *“It is like buying a load of things on your credit card and just letting the interest accrue. You got to come back and fix it up. You can't just let it get out of control”.*
- *“It's like trying to change a tyre on a car when it's driving down the motorway at 150km!”.*
- *“If you were running oil rigs and things like that you would be obliged to tell the shareholders about the upkeep and maintenance and the various risks and everything”.*
- *“If somebody said I'm going to throw junk on to the roof of Harrods and I'm going to keep doing that and one day the roof falls in and Harrods must close for a few months, they would lose millions and millions. They would never do that because that is not a sensible thing to do. Software is abstract and intangible even though it's something that runs businesses and people just look at the surface”.*
- *“It's like the ‘Buy cheap, buy twice advice”.*
- *“Carrying additional time and effort forward from one sprint to the other which is a wall of debt”.*
- *“It's like students taking out student loans so that they can go to college and maybe earn more in the future”.*

4.3.3.1.2 Attitudes

The attitudes of the interview participants are summarised in Figure 19.

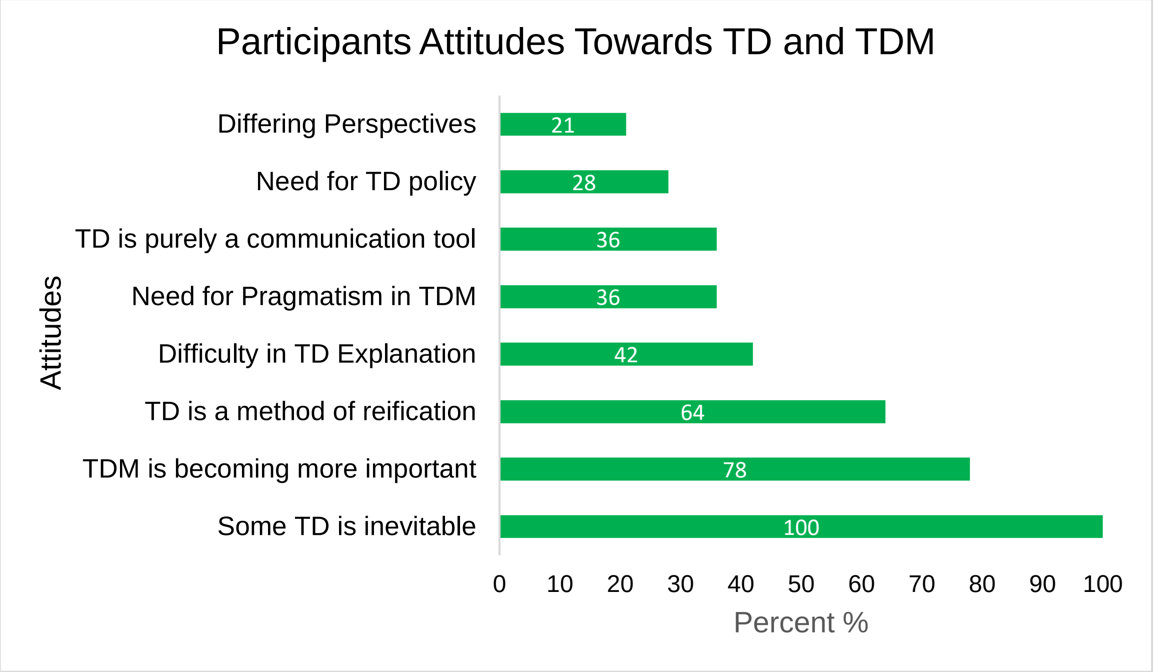


FIGURE 19 – Participants' Attitudes Towards TD

In addition to the above, 93% of participants indicated that they believed that attitude impacted on the management of TD and influenced the consultant's decisions to either personally make the decision or advise the client to accrue new TD.

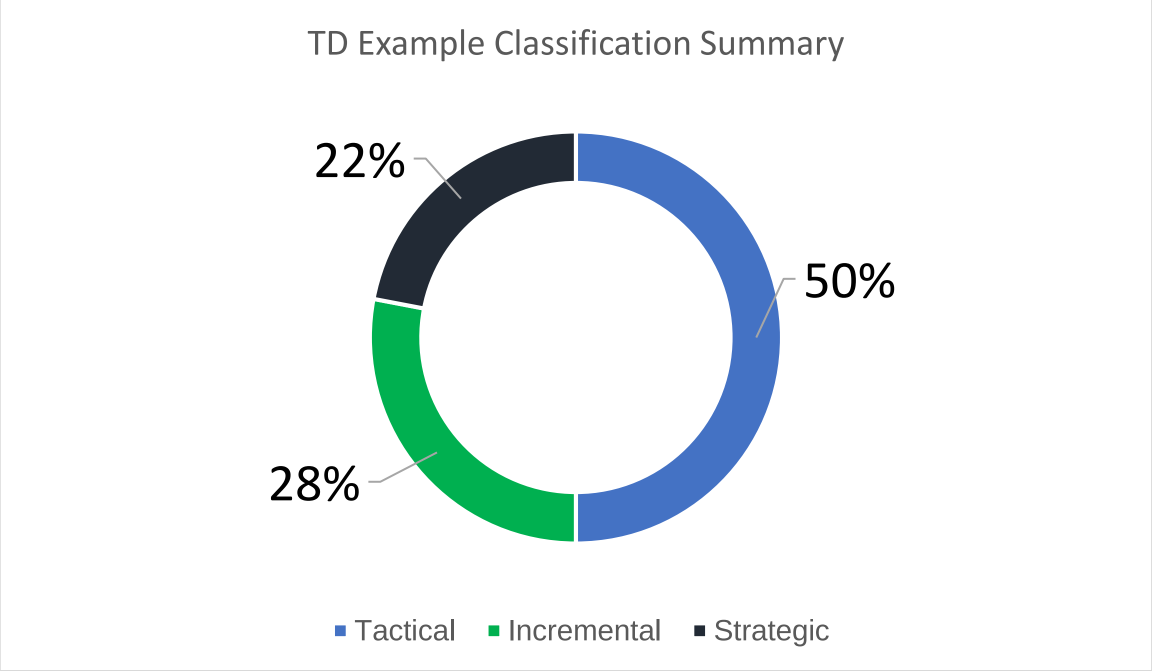


FIGURE 20 – Classification of TD Examples Offered by Participants

4.3.3.2 Organisational Factors that Influence TDM (RQ2)

While risk, impact, urgency and budget were mentioned by all participants as quantitative factors that influence TDM. The organisational factors are outlined below:

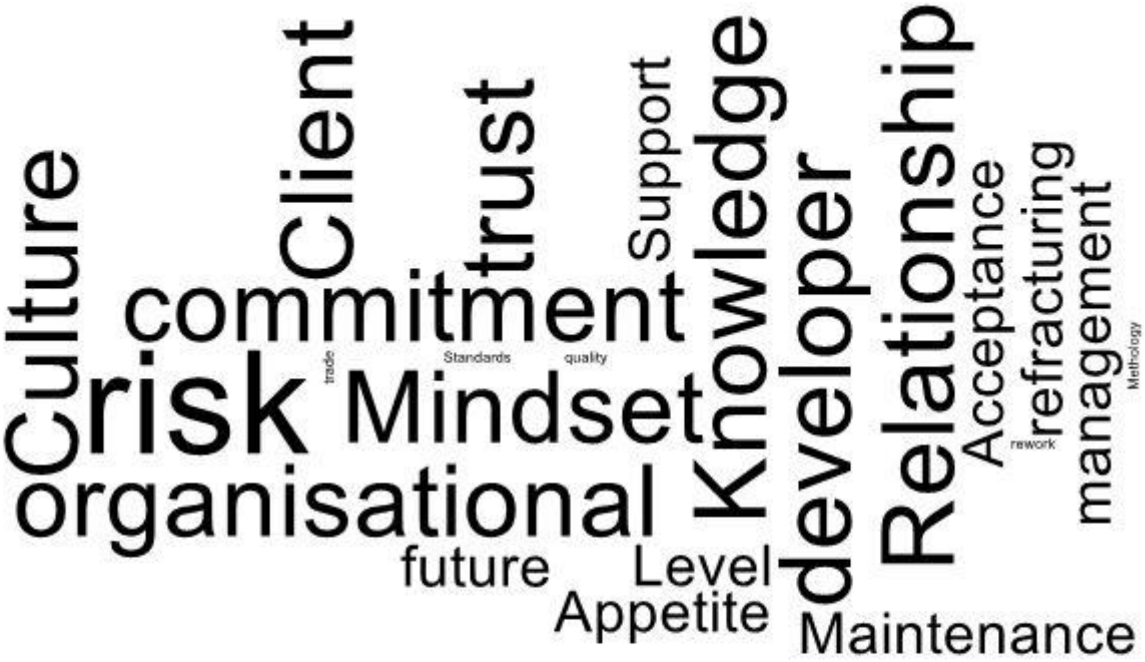


FIGURE 21 – Organisational Factors That Influence TDM

4.3.3.2.1 Risk Appetite

93% of participants viewed the risk appetite of the organisation as influencing TDM. In one example, P1 stated that the organisations appetite to risk influenced how it managed TDM decisions - *"The developers are being extra cautious with what they are doing and there may be some room for them to take a decision to accrue some technical debt, but they haven't because I think they're too afraid of the consequences downstream".*

4.3.3.2.2 Organisational Culture

93% of participants perceived organisational culture as a factor in TDM decisions. The set of core values to which studied organisation's culture is focused was mentioned by most participants. P10 states *"Culture is a huge factor in technical debt management".* P11 felt that *"Having a culture that promotes integrity and honesty and being clear with the* *customers is going to be good in terms of minimising technical debt".* P1 mentions *"The honesty and integrity and customer first values I think do have a big impact because you are actually think about it".* P12 says that *"we have an obligation to manage our client's technical debt as much as we can"* and *"technical debt occurs as we agreed and at that point it comes down to an integrity thing".* P9 admits that *"We don't hide problems and we go to the client when necessary"* with another participant corroborated this by reporting that *"You just can't have a culture where you have poorly implemented solutions, poor coding standards and poor design"* P9 stressed that *"There must be a culture where everybody has to try to follow best practise and meet coding standards and code reviews, those kinds of things are very important".* Although difficulty arises when assessing the trade-off between organisational culture and the ability to win new business - *"You need to find a balance there when you are responding to a tender between your honesty and integrity, your excellence etc. They are important, plus your ability to win the tender or win any piece of business and that's a really hard trade off to make".*

4.3.3.2.3 Management IT Knowledge

72% of participants cited that the extent of managements IT knowledge is an influencing factor for TDM decisions. There was a general appreciation by participants for managements understanding of TD issues. P4 recounts an example of a new product manager for a project they were involved in – *"I raised a TD issue with [the manager] and they were able to understand the predicament for the project straight away and was ultimately able to make a more informed decision based on their experience in software development".* The opposite experience was encountered by P13 when trying to articulate the reduction in the software development team velocity due to TD incurred for previous releases – "*They could not understand why changes and new functionality was taking so long, they just did not have the IT knowledge to understand the consequences of incurring TD".*

4.3.3.2.4 Trust & Client Relationship

64% of participants gave trust and the relationship with the client as factors which influenced TDM decisions. Additionally, the contractual agreement and terms with the client were also influential. P7 stated *"we would hopefully already have a reasonable working relationship with the client".* P14 revealed that *"When we talk about the conditions for success. A fixed price situation has more risk in it and therefore is more likely to come under pressure. A T&M will have less pressure and you're more likely to make the right decisions".* P8 agrees with P14 and said in relation to a fixed price contractual agreement “*we are supporting the development of the system then the cost comes back on us if we incur debt then maybe I would make a decision differently”.* P8 said *“The risk is with the client in the sense of the functionality is broken under these circumstances but actually that decision was a short term one on our part which we are now paying to put right. So, that is why the contractual relationship comes into it”.* P9 admitted that *“We went to the customer and we needed to have a difficult conversation to say 'Look this was not done in the proper way and this needs to be fixed or we will have problems in the future”.* P6 disclosed that *“Whatever about implementing a quick fix that has detrimental effects on the scalability, decisions that will actually give you more work then that's where you would hold up your hands and say this is not the right thing to do”.*

4.3.3.2.5 Developer Mindset

50% of participants cited developer mindset as a factor that positively influences TDM decisions. Regarding software developers, P12 believes that *“They want to do the right thing and they want to do it the right way, they don't want to just throw something together and not be bothered… You really need that mindset to be good in IT anyway”.* Other participants note that developer mindset can also influence TDM decisions negatively [P12, P7 and P4]. They mentioned that they have experienced working in teams where colleagues sought *“self-actualisation through perfect code”* and unnecessary perfectionism - *“they take it as an element of pride to have the stuff squeaky clean”, “quest for purity in a code”* and*, “to satisfy some demand for purity”.*

4.3.3.2.6 Management Commitment

36% of participants viewed both the studied organisation and the client organisations management commitment or “Buy-in” as an important factor in TDM decisions. In reference to TDM as a component of tender evaluation, P14 mentioned that *“When you're trying to win a project it can be how much does the customer value those things”* with P12 attesting to the impact management commitment made on a particular project -*“We got that from management and it made a massive difference”* The timing of attaining management commitment was referenced by P10 - *“You need to say 'I know you want to ship it now and we will go and ship it now'. Next week it becomes more difficult. Even though you have done what you expected to do it is more difficult to get that buy-in after the fact”.* P4 cited the motivation of management in to TDM commitment - *“If there is an IT director who again* *buys into TDM as a legitimate concern, they can use it themselves to get additional resources as well".*

4.3.3.2.7 Future Maintenance

21% of participants viewed the prospect of future maintenance as a factor in TDM decision-making. The future velocity of the development team was considered by P13 - *"A lot of clients will appreciate that if something is more maintainable it means that we are working more quickly, so therefore we are able to do things more quickly for them".* The negative aspects of future maintenance were mentioned "*it is something that has been dragged from previous design".* P2 went as far to say that the future maintenance might be beneficial *"you're trying to keep the show on the road and it gets to a point where you have all this technical debt it which causes you to re-visit the design of some fundamental system as well and then gives you an opportunity to find new benefits".* Some short-sighted thinking was frustrating for some participants as P10 says the following: *"we could of had a finished fully working solution in place and instead it's just been shelved again until the next emergency".*

4.3.3.3 TDM Techniques & Approaches (RQ3)

There were six main techniques used for TDM activities by the studied participants (See Figure 22).

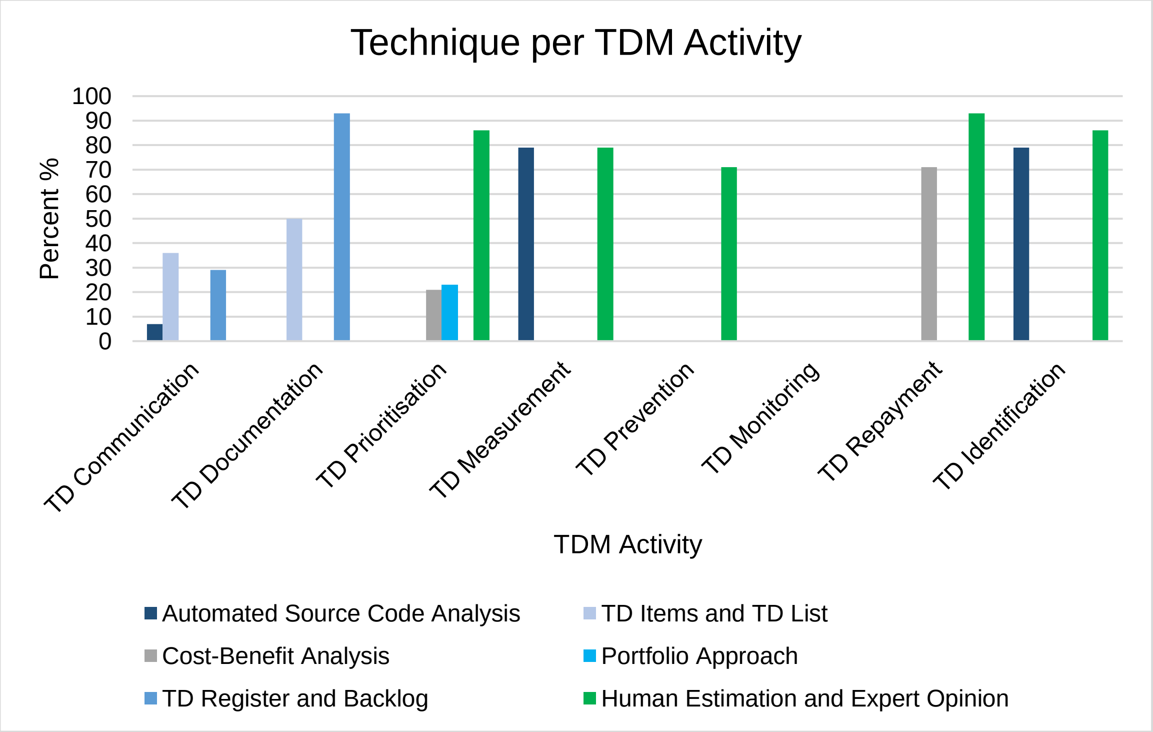


FIGURE 22 – Techniques Per TDM Activity

4.3.3.3.1 TD Register and Backlog

(TD Documentation, TD Communication)

Most participants (93%) used a TD register or backlog of TD issues for documentation or communication activities. P7 admitted aversion to incurring TD intentionally but views the TD register as valuable for tracking purposes - *“You don't want to be filling that up but you do need to keep track of the decisions that you've made and your reasons for making it”.* The adoption of agile was cited by many participants as the rationale for registering TD and creating a backlog of TD issues. P10 mentions *“a complete JIRA board that was just dedicated to project clean up tasks”* and P2 says that *“they have adopted agile practices and on their board, they have a technical debt backlog”*. Even though the TD was being documented via a TD register in most cases, participants indicated there was little appetite by clients to repay the TD that was documented in the register – *“They are not very willing to tackle the debt backlog”* and *“We accrue technical debt but we don't actually tackle it most of the time. There is no appetite there to actually fix the issues”.* Another participant describes the TD registration motivation with the following - *“we will make a [user] story in the technical debt backlog and this could be put into a sprint if we could somehow convince the product owner to reserve a percentage of their sprint for fixing things”*. The same participant goes on to reveal that although repayment is the intention, the TD Register is *“basically a graveyard”.* P5 states similar experiences with TD registration – *“An awful lot of the debt is obsoleted, it's probably not an issue any more”.*

An issue ranking visualisation in the context of all other issues was the only technique used for TD Prioritization and TD communication to reach a common understanding with clients and other stakeholders on the importance they place on each TD Item being repaid. *“We work with the customer to prioritise the work. Generally, we will highlight like technical debt at that stage and improvements we would like to make in terms of must haves, should haves and nice to haves”.*

“If it's something that causes that individual to be less productive because they cannot do a task and it is a must have for the business, either it gets prioritised because it's a legitimate concern or maybe it's handled by some other business function because it is not crucial”.

4.3.3.3.2 Human Estimation and Expert Opinion

(TD Measurement, TD Prioritisation, TD Repayment, TD Prevention)

Human estimation and expert opinion featured highly for the above TDM activities amongst most participants (between 79%-93%). P10 describes the difficulty in estimating the risk in accruing TD as *“It generally comes to interpretation of the issue”* with P11 echoing the TD research community’s sentiment – *“I don't know of any methodology for calculating the severity of the technical debt”.* P13 describes the estimation of TD in their team – “*These are the TD issues and for some of them we gave a very high level rough estimate to give them a sense of the improvement effort”.* P14 reiterates the estimation difficulty for TD issue translation to a monetary figure - “*When you try to dig into it in terms of monetary values it's very finger in the air and it's very difficult to nail down and quantify so there might be a better way of doing it”.*

4.3.3.3.3 *Automated Source Code Analysis*

(TD Measurement, TD Identification)

79% of participants used automated source code analysis for TD measurement and TD identification activities. The majority combined this technique with manual review of the code base by the team’s software architect or senior developer. The SonorQube source code analysis tool was the only tool mentioned for the measurement and identification activities. Participants who had used SonorQube were generally pleased with the functionality it offered –“*It gives you the opportunity to actually analyse the things that you have developed”.* On further investigation, the participants admitted the manual code reviews were conducted largely due to failings of the SonorQube tool. For example, integration and dependency of the system on third party code and\or architectural considerations that SonorQube did not account for. Additionally, P14 noted that the monetary amount that SonorQube estimated for TD should be *“taken with a pinch of salt”.* They continue to argue that *“It’s very useful but I think that if you handed those estimations over to an accountant they would say 'the are a lot of ifs and buts here”.*

Another participant weighs up the pros and cons of static code analysis in general - *“You must cost all those things as best as you can and you have to use tools like SonorQube to quantify the cost at a high level to say this is the technical debt that we're carrying and it is X figure according to this tool. Big strong caveats on the tool and the number that it gives* *you in both directions. It could be under or over estimating the debt. It's only looking at the Java code".*

14% of participants who primarily supported existing systems for clients where the nature of the project was primarily small to medium change requests and incident management cited Debt Service Coverage Ratio (DSCR) as a means of TD measurement, although not using the specific DSCR term.

4.3.3.3.4 TD Items and TD List

(TD Documentation)

50% of participants advised that TD Items which arose during the development or support of a system were logged in a TD Backlog via the agile focused issue management system, 'JIRA'. Of these participants, approximately 30% said that TD decisions are documented in a proposal format, uploaded to SharePoint and sent to the customer as well so that they can agree with the decision that was reached: *"We also have the incident tool and TD decisions are documented there so that everything is traceable so we always send it directly to this tool and to our clients to document the decision process between us".* The TD documentation artefacts took different formats for 57% of participants depending on the audience. An example offered by one participant is the following: *"Documentation for the technical people, for a business person, for the management team, to make sure that we are going to deliver or what was going to be built is what they are expecting. We have had to change things during the development phase in the past precisely because of a misunderstanding from documentation."*

86% of participants cited a lack of TD item prioritization and misjudged focus on solutions and problems by client organisations. 21% of participants recounted instances where they put the systems TD items in perspective by consolidating all the current development requirements, changes and initiatives (product backlog) and asking the list to be ranked in order of importance. This allowed the prioritisation of the TD items which were associated with the ranked tasks.

4.3.3.3.5 Cost-Benefit Analysis

(TD Repayment)

71% of participants indicated that they use cost-benefit analysis for to assist with TD repayment decisions. P12 stressed that *"you have to weigh up the cost benefit of things at times"* when contemplating repaying an incurred TD item. P12 also added that caution should be taken when conducting cost-benefit analysis to ensure that too much time is not being spent on it - *"you have to be careful that you don't go too overboard on that like I said before about the whole costs benefit".* Another participant, P5, echoed this cautious sentiment by stating that *"the opposite extreme means that you're also sinking an awful lot of time that's not actually an issue".* The return on investment was the upmost concern and ultimate outcome of the cost-benefit analysis. For example, two participants [P1 and P9] said that *"Return on the investment should be factored into the decision"* and *"ROI always should be factored in the decisions because in a commercial setting that is important".* The consequences of the decision to strategically incur TD was summarized by P3 with the following statement - "*The cost benefit is that whatever savings we make now we will lose and possibly make a massive loss down the line if we don't go back and address them".*

4.3.3.3.6 Portfolio Approach

(TD Prioritisation)

The portfolio technique was in use for TD prioritisation in 23% of the cases where TD items were being recorded by participants. An application, module or function would be classified according to its value to the organisation in order to prioritise TD items. Release Planning was another technique that featured in 36% of the participant interviews. Planning for new features that were being scheduled in a release would help to prioritise TD items.

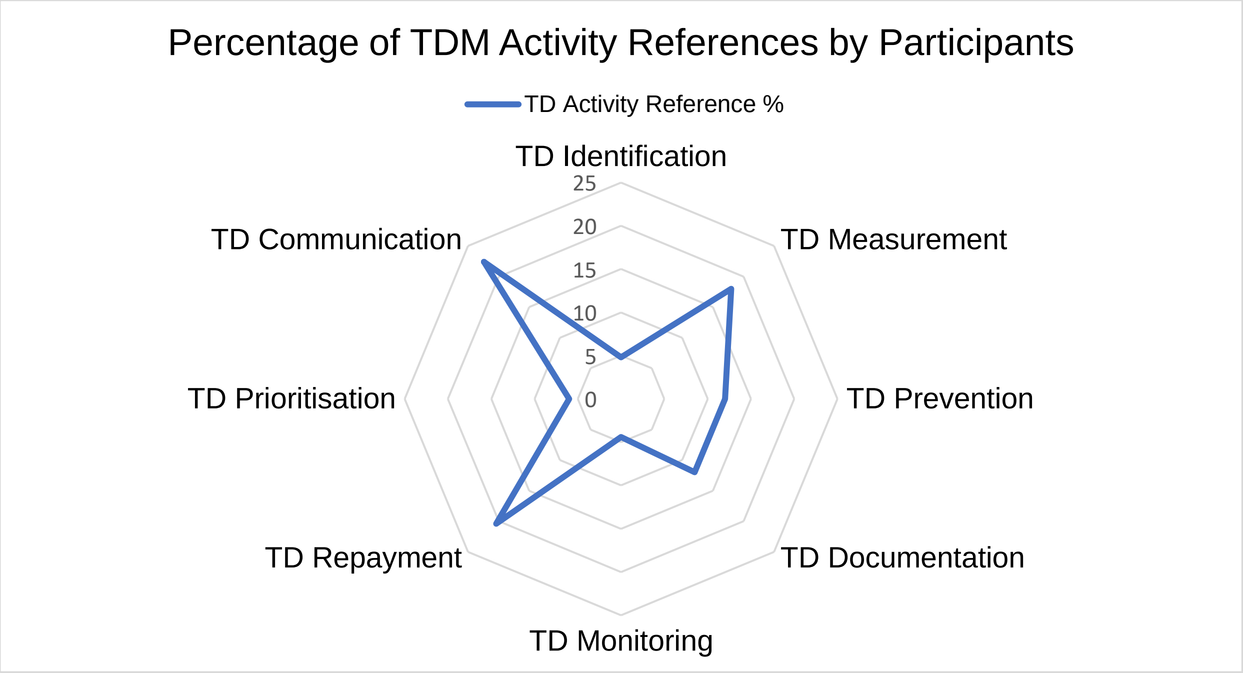


FIGURE 23 – Percentage of TDM Activity References by Participants

Total TD activity references by all participants was 250.

4.3.3.4 Challenges for TDM (RQ4)

Most of the TDM challenge themes emerged from the final question in the interview which was "Do you have any other thoughts on Technical Debt Management?".

4.3.3.4.1 Justification of TDM Investment

93% of participants referenced a current challenge of TDM being the justification of investment in processes and initiatives to implement all TDM activities properly. The initial intangibility of TDM were too much to overcome for participants to convince clients to invest in. *"It would be a very hard sell to go into a business and say we are not going to deliver any business functionality but your codes got to be really good. It's like selling someone snake oil. They're not going to see anything".* The consensus amongst participants was that although TDM is important it was difficult to justify to a non-technical audience - *"It's a hard sell. Managing technical debt is extremely important but no one is going to notice any difference initially. It's difficult to quantify."* One participant gave the example of justification of writing automated unit tests for new development. "*Development takes a bit longer, but it avoids some nasty surprises*". P8 expressed their main concerns – "*how to really make clients value technical debt and to make them understand the importance of improving their code and maybe finding a way of quantifying the savings and the gains over time. You* *know you keep seeing the same issues cropping up again and again and having to fix them repeatedly".* The challenge was reiterated by P7 – "*I don't think there is anything to be lost from making the client aware of issues but it is really how do you make them appreciate that they should be fixed".*

4.3.3.4.2 Difficulties in business and economic value transformation

86% of participants viewed the economic value transformation of TD to be one of their main challenges for TDM. For example, P8 says that *"It is very hard for us to write business cases which directors will sign off to say 'Yes, it is worth me spending half a million pounds on this' because they don't see that risk as being costly enough to justify the investment".*

4.3.3.4.3 Lack of TDM Tools

TDM activities were viewed by 50% of participants as time-consuming. Tools that could be used to assist with a TDM framework were desired to decrease the amount of time consumed by TDM activities. This requirement was articulated by P1 as *"We need to do our technical debt piece and we need a tool that you take out of the box almost. Almost the same way that you would say OK we are going to implement agile, here is our agile processes".* Participant 10 cites deficiencies with existing tools and their single focus nature: *"Some, like SonorQube, help with recording your technical debt but it is a little bit of a crude tool it only works with certain types of debt".* P10 notes that existing tools don't cater for the entire technology stack – *"It is much broader right now than it was a decade ago for example when you had only one technology. Right now, we handle at least 10 technologies that I am aware of"* and *"When you use a large range of technologies, it's harder to use a tool like that".*

4.3.3.4.4 Knowledge Management

Knowledge Management (KT) for TDM was viewed as a challenge by 29% of participants especially during stages of organisational growth – *"I think that we are pumping out high quality stuff but I think as we scale it's going to become harder to guarantee that level of quality across the organisation".* KT sessions and system support transition processes are important - *"Because in our department we are usually inheriting [code bases], we are supporting applications that you actually don't know exactly how they are working in some cases due to code complexity".* Participants also mentioned that KT in a collaborating software service provider setting was also a challenge for TDM as due to commercial sensitivities, pertinent TDM information was not always shared *“it means that maybe some information is not shared between those teams”.*

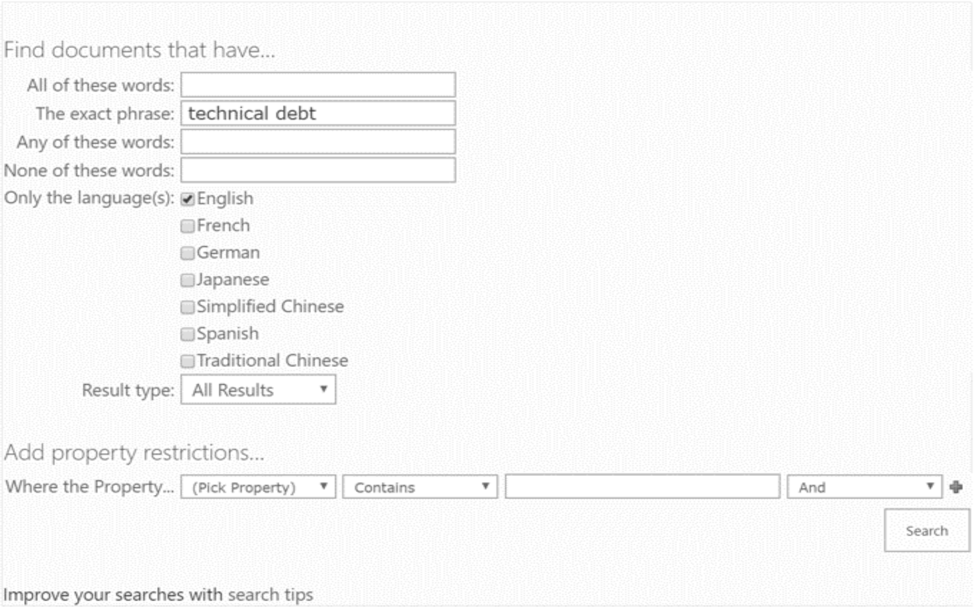
4.4 Document & Textual Analysis

Semi-structured interviews provided the research with the relevant information to contextualise the project. However, that analysis by itself does not provide enough information on influential organisational factors on TDM and TDM techniques that are in use for the studied organisation. Data source triangulation was required to test the validity of the interview findings. The purpose of triangulating is to provide a confluence of evidence that breeds credibility (Bowen, 2009). Before document analysis takes place, a detailed planning process was outlined to ensure reliable results (See Appendix H).

4.4.1 Data collection

4.3.1.1 Document Search

The studied organisations’ online document repository, was searched on the 15th June 2017 for any document with the phrase ‘Technical Debt’ included in its in-text content ("technical debt" (DetectedLanguage="en")) (See Figure 24). 83 results were returned. The same repository was searched again on the 26th July 2017 but did not return any additional relevant documents.



Find documents that have...

All of these words:

The exact phrase: technical debt

Any of these words:

None of these words:

Only the language(s): ☑ English
□ French
□ German
□ Japanese
□ Simplified Chinese
□ Spanish
□ Traditional Chinese

Result type: All Results ▾

Add property restrictions...

Where the Property... (Pick Property) ▾ Contains ▾ And ▾

Improve your searches with search tips

FIGURE 24 – Document Search

4.4.2 Document Data Analysis Procedures

4.4.2.1 Document Filtering

34 of the 83 returned documents were selected for analysis (55 megabytes (MB) of documents). The remaining documents were discarded due to one or more of the following reasons; not relevant, duplicated, or the file was missing at the time of download. The collected documents formatting consisted of the following: .docx,, .xls, .csv, .pptx, .pdf and .txt. The documentation format and filtering breakdown are illustrated in Table 7 and Figure 25 respectively (See Appendix I).

TABLE 7 – Document Search Result Categories

Document Search Result Category	**Total**
Relevant to research question	34
Duplicate	8
False Positive - Not relevant to research question	32
Not written for purposes of studied organisation purposes	8
No longer available	1
Total	**83**

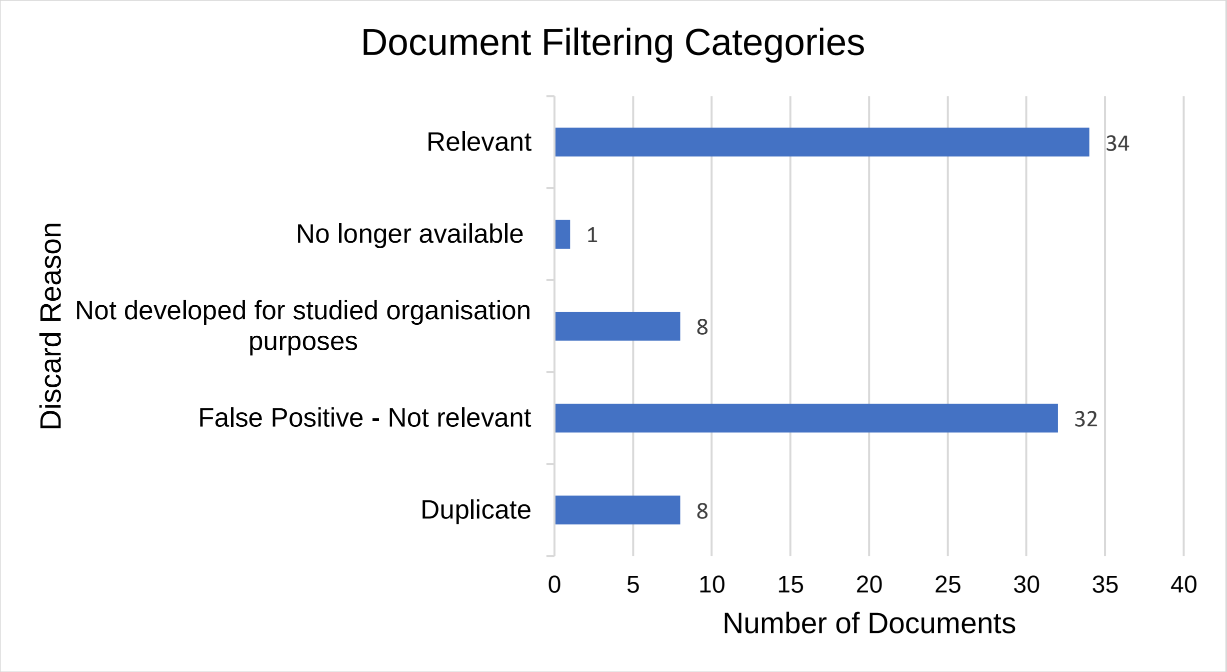
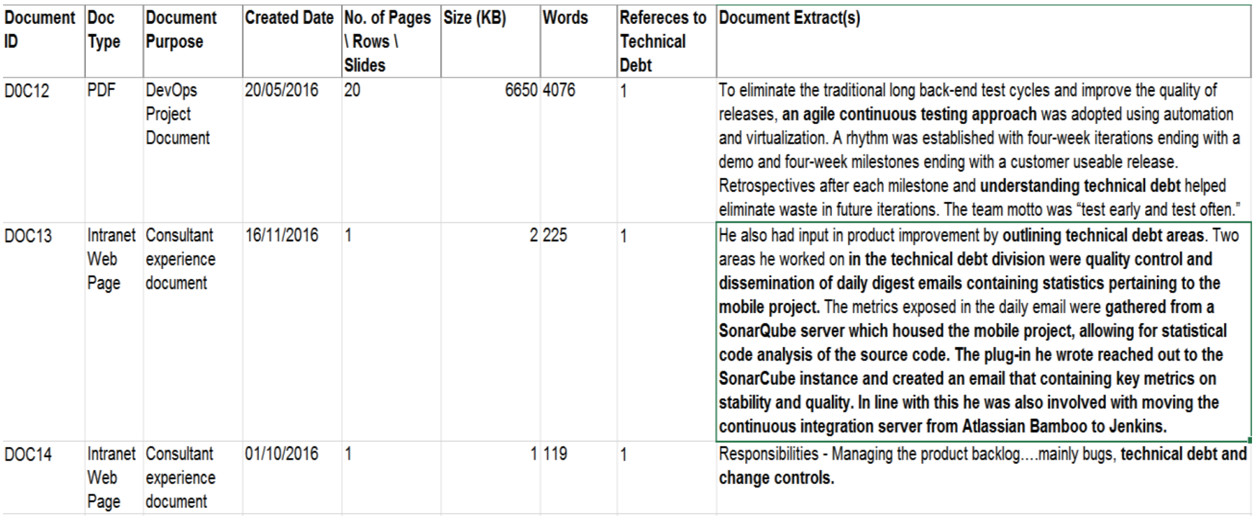


FIGURE 25 – Document Filtering Categories

4.2.2.2 Document Analysis Process

After the documents were gathered, an organisation and management scheme was put in place to create a case study database (See Figure 26). The original documents were copied for annotation and authenticity assessed. The documents agenda was assessed and any bias documented. Each of the selected documents title\name, document type, creation purpose and target audience were recorded in a Microsoft Excel file. The “unwitting” evidence, or latent content, of the documents were analysed. Latent content refers to the style, tone, agenda, facts or opinions that exist in the document. To do this, the use of particular words, phrases and concepts were quantified (O’Leary, 2014). The selected documents were then explored via thematic analysis for references to TD and TDM. The number of references and the frequency and number of occurrences within the document were extracted and categorised per the Miles and Huberman (1994) method. The information was then organised into what was related to central questions of the research (Bowen, 2009, p. 32). The purpose of this activity was to “integrate data gathered by different methods” (ibid.) to complement the semi-structured interview method. The filtered useable data were then obfuscated to remove personal information and client references. A summary of the document meta data is illustrated in Figure 27 and 28 below.



Document ID	Doc Type	Document Purpose	Created Date	No. of Pages \ Rows \ Slides	Size (KB)	Words	Refereces to Technical Debt	Document Extract(s)
D0C12	PDF	DevOps Project Document	20/05/2016	20	6650	4076	1	To eliminate the traditional long back-end test cycles and improve the quality of releases, **an agile continuous testing approach** was adopted using automation and virtualization. A rhythm was established with four-week iterations ending with a demo and four-week milestones ending with a customer useable release. Retrospectives after each milestone and **understanding technical debt** helped eliminate waste in future iterations. The team motto was “test early and test often.”
DOC13	Intranet Web Page	Consultant experience document	16/11/2016	1	2	225	1	He also had input in product improvement by **outlining technical debt areas**. Two areas he worked on **in the technical debt division were quality control and dissemination of daily digest emails containing statistics pertaining to the mobile project.** The metrics exposed in the daily email were **gathered from a SonarQube server which housed the mobile project, allowing for statistical code analysis of the source code. The plug-in he wrote reached out to the SonarCube instance and created an email that containing key metrics on stability and quality. In line with this he was also involved with moving the continuous integration server from Atlassian Bamboo to Jenkins.**
DOC14	Intranet Web Page	Consultant experience document	01/10/2016	1	1	119	1	Responsibilities - Managing the product backlog….mainly bugs, **technical debt and change controls.**

FIGURE 26 – Case Study Database

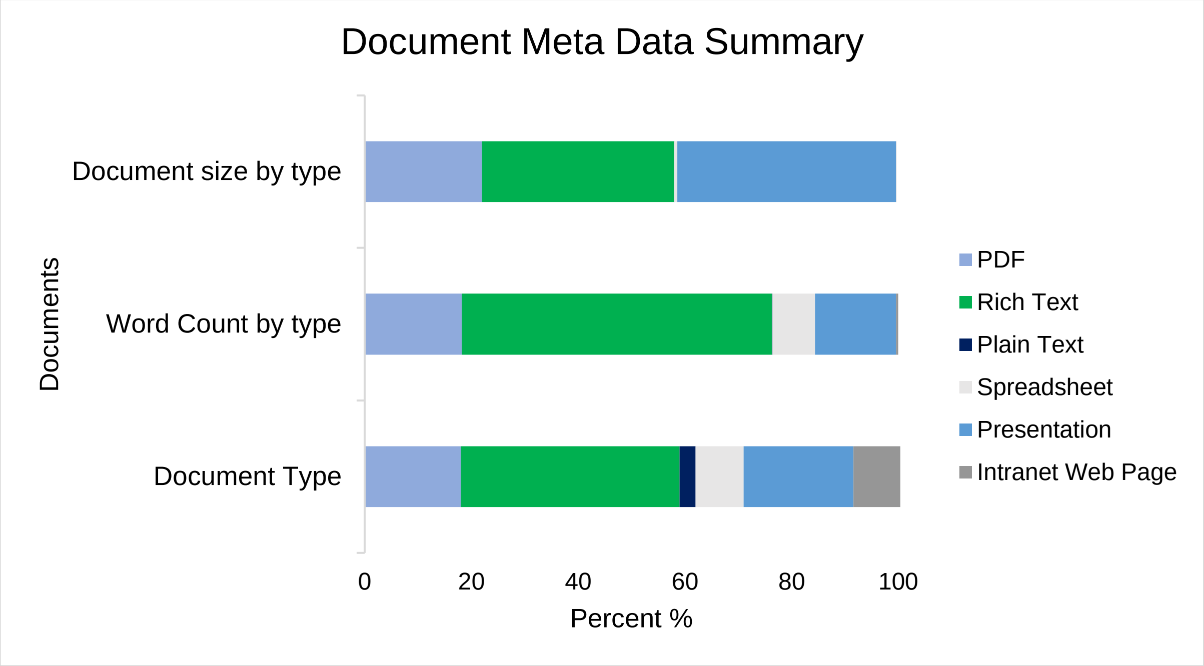


FIGURE 27 – Document Meta Data Summary

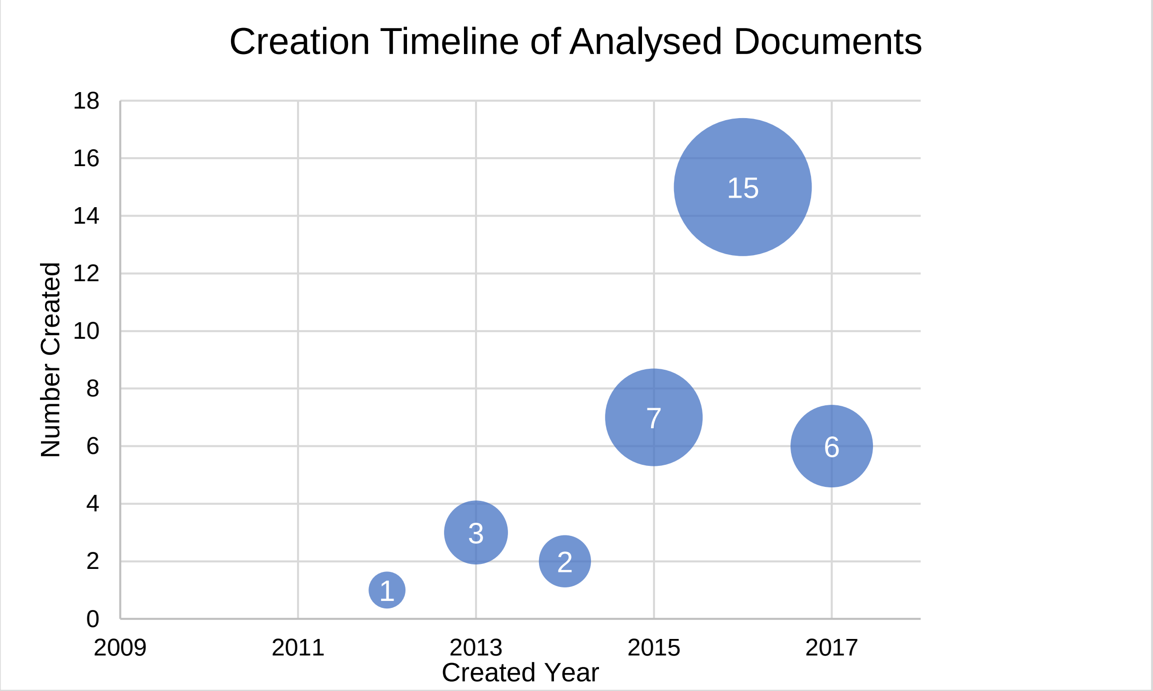


FIGURE 28 – Creation Timeline of Analysed Documents

4.4.3 Document Findings

4.4.3.1 TD Understanding and Attitudes towards TDM (RQ1)

4.4.3.1.1 TD Understanding

All documents were searched for definitions of TD. The findings are outlined below.

1. *"Functionality in existing systems includes redundant code and some ad-hoc implementations of core features"*
2. *"It is best described as a situation in software development where a shortcut or workaround is used in a technical decision. Things that save time and money today but can cost you down the road".*
3. *"A recently coined metaphor used to refer to the eventual consequences of sub-optimal software development within a code base. The debt refers to the work that needs to be done in order for a particular task to be considered complete".*
4. *"Another Moving Part/Extra Schedule (technical debt)"*

The 34 documents analysed contained 112 specific references to the term "Technical Debt". 4 (12%) documents offered an explicit definition of TD (Figure 29).

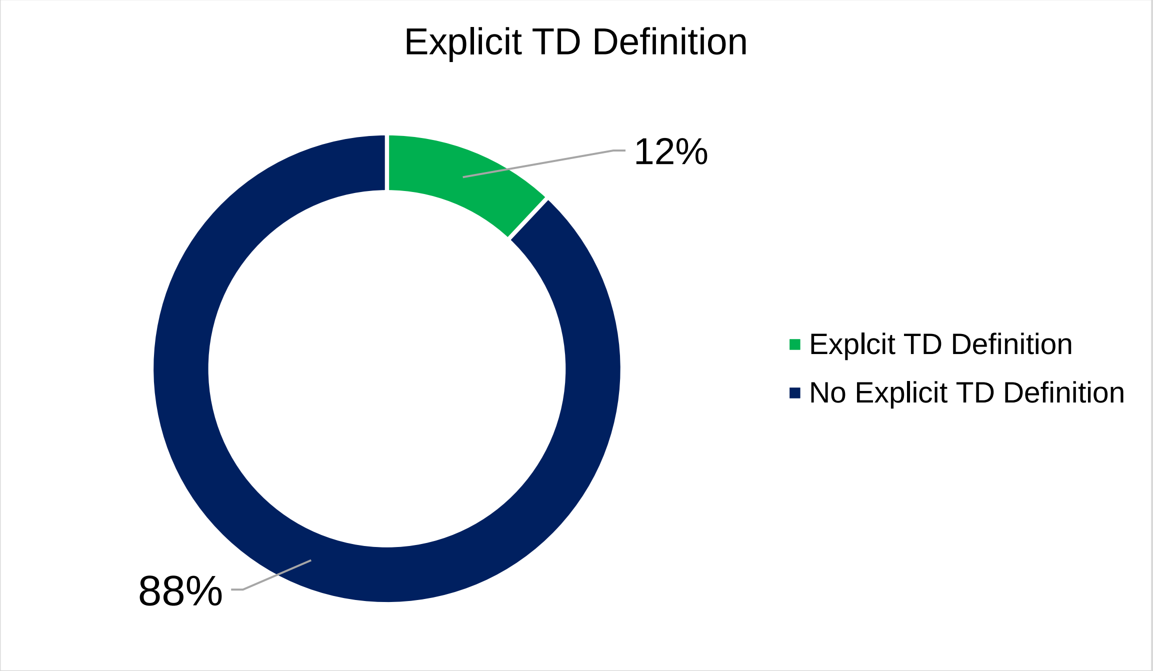


FIGURE 29 – Explicit TD Definition

Two (6%) documents offered an additional analogy to describe the TD phenomenon.

1. *"When you think of unintentional technical debt – picture it as an iceberg – mostly hidden and out of sight – until you crash into it. From an IT point of view, it's everything that the client … and the user, won't see, ever"*
2. *"The words "Big Snowball" was used to describe the large amount of ad-hoc programming changes implemented over the years in existing systems".*

4.4.3.1.2 Attitudes

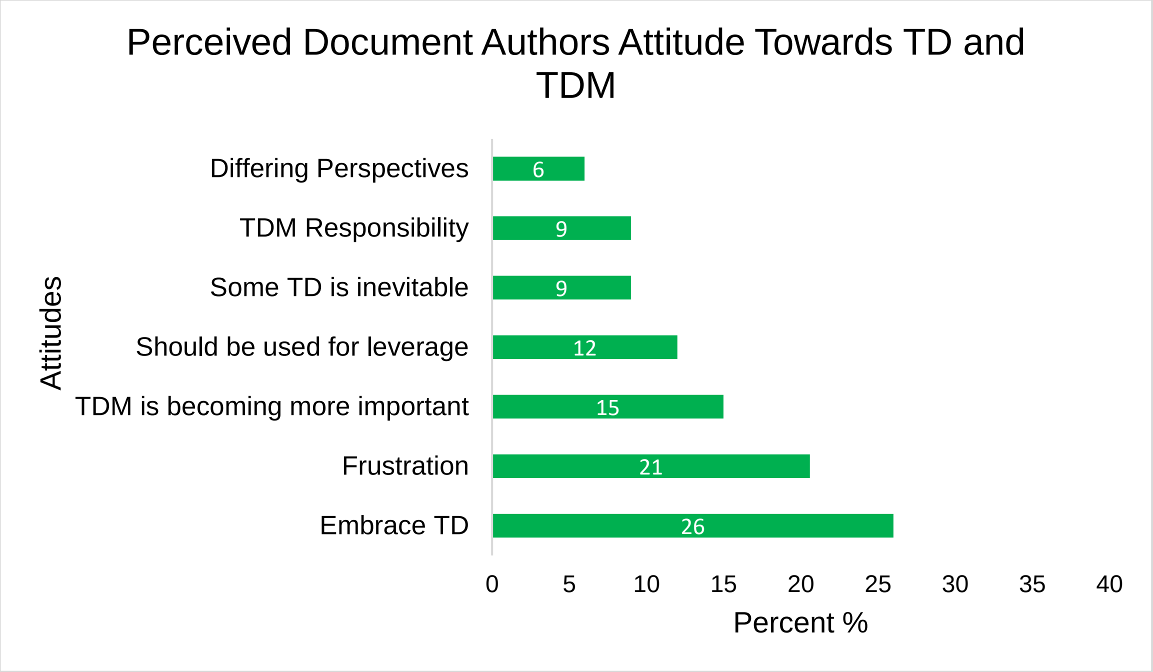


FIGURE 30 – Document Authors' Attitudes Towards TD

- TD was described by P14 as *"always to be expected and accepted in software projects".*
- Documents such as DOC29 and DOC30 state that striving for zero TD was not productive – *"Attempting to tackle the entire technical debt of a project would be prohibitively expensive and not required" and "Technical debt is a common, and often acceptable, side-effect of new projects, particularly in a fast-paced innovative environment".*
- DOC34 advocates embracing TD and describes it as *"an essential part of sustainable software development".*

4.4.3.1.3 TD Classification

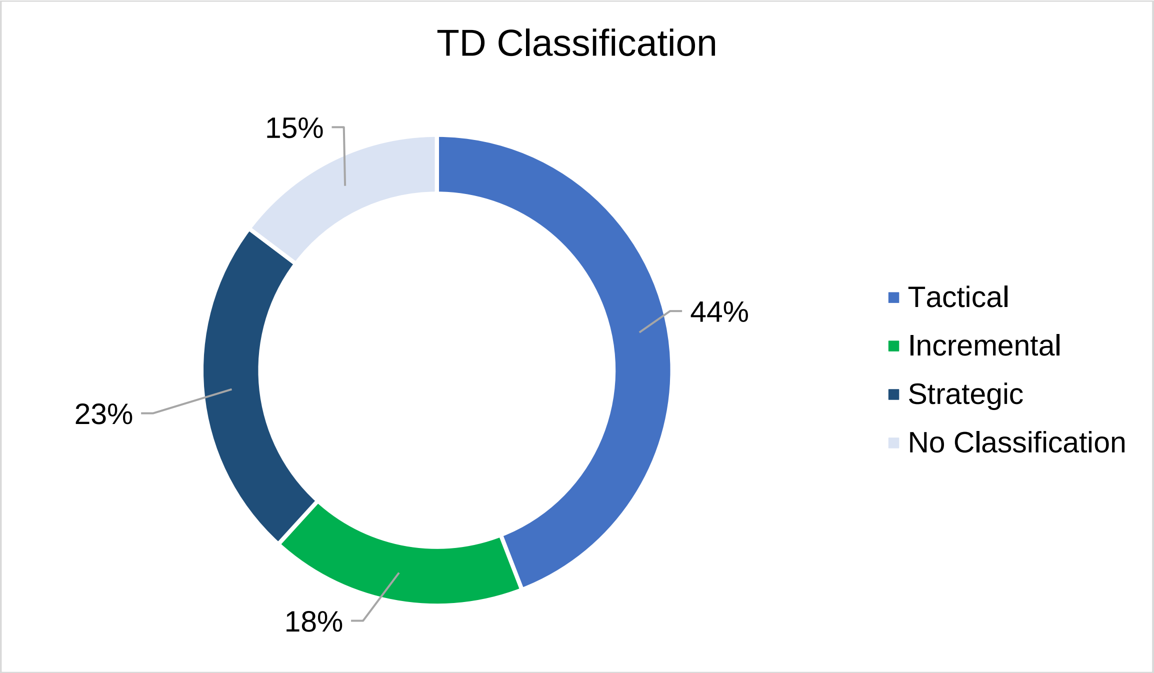


FIGURE 31 - Classification of TD Examples Outlined in Documents

- DOC28 stipulates that “many of the [source code analysis] findings should be considered as long term areas for improvement within wider development efforts rather than immediate areas of concern”
- Another document states that an *“[Application] needs to be transitioned to the strategic platform (technical debt)”.*

4.4.3.2 Organisational Factors that influence TDM (RQ2)

35% of documents analysed contained influencing factors for TDM decisions.

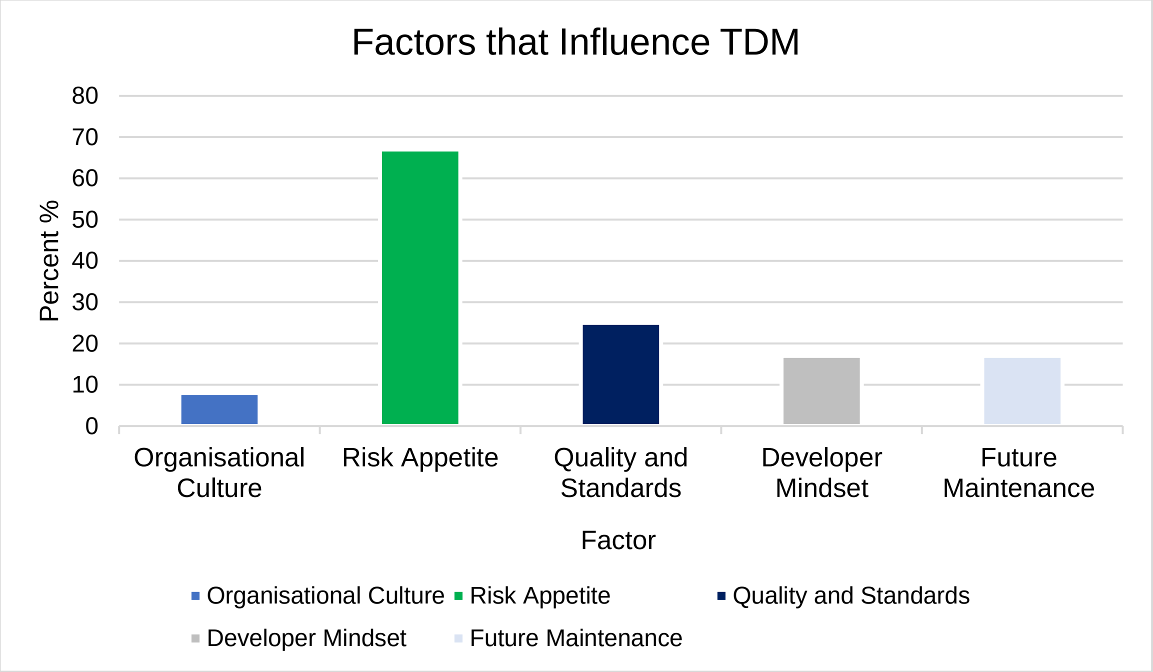


FIGURE 32 – Factors That Influence TDM

4.4.3.2.1 Organisational Culture

Organisational culture featured the least as an explicit factor for influencing TDM decisions. One document states that a core value of the studied organisation is fostering the correct mentality in various teams - *“The [organisations core values] thinking is helping to change attitudes regarding managing technical debt and forcing other teams to do likewise, improving quality across the board”.*

4.4.3.2.2 Risk Appetite

60% of the documents cited risk as a factor in TDM decision. Below are some text extracts from these documents which highlight risk appetite as an influencing factor.

- *“To accrue some debt with low-risk releases, you can quickly adapt to business requirements and user needs”.*
- *"[Release 10] de-risks the programme delivery but leaves some technical debt from an integration perspective (i.e. not all interfaces using the e-Business Suite)”.*

- *"Top risk and issues for integration: RISK: Potential risk of Introducing not easily repayable TD in [chosen solution]".*
- *"By specifically analysing and identifying where such debts will be introduced in advance, they can be planned for and managed. Risk and long-term costs are reduced, and change confidence increased".*
- *"Any decision to proceed with the deployment on the existing stack must be treated as high-risk and the resulting residual debt clearly defined and documented".*
- *"Stability in the current system is a very important consideration".*

With regards to the code complexity and area of the system that was at risk, DOC30 states that *"Some methods have a large number of independent routes through which logic can flow (referred to as 'Cyclomatic Complexity' when over 10 routes). These are difficult to debug and modify, especially as there are no unit tests, which also creates more risk".*

In addition to the above, DOC29 noted that by accruing TD there was *"Increased risk in:*

- *Determining the 'fitness' of modifications to code*
- *Assessing the knock-on effects of code changes*
- *Assuring that the expected behaviour of code is understood by the developer"*

The lack of test coverage in applications was a risk for TDM decisions as shown in the following extract from DOC17 and DOC03 respectively (See Figures 33 and 34).

- *"Straight forward development decisions are hampered by the lack of any test coverage outside of manual developer and user testing. Less than 1% of the code base is covered by unit tests, there are no automated smoke, integration or acceptance tests nor is there any evidence that thorough performance testing has been performed in the past".*
- *"Automated testing and applying software development best practices to reduce technical debt and improve technical debt management, maintainability and extensibility of the platform. The technologies are used as a matter of course for all new development"*

An extract of the SonorQube code base analysis document, DOC28, reiterates the above sentiment risk assessment and test coverage.

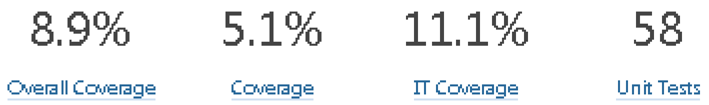

8.9% Overall Coverage
5.1% Coverage
11.1% IT Coverage
58 Unit Tests

FIGURE 33 – SONORQUBE REPORT SUMMARY

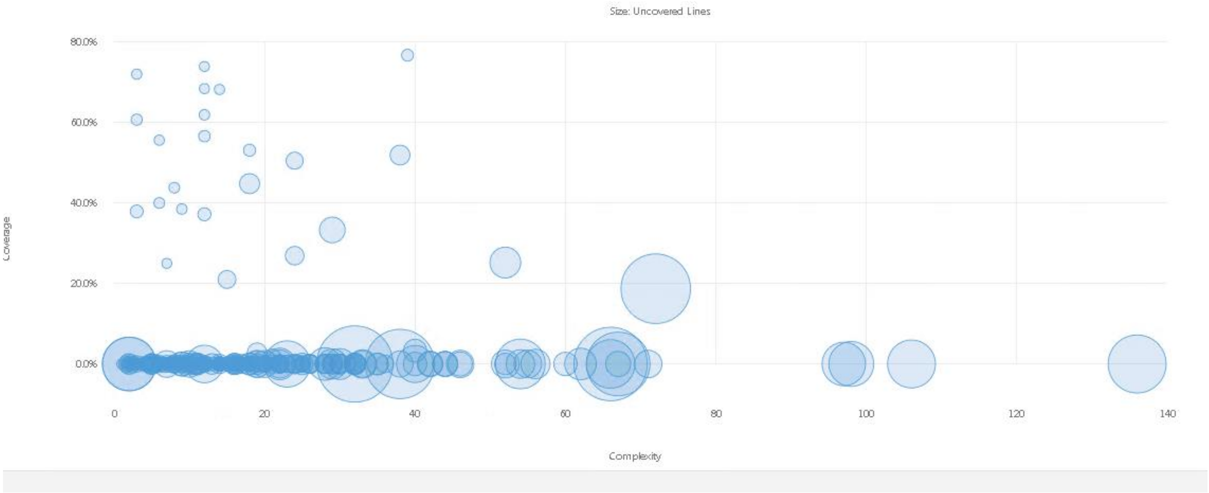


FIGURE 34 – System Automated Unit Test Coverage Diagram

The documents author states the following: *“At less than 10% the test coverage rating is quite low (only 5.1% for unit tests). The graph above indicates a number of opportunities for improvement. Given that the applications have been successfully deployed in production, the positive impact of improving test coverage must be judged against the risk, effort and cost required to retrospectively increase test coverage”.*

4.4.3.2.3 Future Maintenance

Future maintenance, re-work and refactoring were identified as factors in TDM decisions with consideration given to increased effort and consequences predicted based on the decision to intentionally accrue TD.

DOC29 for example describes *“Increased effort in:*

- *Analysing and reproducing issues;*
- *Testing changed code, as this must be done manually;*
- *Determining where to find and place new code;*

- *Finding relevant documentation; and*
- *Conducting code releases."*

Another document, DOC27, states that the potential rework necessary was an influencing factor in TDM: *"For Technical Debt, the estimates for remedial work were felt, at the time, to err on the side of caution".*

Future maintenance due to TDM decisions to incur tactical TD was outlined in DOC10: *"For tactical solutions, the business value must cover the cost of creating and removing any technical debt".*

4.4.3.2.4 Quality and Standards

25% of documents indicated that commitment to code quality was an TDM decision factor - *"Failure to change old habits leads to technical debt and eventual design death"* and is reiterated in DOC24 which states that *"Strong commitment to Code Quality – A focus on doing the right thing and reducing Technical Debt by creating common reusable libraries which other teams can reuse and share".*

4.3.3.3 TDM Techniques & Approaches (RQ3)

There were seven techniques used for TDM activities according to the analysed documents.

4.4.3.3.1 TD Items & Lists

(TD Documentation, TD Prioritisation)

DOC34 identifies a TD List as a method for prioritisation. The document describes the TD list a "living document" which is the result of the following TDM activities *"TD Identification, TD Estimation and TD Decision making".*

4.4.3.3.2 Continuous Integration & Test Driven Development

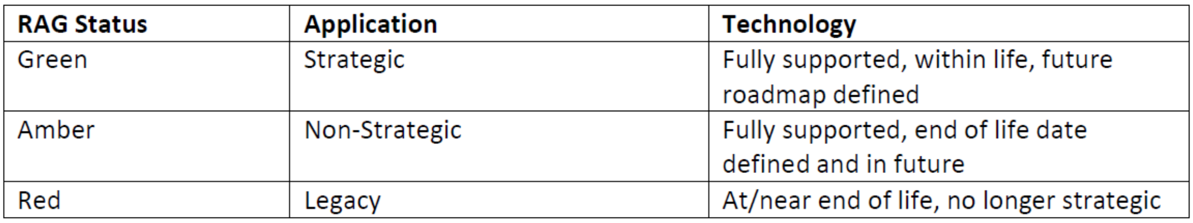
(TD Prevention)

Strategies identified for TD Prevention were continuous integration and test-driven development (TDD), with one document including a well-defined product-wide definition of done as a prevention approach. The team *"Has a product-wide definition of done to prevent technical debt. The Scrum Master can inspire the team to learn engineering practices* *associated with XP: Continuous Integration (continuous automated testing), TDD, constant merciless refactoring, pair programming, frequent check-ins, etc. Informed application of these practices prevents technical debt"* (DOC25). The change request process was also defined as an area where un-focused TD could be prevented – "*Change Request process should explicitly include consultation with Solution Architect"* (DOC30). DOC01 describes the continuous integration technique for TD prevention, *"By adopting both Continuous Integration and Continuous Deployment, you not only reduce risks and catch bugs quickly, but also move rapidly to working software".* DOC23 included the recommendation to *"Embrace Test-Driven Development and design in code production".*

4.4.3.3.3 Portfolio Approach

(TD Prioritisation)

DOC30 outlines a classification approach and states that each system component (technology and application) is given a Red, Amber, Green (RAG) status (Figure 35). The stated justification for this classification approach was that it would *"serve to manage and control technical debt".*



RAG Status	**Application**	**Technology**
Green	Strategic	Fully supported, within life, future roadmap defined
Amber	Non-Strategic	Fully supported, end of life date defined and in future
Red	Legacy	At/near end of life, no longer strategic

FIGURE 35 – RAG Status

4.4.3.3.4 Debt Service Coverage Ratio

(TD Measurement)

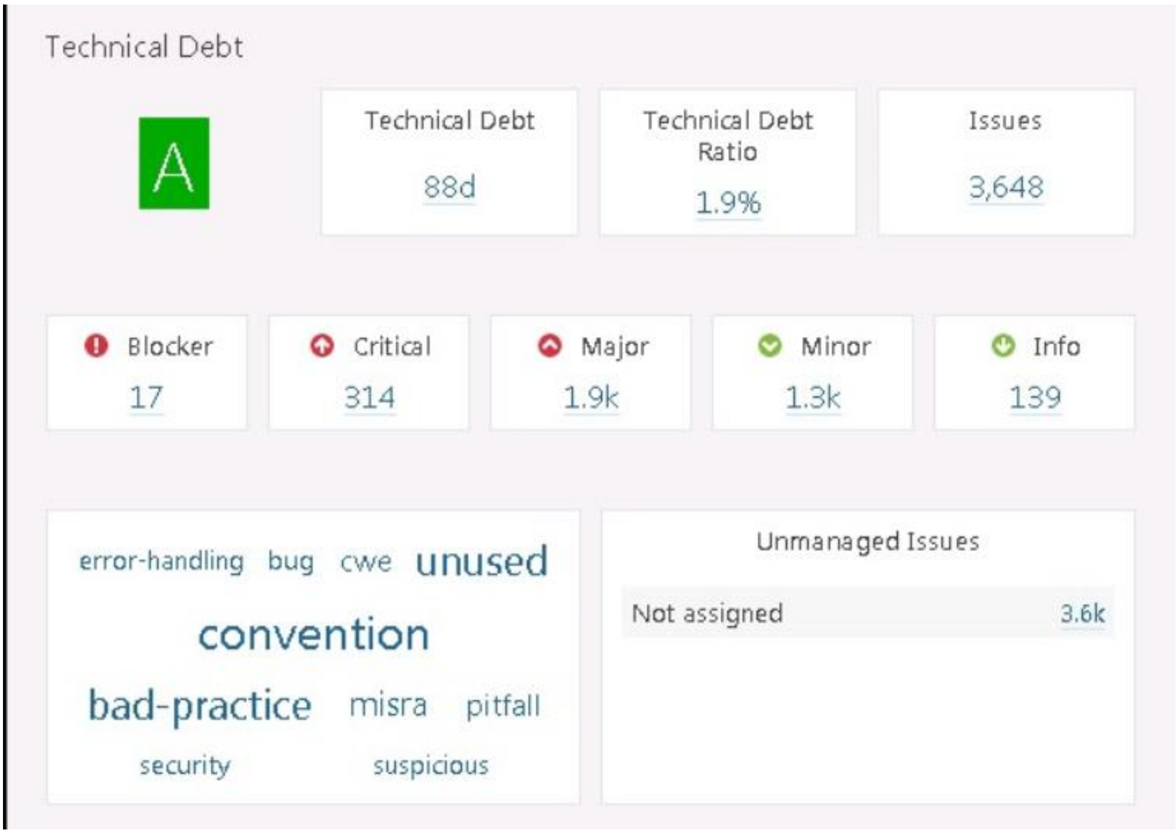
DSCR was defined as the main method in 2 documents. One TD interest measurement technique was described as the following, *"One simple way would be to ask the development team the time it has taken them to develop some functionality (say 5 days) and then ask them the time it would have taken them if working with a clean system (say 3 days). The difference is the interest paid on your technical debt which in this case, is 2 days".*

4.4.3.3.5 Source code analysis

(TD Identification, TD Monitoring, TD Communication)

In a source code analysis report (DOC28), TD is being monitored at the release stage. "*From the figures, it can be seen that recent development work has reduced the level of technical debt by 3 hrs 52 mins. Further, with the addition of some integration tests, overall code coverage, though still below 10%, has improved".*

The only source code analysis tool mentioned by all documents was SonorQube. For example, DOC29 states that *"The tool can be used to target particularly problematic areas for rework"* another highlighting that *"SonarQube provides extremely useful metrics on code quality and can highlight critical areas of weakness".* DOC30 justifies the use of TD identification tools with the following - *"By specifically analysing and identifying where such debts will be introduced in advance, they can be planned for and managed. Risk and long-term costs are reduced, and change confidence increased".* DOC28 (Fig 36) shows an example of the identification of TD in a system.



Technical Debt

A

Technical Debt 88d

Technical Debt Ratio 1.9%

Issues 3,648

Blocker 17

Critical 314

Major 1.9k

Minor 1.3k

Info 139

Unmanaged Issues

Not assigned 3.6k

FIGURE 36 – Identified TD

Expert opinion was commonly coupled with the source code analysis tool SonorQube to make TD accrual decisions – *“The analysis provided by SonarQube is highly informative and can positively influence future development decisions”.*

4.4.3.3.6 TD Register

(TD Documentation)

Integration of TD aspects to existing technical artefacts and use of TD Registers were referenced as being in use. DOC30 suggests that the TD documentation should be integrated as part of existing technical artefacts – *“An extra page of the Project Appraisal Document (PAD) deliverable should document the approach defined to manage technical debt introduced by this project”* and in another section of the same document, repeats this initiative to “*Include specific coverage of management of technical/legacy debt”.* DOC22 highlights another technique used for TD decision documentation in the form of a TD register - *“the resulting residual debt clearly defined and documented in the register”.* DOC23 advocates the use of a TD Register to document TD – *“Maintain a technical debt register which ensures awareness and helps plan how TD this can be removed”* and suggests TD should be *“included as consideration for a line item on PID document”.*

4.4.3.3.7 Continuous Service Improvement

(TD Communication)

A continuous customer service improvement initiative, retrospectives and lessons learned in agile and waterfall methodologies respectively were strategies used for TD communication.

“[Studied Organisation] introduced each of these technologies and processes as part of our commitment to Continuous Service Improvement, instrumenting inherited legacy applications with automated testing and applying software development best practices to reduce technical debt and improve maintainability and extensibility of the platform. The technologies are used as a matter of course for all new development” (DOC03).

“Retrospectives after each milestone and understanding technical debt helped eliminate waste in future iterations” (DOC12).

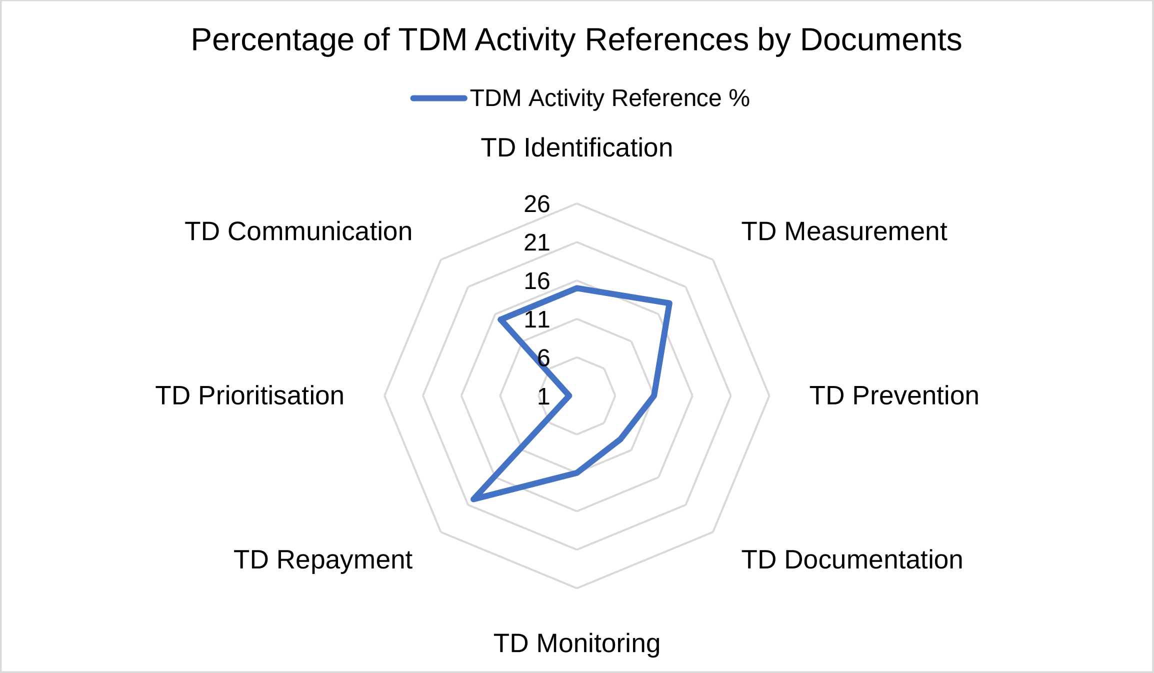


FIGURE 37 – Percentage of TDM Activity References by Documents

4.4.3.4 Challenges for TDM (RQ4)

4.4.3.4.1 Knowledge Management

DOC30 mentions that knowledge management is a challenge for TDM in one system due the requirement for significant business domain knowledge and complexity of the code base: *“The key challenges to performing adequate technical debt management for [the system] are the domain knowledge requirement, the existing technologies and the complexity of the code base”.*

4.4.3.4.2 Lack of TDM Tools

Inaccuracy of existing TDM tools was cited: *“Many of [the SonorQube Tool] findings do relate to best practices in code style which can be subjective and not necessarily indicative of issues warranting urgent action”.*

4.7 Conclusion

This chapter presented the analysis from the data collected as part of the research. The data sources used in this study were 14 semi-structured interviews. The participants were asked questions in relation to their understanding of TD, their attitudes towards TDM as well as what techniques they use to conduct TDM and what challenges TDM activities posed. Documents were also collected from the studied organisations document repository. The data collected were analysed, and initial themes and observations were noted. Finally, the emergent themes show the understanding and attitudes towards TDM, techniques for conducting TDM and the challenges it presents. Figure 38 below presents an illustrated summary of the research findings. The findings are discussed in the next chapter.

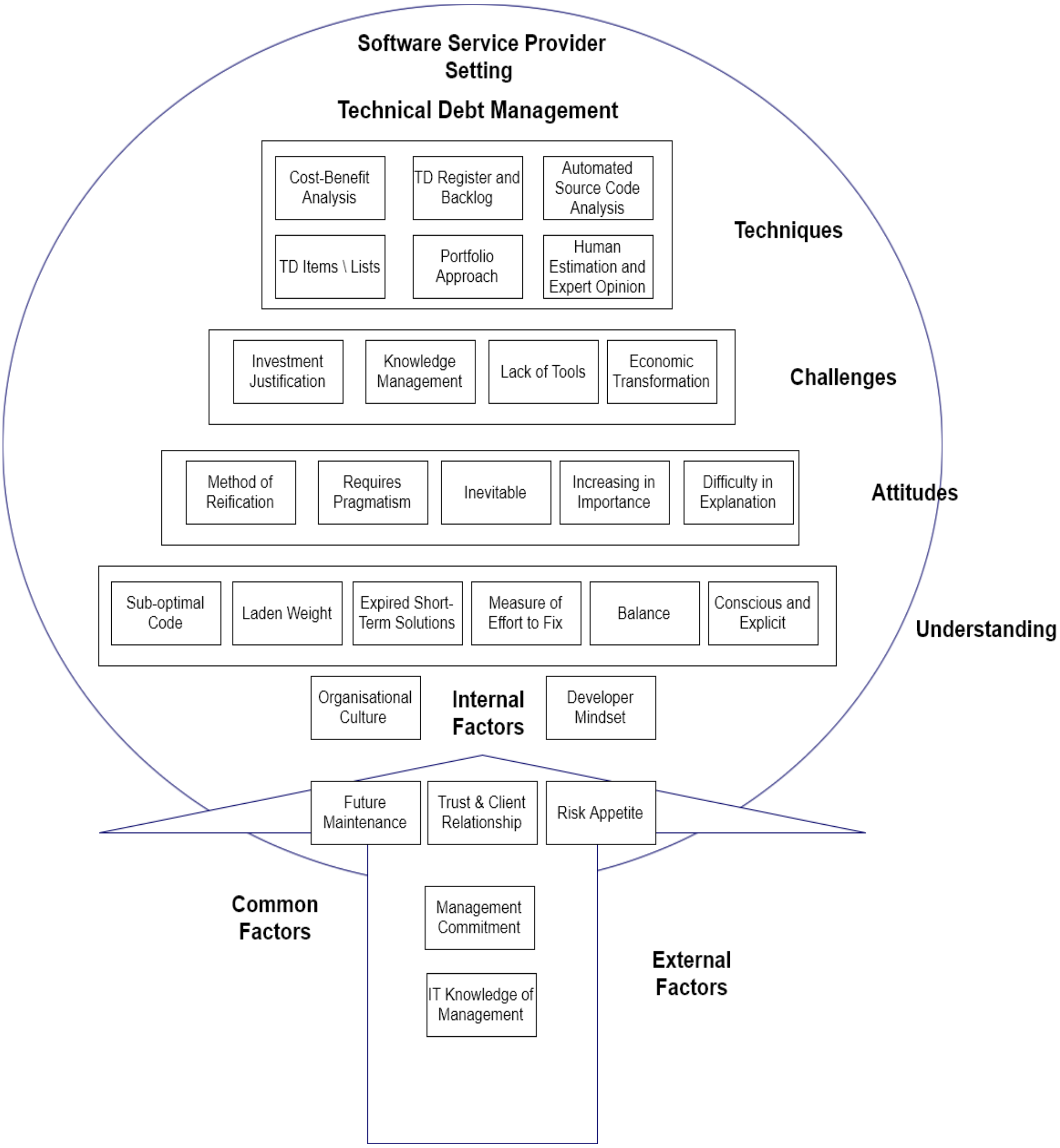


FIGURE 38 – Framework of TDM from Findings

Chapter 5: Conclusions and Future Work

5.1 Introduction

This chapter discusses how the findings of the study answered the research questions. It highlights the key findings and interesting new findings, and discusses the contribution of this research to the existing body of knowledge. Subsequent sections discuss the limitations of the research and outlines recommendations for future research in this area.

5.2 Answering the Research Question

The literature review shows that there is a lack of empirical studies on how TD is managed by software service providers in Ireland. This qualitative research focuses on analysis of how current senior software practitioners conduct TDM. To answer this question, analysis was conducted to investigate the understanding and attitudes towards TDM, factors that influence TDM, the techniques for TDM activities and the challenges that TDM poses.

Answers to research questions were obtained through the analysis of interview transcripts collected via semi-structured interviews with senior software practitioners having ten or more years of experience in software development and support projects. The findings were triangulated through analysis of documents collected from the studied organisation.

This dissertation's findings are discussed in more detail in the next section.

5.3 Research Findings and Discussion

5.3.1 RQ1. How do senior software practitioners view the concept of technical debt and its management?

Participants had differing understandings and interpretations of the TD metaphor as was originally suggested by the systematic mapping conducted by Tom et al (2013). Some participants viewed TD as a conscious and explicit decisions only whilst other participants viewed TD as being legacy issues. Others understood TD to be a quantified measurement of effort to refracture the source code. Interestingly, the reference by participants to TD as a *"laden weight"* or something that was *"dragging"* the system down has not been referenced before in the literature. TD was also understood to be a consequence of a short-term solution to achieve something that was otherwise unattainable which Tom et al (2013) describes as fundamental to the TD concept. The most common understanding of the TD term was the recognition of sub-optimal code. This is important as recognition implies conscious and explicit identification and measurement of TD issues.

93% of participants indicated that they believed attitude impacted on TDM and influenced the consultant's decisions to either personally make the decision or advise the client to accrue new TD. This corresponds with the findings of Tom et al (2013). Developers tended to be more pessimistic about the accrual of TD whereas technical team leads or non-developers were more optimistic towards intentional TD accrual. The findings suggest that TD understanding forms the basis for attitudes towards TD described next.

Generally, participants were empathetic towards the issue of TD, the need for its management and the necessity of accruing TD tactically and strategically. This finding is similar the finding by Tom et al (2013) but differs from McConnell's (2007) statement that technical staff are generally pessimistic about incurring all TD. Some participants viewed TD simply as a means to articulate a wide variety of sub-optimal code issues which does correspond to McConnell's (2007) TD suggestion that the TD term is a reifying a concept that is otherwise too intangible to be handled well.

An attitude prominent amongst participants and documents alike is the inevitability of TD and the need to focus on its adequate management as opposed to striving for zero TD. This correlates to the document analysis findings in section 4.4.3.1.2 and to the literature (Ebert, 2007; and Tom et al., 2013). The difficulty in explaining TD may be due to the participant's lack of full understanding of the concept discussed above. The need for pragmatism in accruing TD was mentioned less by participants (36%). The explanation of this finding may be due to the characteristics of the clients that the studied organisation works with. As identified by Brazier (2007), time-to-market is more important for the futures of small companies in niche markets that rely on being the first to market. This is not the case for most of the studied organisations clients which are either public sector departments or larger private organisations. Frustration towards TD issues featured prominently in the analysed documents and did not appear as evident from data collected from participants. This may be due to participants' unwillingness to disclose their frustration with TD issues during the interviews.

The perceived usefulness (or lack thereof) of explicit TDM also hinders the application of the TDM approach. Most participants (78%) believed that TDM is becoming more of an important topic in recent years. Some participants (28%) desired a more structured approach to TDM and suggested it should be manifested in the form of a TD policy as recommended by Ramasubbu et al (2015). This agrees with the view that TD should be embraced as discovered during document analysis.

The attitudes of software practitioners appear to influence the TDM challenges that are discussed next.

5.3.2 RQ2. What organisational factors influence the management of Type II TD?

Several external, internal and mixed organisational factors were presented in section 4.3.3.2 and illustrated in the TDM Framework proposed by this study (Figure 38). Risk appetite and organisational culture were the deemed the most influential organisational factors for TDM. This corresponds with the finding by Tom et al (2013) that risk appetite could influence the creation of TD and ultimately impact on project risk. It is expected that the organisations appetite for risk would influence how TD is managed. It is however, a different dynamic when one or more of an organisations software systems are being managed by a third party. Both the software system owner and the software service provider may have differing appetites for risk and levels of risk acceptance creating a potential problem for TDM. The reason for risk appetite featuring more prominently in the documents may be caused by the requirement to explicitly state risk in official documentation. The mindset of developers can influence TDM positively or negatively. This finding agrees with the findings by Yli-Hummo et al (2016).

Organisational culture and its influence on TDM decision making appears to stem mostly from the core values that are stated as the guiding principles of the studied organisation. Participants regularly cited these core values as being a consideration when conducting TDM activities and decisions. The documentation analysis corroborated this finding. The explanation for this finding may be the level of commitment of the studied organisations management to these core values or could equally indicate individual's justification for attempting to make quality TDM decisions.

The extent of managements IT knowledge also featured commonly in both data sources. The finding corresponds with the finding by Lim et al (2012) that software development knowledge strongly influenced decisions to incur TD. This finding may be linked to the understanding of TD outlined in sections 4.3.3.1 and 4.4.4.1. The finding that trust and the relationship with the client influences TDM corresponds with previous findings by Babar et al (2007) and Heiskanen et al 2008. The critical factors identified by Ova et al (2006) for maintaining trust in an established outsourcing relationship tie in with the studied organisations core values. Management IT Knowledge may influence TDM buy-in and commitment to controlling TD responsibly.

The perceived level of commitment that the management of IT exhibited influenced TDM decisions for accruing and repayment TD items in the experience of 36% of participants. As this factor was considered a factor by only 36% of interview participants, it is unclear if this factor has a significant impact on TDM. The much-cited lack of approaches and techniques for visualising TD may be linked to a difficulty in obtaining management buy-in (Fernadez-Sanchez, 2015).

The challenges that TDM poses may have a significant impact on software practitioners ability to effectively implement TDM techniques and conduct them successfully.

5.3.3 RQ3 What methods and techniques are used for explicit TDM?

Generally, participants had no pre-defined and standard TDM practice. While there was no standard accepted approach for managing TD, there were TDM activities evident within existing processes for all participants. As presented in section 4.3.3.3, there were six main techniques used for TDM activities. Most of these techniques such as source code analysis, expert opinion and TD Registration were used for multiple TDM activities. The use of one technique for multiple activities links with findings by Alves (2016).

The most commonly used techniques were for TD documentation and TD communication activities. The method of documenting TD in a TD register was mainly evident in projects where an agile methodology was being used. This suggests that the use of agile lends itself more effectively than waterfall to the continuous documentation of TD. Although TD was documented, it was not always documented as TD Items with the properties suggested by the literature (Guo et al,, 2011; 2011a; Holvitie et al., 2013, Zazworka et al., 2013). The finding that TD items are used for TD documentation less than TD Backlogs adds credence to this finding.

TD was mainly logged as “issues” and kept in the product backlog using the JIRA tool. Various participants mentioned that the TD Backlog was not regularly updated after the TD had been documented initially. This suggests an inaccurate representation of unaddressed TD issues. This finding is consistent with the suggestion by Lethbridge et al (2003) that developers’ documentation of TD may be subjective.

The popularity of expert opinion for the TD estimation and measurement activities backs up the lack of sophisticated and trusted techniques for TD measurement techniques. The results exhibited in 4.3.3.3 suggest that expert opinion is commonly used for TDM activities in collaboration with other techniques. This finding agrees with the literature (Fernadez-Sanchez, 2015) which suggests there is a need to use expert knowledge to add TD information that cannot be estimated in another way currently.

Source code analysis was the most widely used technique for TD identification and TD Measurement. The evidence that source code analysis is being actively used for TDM matches findings by Letouzey (2012; 2012a). Although the findings show that source code analysis is being used for TDM, it is recognised as being in a relatively new addition in most instances and is used in an ad-hoc manner. Where source code analysis was being used for TD identification or TD measurement, the data showed that there were plans to integrate the technique fully with release management.

Cost-benefit analysis was commonly used by participants to assist in TD repayment decisions. Although cost-benefit analysis was used, how it was used varied greatly amongst the usage instances. Mostly, the analysis used for cost-benefit was based on a rough RAG rating with not much effort being given to accurate attempts at interest and principal calculation. This finding is in line with the theory by Seaman et al (2012) that a crude estimation is still the most common one. The special considerations for cost-benefit analysis as found by (McConnell, 2007; Tom et al., 2013; Tom et al 2013) were evident. Applications that were identified by management as candidates for decommission were often in use for longer than was anticipated.

The use of a portfolio for prioritising TD was less common with 23% of participants using this technique. The literature cites the portfolio approach as one of the most cited management strategies however this study finds that it is not commonly used in practice (Alves, 2016). It is possible that the reason for this is the lack of the techniques evaluation in a real-setting. It could also be attributed to to the lack of strategic application portfolio planning within the client organisations.

Finally, attention to TDM activities between both data sources were somewhat similar with some more obvious variations (Figure 39). TD identification and TD monitoring was referenced more in the documentation than was mentioned by participants during interviews. Instances of TD communication were mentioned considerably more during interviews. The reasoning for these discrepancies may be that the semi-structured interview allowed for deeper and more verbose descriptions of TDM activities than the documents. In contrast to findings by Alves et al (2016) referenced in Table 2 of Chapter 2, TD communication was a commonly referenced activity.

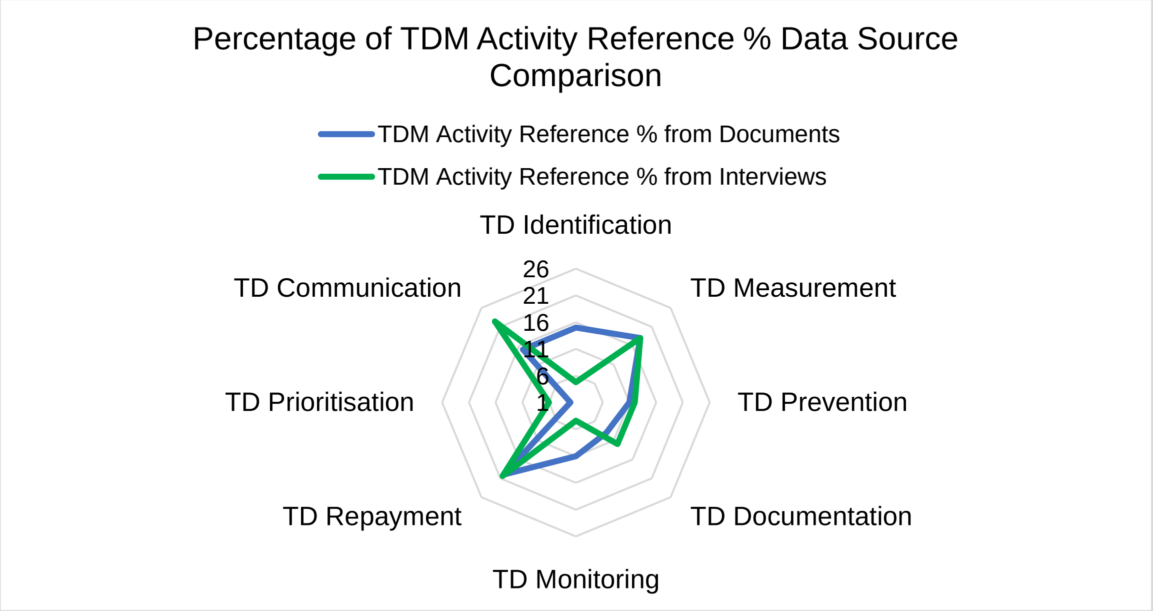


Figure 39 - Percentage of TDM Activity Reference % Data Source Comparison

5.3.4 RQ4 What challenges exist in the management of technical debt?

As presented in Section 4.3.3.4 and 4.4.3.4, this research found several current challenges to conducting effective TDM. The greatest challenge for interview participants is the justification of TDM investment. The explanation for this challenge appears to be the lack of visible results to project stakeholders which increases the difficulty in arguing its importance. The reasons offered for this challenge agree with literature which posits that TDM is time-consuming and can create additional work in the development processes (Yli-Huumo et al., 2016). There seems to be a correlation between organisational factors and the difficulty in software service providers receiving investment (time and resources) in TDM activates. The level of client management IT knowledge appears to impact on the difficulty in overcoming this challenge. This challenge was not apparent from the document analysis results. Again, the reason for this may be due to an adversity to document challenge explicitly.

The lack of available tools for TDM as a current challenge agrees with findings by Yli-Huumo et al (2016). Participants outlined the constraints of TD identification and measurement tools such as SonorQube which they claimed did not give the full picture of all technologies used by client systems such as those mentioned by Falessi et al (2013).

Organisational growth, support transitions and vendor collaboration were areas identified by participants where KT for TD was considered a challenge. As the studied organisation is in a growth phase and mostly expanding by acquisition, there is the risk that this changing and fluxing will cause valuable TD knowledge to get lost. The low use of quality TD documentation may be a contribution to this TDM challenge as found by Slinker (2008). It seems that the difficulty in overcoming developers' aversion to document TD items contributes to a large amount of tacit knowledge being retained by individuals as concluded by (Tom et al., 2013).

The results show that the there is considerable difficulty in proving business and economic value of TDM. This may be due to TD being a somewhat abstract concept. The results suggest that it is also difficult to review TDM in terms of its economic consequences as originally suggested by Falessi et al (2013) as cost of TD repayment changes depending on software release scenarios. This finding may be explained by the lack of underlying theory and models in TDM which results in challenges in real-settings.

5.4 Contribution to Research Topic

The focus of this dissertation is TD and its management in the context of an Irish software service provider. This perspective has not been explored before, along with the combination of document analysis of a studied organisation.

In addition, most of the studies analysed the TD subject through systematic literature reviews from an in-house software development team perspective and focused on aspects of TDM such as the use of a single technique for TDM. This research takes a holistic approach to the study of how TDM is conducted in a real setting.

This dissertation adds to the existing body of knowledge by investigating how TD is perceived and how it is managed by a software service provider operating in Ireland. This study proposes to fill gaps in the literature by exploring attitudes to TD, techniques used in industry for TDM and the challenges that TDM presents software practitioners.

The results found in this investigation provide an insight into how TDM is conducted in a real setting. Therefore, it may be of interest to software development and support teams within software service providers that wish to ascertain what considerations are necessary when creating a TDM strategy. It might also be useful to IT managers who are searching for techniques to manage specific TDM activities. Equally, this research could serve academics and researchers investigating the studied topic.

5.5 Limitations and Threats to Validity

This section addresses potential research validity threats and actions taken to minimize the effects. This research has at least four limitations which are outlined in the below sections.

5.5.1 External Validity – Transferability

The main limitation of this study shares the limitation of all case studies. The generalisation or external validity (Yin, 1994), of the study is restricted by the case selected. The findings outlined in chapter 4 reflect experiences in one organisation with the aim of transferability, not generalisability. Specific findings of this study may not apply in other contexts such as in-house software development teams. There is a chance that the findings of largely positive attitudes to conducting TDM may be a Hawthorne effect of the studied organisation being a provider of software services. However, none of the open-ended interview questions (see Appendix G) suggest that TDM is a worthwhile activity. The results reflect the views of software practitioners who had an interest in the issue of TDM. The study attracted participation across many teams and departments in the studied organisation.

5.5.2 Internal Validity – Credibility

There is the risk that the phenomenon described by TD might be described with different terminology that is not covered by search terms of the literature review described in Chapter 2, section 2.1. This risk was mitigated by searching the main digital libraries and conducting an initial scoping review but remains as an acknowledged validity threat.

5.5.3 Reliability – Dependability

Runeson et al (2008, p.154) describe reliability as being concerned with ‘to what extent the data and the analysis are dependent on the specific researchers’. A limitation of this study is the semi-structured interview data collection method. The reason for this is that oftentimes the questions are open-ended which results in participants’ responses varying and also generating a considerable amount of data to reduce and categorise. The results from another researcher conducting this study may produce somewhat different findings based on the data collected. The reliability of this study was improved by describing the data collection, analysis and coding processes in detail which increases repeatability for other researchers (Yin, 1994). Another limitation of this study is that only one human coder was used for content analysis (Neuendorf, 2002). The reliability of human coding which is often measured using statistical measure of intercoder reliability or ‘the amount of agreement or correspondence among two or more coders' (ibid). The reliability of this study was improved by ensuring that the classification procedures for the document analysis was consistent so as other researchers could make valid inferences from the text (Weber, 2005).

5.6 Implications for Future Research

This paper has offered an insight into how TDM is conducted. It is important to note that the research results do not show the impact of techniques used for TDM activities and what combination of techniques work most effectively. Future research in this area could lead to interesting findings on how best TDM could be implemented in organisations that develop and support software systems.

Although this research did not identify any additional TDM activities than those identified by Li et al (2015), future research which investigates these activities at a more detailed level could produce sub-activities or entirely new activities for TDM.

Finally, even though the research participants and studied documents came from a variety of software development teams operating in different industries, the relationship between particular industries and TDM were not explored. The further analysis of particular TDM techniques within a specific industry could present interesting results.

5.7 Summary

Information systems and their application to business purposes have progressed significantly since the LEO I. To counteract the first law of software evolution and at least minimise the deterioration of a systems usefulness, TDM should be practiced to control the systems decay. TDM can also help to reduce the inevitable increasing complexity of a used software system described by Lehman (1980).

This qualitative research focused on current views of TD and its management, influential TDM factors, techniques for conducting TDM activities and challenges it presents.

Findings suggest that TDM is a necessary task to control technical imperfections and where necessary leverage TD. TDM practices are ad-hoc and is being conducted in various ways and via different techniques.

This evidence from the studied organisation in isolation may reflect the growing interest in the TD phenomenon and the importance of its management for organisations which are increasingly embracing digital transformations.

References

Agile Alliance. n.d. *Technical Debt is a Systemic Problem.* [online] Available at: https://www.agilealliance.org/technical-debt-systemic-problem/#_ftn3 [Accessed 7 August 2017].

Allman, E. 2012. Managing technical debt shortcuts that save money and time today can cost you down the road, *Communications of the ACM* 55(5) 50–55.

Allman, E. 2012. Managing technical debt: shortcuts that save money and time today can cost you down the road/ *ACM Queue*. [e-journal] 10(3) Available at: http://queue.acm.org/detail.cfm?id=2168798 [Accessed 9 July 2017].

Alves, N., Mendes, T., de Mendonça, M., Spínola, R., Shull, F. and Seaman, C. 2016. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70 (C), pp.100-121.

Alves, N., Ribeiro, L., Caires, V., Mendes, T. and Spinola, R. 2014. Towards an Ontology of Terms on Technical Debt. *2014 Sixth International Workshop on Managing Technical Debt*.

American Heritage Dictionaries. 2015. American heritage dictionary of the English language. 5th ed. s.l.: Houghton Mifflin Harcourt.

Atkinson P., and Coffey A., 2002. Revisiting the relationship between participant observation and interviewing, in J F Gubrium and J A Holstein (eds) *Handbook of Interview Research* Thousand Oaks CA: Sage pp.801-14.

Avgeriou, P., Kruchten, P., Ozkaya, I. and Seaman, C. 2016. *Managing Technical Debt in Software Engineering* [pdf] Available at: http://drops.dagstuhl.de/opus/volltexte/2016/6693/pdf/dagrep_v006_i004_p110_s16162.pdf [Accessed 1 Aug. 2017].

Babar A., Verner, J. and Nguyen, P. 2007. Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *Journal of Systems and Software,* 80(9), pp.1438-1449.

Babu, M. 2016. Vendor-Driven Technical Debt: Why It Matters and What to Do About It. *Cutter IT Journal: The Journal of Information Technology Management*, 29(3), pp.33-40.

Basit, T. 2003. Manual or electronic? The role of coding in qualitative data analysis. *Educational Research.* 45(2) pp.143–54.

Bernard, H. *Research methods in anthropology: Qualitative and quantitative approaches*. 3rd Alta Mira Press; Walnut Creek, CA: 2002.

Bjørn-Andersen, N. 1985. Conference Review: *IS Research – A Doubtful Science*, In: Mumford, E, Hirschheim, RA, Fitzgerald F and Wood-Harper AT (Eds) . Research Methods in Information Systems, Proceedings of the IFIP WG 8.2 Colloquium, September 1-3 1985, Manchester Business School, Elsevier: Amsterdam

Boogerd, C. and Moonen, L. 2009. Evaluating the relation between coding standard violations and faults within and across software versions. In: *6th IEEE International Working Conference on Mining Software Repositories.* Vancouver: IEEE.

Bowen, G. A. 2009. Document analysis as a qualitative research method. *Qualitative Research Journal* [e-journal] 9(2), 27-40. http://dx.doi.org/10.3316/QRJ0902027 [Accessed 13 July 2017].

Bower, M. (2003). *Company philosophy: The way we do things around here'.* [online] McKinsey & Company. Available at: http://www.mckinsey.com/global-themes/leadership/company-philosophy-the-way-we-do-things-around-here [Accessed 19 August 2017].

Brazier, T. 2007. Managing Technical Debt. *Overload Journal,* [e-journal] 15(77). Available at: https://accu.org/index.php/articles/1301 [Accessed 20 August 2017].

Brenner, R. 2017. *Managing Technical Debt: Nine Policy Recommendations.* [online] Available at: https://www.cutter.com/article/managing-technical-debt-nine-policy-recommendations-495281 [Accessed 20 May 2017].

Brown, N, Cai, Y, Guo, Y, Kazman, R, Kim, M, Kruchten, P, Lim, E, MacCormack, A, Nord, R, Ozkaya, I, Sangwan, R, Seaman, C, Sullivan, K, and Zazworka, N., 2010. Managing technical debt in software-reliant systems, Proceedings of the FSE/SDP Workshop: *Future of Software Engineering Research*, p. 47-54 http://dx.doi.org/10.1145/1882362.1882373.

Brown, R., 2006. *Doing your dissertation in business and management: the reality of researching and writing*, Sage.

Bryman, A. 1984. The debate about quantitative and qualitative research: a question of method or epistemology? *The British Journal of Sociology*, 35, 75-9

Bryman, A. 2012. *Social research methods* 5th ed. Oxford: Oxford University Press.

Bryman, A. and Bell, E. 2015. *Business research methods.* USA: Oxford University Press.

Cai, Y, R. Kazman, C. V. Silva, L. Xiao, and H.-M. Chen. 2014. Chapter 6 – a decision-support system approach to economics-driven modularity evaluation. In *Economics-Driven Software Architecture,* I. Mistrik, R. Bahsoon, R. Kazman, and Y. Zhang, Eds. Boston: Morgan Kaufmann, 2014, pp. 105 – 128.

Campbell, T. 1996. Technology, multimedia, and qualitative research in education. *Journal of Research on Computing in Education*, 30(9), 122-133.

Cavaye, A. 1996. Case Study Research: A Multi-Faceted Research Approach for IS, *Information Systems Journal,* 6(3), pp. 227-242. http://dx.doi.org/10.1111/j.1365-2575.1996.tb00015.x.

Charmaz, K. 1983. The grounded theory method: An explication and interpretation. In *Contemporary Field Research: A Collection of Readings*, Robert M. Emerson, ed., Boston: Little, Brown and Company, 109-128.

Chiesa, M and Hobbs, S. 2008. Making sense of social research: how useful is the Hawthorne Effect*? European Journal of Social Psychology*,38(1) 67-74 http://dx.doi.org/10.1002/ejsp.401.

Clont, J. 1992. The concept of reliability as it pertains to data from qualitative studies. Paper Presented at *the annual meeting of the South West Educational Research Association.* Houston, TX.

Codabux, Z, and Williams, B. 2013. Managing technical debt: An industrial case study. *4th International Workshop On Managing Technical Debt (MTD),* p. 8

Collinsmcnicholas.ie. 2016. *The ICT Industry in Ireland 2016.* [pdf] Available at: http://www.collinsmcnicholas.ie/wp-content/uploads/2016/07/The-ICT-Industry-in-Ireland-2016.pdf [Accessed 15 Aug. 2017].

Collis, J. and Hussey, R. 2014. *Business research*. 4th ed. Basingstoke, Hampshire: Palgrave Macmillan.

Conway, M. 1968. *How do committees invent* [pdf[*Datamation*. 14(4). pp. 28-31 Available at: http://www.melconway.com/Home/pdf/committees.pdf [Accessed 5th August 2017]

Cooper, D. and Schindler, P. 2013. *Business research methods.* 12th ed. McGraw-Hill Education.

Corbin, J. and Strauss, A. 2008. *Basics of qualitative research: Techniques and procedures for developing grounded theory.* 3rd ed. Thousand Oaks, CA: Sage

Creswell, J. 2003. *Research design: Qualitative, quantitative, and mixed methods approaches*. 2nd ed. London: SAGE Publications.

Creswell, J. 2007. *Qualitative inquiry and research design: Choosing Among Five Approaches*. 2nd ed. Thousand Oaks, CA: Sage

Cunningham, W. 1992. *The WyCash portfolio management system*. [online] OOPSLA Experience Report; Available at: http://c2.com/doc/oopsla92.html [Accessed 6th June 2017]

Das, S., Lutters, W., Seaman, C., 2007. Understanding documentation value in software maintenance. In: *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology* http://dx.doi.org/10.1145/1234772.1234790.

Davies, D. and Dodd, J. 2002. *Qualitative Research and the Question of Rigor. Qualitative Health Research,* 12(2), pp. 279-289.

Denzin, N. 1988. *The research act: A theoretical introduction to sociological methods*. 3rd ed. United Kingdom: Prentice Hall.

Easterby-Smith, M, Thorpe, R and Lowe, A. 2006. *Management Research: An introduction,* 2nd Edition, Sage Publications.

Ebert, C. 2007. *Managing Technical Debt: Practical Decision-making and its business relevance.* [pdf] Available at: https://vector.com/portal/medien/vector_consulting/publications/Ebert_TechnicalDebt.pdf [Accessed 12 August 2017]

Einar, Ample Code. 2015. *Technical debt isn't technical.* [online] Available at: https://einarwh.wordpress.com/2015/12/05/technical-debt-isnt-technical/ [Accessed 5 August 2017].

Eisenhardt, K. 1989. Building theories from case study research. *Academy of Management Review,* 14(4), 532-55.

Ernst, N. 2012. On the role of requirements in understanding and managing TD. In: *Proceedings of the 3rd International Worksop on Managing Technical Debt*, pp. 61-64.

Ernst, N., Bellomo, S., Ozkaya, I., Nord, R. L., and Gorton, I. 2015 Measure It? Manage It? Ignore It? Software Practitioners and Technical Debt. In: *Proceedings of the 10th Joint*

Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 50−60. ACM, 2015.

Falessi, D, Shaw, M, Shull, F, Mullen, K, and Keymind, M. 2013, Practical considerations, challenges, and requirements of tool-support for managing technical debt. *4th International Workshop On Managing Technical Debt (MTD),* p. 16.

Fernández-Sánchez, C, Garbajosa, J, and Yague, A. 2015. A framework to aid in decision making for technical debt management, *IEEE 7Th International Workshop On Managing Technical Debt (MTD),* p. 69.

Fernández-Sánchez, C, Garbajosa, J, Vidal, C, and Yague, A. 2015., An Analysis of Techniques and Methods for Technical Debt Management: A Reflection from the Architecture Perspective, *IEEE/ACM 2nd International Workshop on Software Architecture & Metrics*, p. 22. http://dx.doi.org/10.1109/SAM.2015.11.

Fernández-Sánchez, C, Garbajosa, J, Yagüe, A, and Perez, J. 2017. Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study, *The Journal Of Systems & Software*, 124, pp. 22-38.

Flick, U. 2011. *Introducing research methodology: A beginner's guide to doing a research project.* London: Sage.

Foganholi, L., Garcia, R., Eler, D., Correia, R. and Olivete, C. 2015. *TD-Manager: a tool for managing technical debt through integrated catalog*. [pdf] Available at: http://worldcomp-proceedings.com/proc/p2015/SER2890.pdf [Accessed 7 August 2017].

Fontana, F., Roveda, R. and Zanoni, M. 2016. Technical Debt Indexes Provided by Tools: A Preliminary Discussion. *IEEE 8th International Workshop on Managing Technical Debt (MTD)*

Forward, A., Lethbridge, T., 2002. The relevance of software documentation, tools and technologies: a survey. In: *Proceedings of the 2002 ACM Symposium on Document Engineering,* DocEng '02. New York, USA. ACM, pp. 26–33.

Fowler, M. 2009. *bliki: TechnicalDebtQuadrant*. [online] Available at:
https://martinfowler.com/bliki/TechnicalDebtQuadrant.html [Accessed 21 February 2017].

Garnett, S. 2013. Technical Debt: Strategies & Tactics for Avoiding & Removing it. [Blog] Our Blogs. Available at: http://blogs.ripple-rock.com/SteveGarnett/2013/03/05/TechnicalDebtStrategiesTacticsForAvoidingRemovingIt.aspx [Accessed 20 Aug. 2017].

Garnett, S. 2013. Technical Debt: Strategies & Tactics for Avoiding & Removing it. [Blog] Our Blogs. Available at: http://blogs.ripple-rock.com/SteveGarnett/2013/03/05/TechnicalDebtStrategiesTacticsForAvoidingRemovingIt.aspx [Accessed 20 Aug. 2017].

Gat, I. 2011. *Technical Debt: Technical Debt: Assessment and Reduction*. [pdf] Available at: https://agilealliance.org/wp-content/uploads/2016/01/Technical_Debt_Workshop_Gat.pdf [Accessed 18 May 2017].

Glaser, B. and Strauss, A. 1967. *The discovery of grounded theory: strategies for qualitative research.* Chicago: Aldine Publishing Company.

Goddard, W. and Melville, S. 2004. *Research Methodology: An Introduction*, 2nd ed. Oxford: Blackwell Publishing.

Golafshani, N. 2003. *Understanding Reliability and Validity in Qualitative Research. The Qualitative Report,* 8(4), 597-606.

Google Trends. 2017. Google Trends. [online] Available at: https://trends.google.com/trends/explore?date=all&q=technical%20debt [Accessed 5 Jul. 2017].

Guaman, D., Quezada-Sarmiento, P., Barba-Guaman, L. and Enciso, L. 2017. Use of SQALE and tools for analysis and identification of code technical debt through static analysis. *12th Iberian Conference on Information Systems and Technologies*.

Guest, G., Bunce, A. and Johnson, L. 2006. *How Many Interviews Are Enough?.* Field Methods, 18(1), pp. 59-82.

Gummesson, E. 1988. *Qualitative methods in management research.* Lund, Norway: Studentlitteratur, Chartwell-Bratt.

Guo Y. and Seaman, C. 2011. Measuring and Monitoring Technical Debt, *Proceedings of the 4th International Doctoral Symposium on Empirical Software Engineering,* pp. 25–46.

Guo, Y, and Seaman, C. 2011. A portfolio approach to technical debt management In: International Conference On Software Engineering, p. 31.

Guo, Y., Spínola, R. and Seaman, C. 2014. Exploring the costs of technical debt management – a case study. *Empirical Software Engineering*, 21(1), pp. 159-182.

Hammersley, Martyn. 1992. Deconstructing the qualitative-quantitative divide. In Julia Brannen (Ed.), *Mixing methods: qualitative and quantitative research*. pp.39-55. Brookfield: Avebury.

Hartley, J. 1994. Case studies in organizational research. In: *Qualitative methods in organizational research: A practical guide*, edited by C. Cassell and G. Symon, 209–29. London: Sage.

Heintz, J. 2016. Using Technical Debt to Make Good Decisions. *Cutter IT Journal*, 29(3), p.11.

Heinz, J. 2011 From assessment to reduction: How Cutter consortium helps rein in millions of dollars in technical debt , In: *MTD Workshop* 2011.

Heiskanen, A., Newman, M. and Eklin, M. 2008. Control, trust, power, and the dynamics of information system outsourcing relationships: A process study of contractual software development. *The Journal of Strategic Information Systems*, 17(4), pp. 268-286.

Hirschheim, R. 1985. Information Systems Epistemology: An Historical Perspective. In: *Research Methods in Information Systems (IFIP 8.2 Proceedings),* Mumford, E.;Hirschheim, R.; Fitzgerald, G. & Wood-Harper, T. (eds), North-Holland, Amsterdam, pp.13 - 36

Holvitie, J. and Leppanen, V. 2013. DebtFlag: Technical debt management with a development environment integrated tool In: *4th International Workshop on Managing Technical Debt (MTD).*

IEEE Xplore Digital Library. 2017. DebtFlag: Technical debt management with a development environment integrated tool [online] Available at: http://ieeexplore.ieee.org/document/6608674/citations [Accessed 2nd August 2017].

Izurieta, C., Vetro, A., Zazworka, N., Cai, Y., Seaman, C. and Shull, F. 2012. Organizing the technical debt landscape. *2012 Third International Workshop on Managing Technical Debt (MTD)*.

Jamieson, S. 2004. Likert scales: how to (ab)use them. *Medical Education*, 38(12), pp.1217-1218.

Jarvenpa, S., Ives, B. 1991. Executive Involvement and Participation in Progressive Use of IT. *MIS Quarterly* 15(1). pp. 33-49.

Jick, T. 1979. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative Science Quarterly,* 24(December), pp. 602-61

Johnson, R., and Onwuegbuzie, A. 2004. Mixed methods research: A research paradigm whose time has come. *Educational Researcher*, 33(7), 14-26.

Kaplan Financial. 2012. The Boston Consulting Group (BCG) growth share matrix. [online] Available at: http://kfknowledgebank.kaplan.co.uk/KFKB/Wiki%20Pages/Portfolio%20analysis%20tools.aspx [Accessed 12 January 2017].

Kelle, U. 2001. Sociological Explanations between Micro and Macro and the Integration of Qualitative and Quantitative Methods. *Qualitative Social Research* [e-journal], 2(1). Available at: http://www.qualitative-research.net/fqs-texte/1-01/1-01kelle-e.htm [Accessed 8th May 2017].

Kenny, D. 1996. The design and analysis of social-interaction research. *Annual Review of Psychology,* 47, 59-86.

Klinger T., Tarr P., Wagstrom P and Williams C. 2011. An Enterprise Perspective on Technical Debt., *Proceedings of the 2nd International Workshop on Managing Technical Debt.*

Kruchten, P., Nord, R. and Ozkaya, I., 2012. Technical Debt: From Metaphor To Theory And Practice. *IEEE Software* 29(6), pp. 18-21.

Kruchten, P., Nord, R., Ozkaya, I. and Falessi, D. 2013. Technical debt. *ACM SIGSOFT Software Engineering Notes*, 38(5), p.51.

Labuschagne, A. 2003. Qualitative research: Airy fairy or fundamental? *The Qualitative Report,* 8(1), Article 7. Available from: http://www.nova.edu/ssss/QR/QR8-1/labuschagne.html [Accessed 18 August 2017].

Lash, F. 2014. *The Broken Windows Theory of Technical Debt.* [Blog] On Technical Debt. Available at: http://www.ontechnicaldebt.com/blog/the-broken-windows-theory-of-technical-debt/ [Accessed 20 Aug. 2017].

Lavington, S. 2009. *A brief history of British computers: the first 25 years (1948–1973).* [online]. British Computer Society. Available at http://www.bcs.org/content/conWebDoc/24070 [Accessed 20 August 2017].

Leedy, P. and Ormrod, J. 2005. *Practical research: Planning and design*. 8th ed. Upper Saddle River, NJ: Prentice Hall.

Lehman M., Laws of software evolution revisited. 1996. *Proc. of the 5th European Workshop on Software Process Technology,* EWSPT'96, Nancy, Fr, LNCS 1149, Springer-Verlag, pp.108-124.

Lehman, M.1980. 'On Understanding Laws, Evolution, and Conservation in the Large-Program Life Cycle'. *Journal of Systems and Software*. 1: 213–221. https://doi.org/10.1016/0164-1212(79)90022-0.

Lethbridge, T., Singer, J. and Forward, A., 2003. How software engineers use documentation: the state of the practice. *IEEE Software*. 20, 35–39.

Letouzey, J. 2012. The SQALE method for evaluating Technical Debt In: *Third International Workshop on Managing Technical Debt (MTD).*

Letouzey, J. and Ilkiewicz, M. 2012. Managing Technical Debt with the SQALE Method. *IEEE Software,* 29(6), pp.44-51.

Levin, D. M. 1988. *The opening of vision: Nihilism and the postmodern situation*. London: Routledge

Li, Z., Avgeriou, P. and Liang, P. 2015. A systematic mapping study on technical debt and its management. *Journal of Systems and Software,* 101, pp.193-220.

Liguo, Y and Mishra, A. 2013. An Empirical Study of Lehman's Laws on Software Quality Evolution. *International Journal of Software and Informatics,* 7(3) pp. 469-481.

Lim, E., Taksande, N., Seaman, C., 2012. A balancing act: what software practitioners have to say about technical debt. *IEEE Software*. 29, 22–27.

Lincoln, Y. and Guba, E. G. 1985. *Naturalistic inquiry*. Beverly Hills, CA: Sage.

Maldonado, E. and Shihab, E. 2015. Detecting and quantifying different types of self-admitted technical Debt. *IEEE 7th International Workshop on Managing Technical Debt (MTD).*

Martin, R. 2009. A Mess is not a Technical Debt. - Clean Coder. [online] Available at: https://sites.google.com/site/unclebobconsultingllc/a-mess-is-not-a-technical-debt [Accessed 5 Mar. 2017].

Maxwell, J. 2005. *Qualitative research design: An interactive approach*. 2nd ed. Thousand Oaks, CA: Sage Publication.

May, T. 2011. *Social research: Issues, methods and research*. London: McGraw-Hill International.

McCabe, T. 1976. *A Complexity Measure*. [pdf] Available at: http://www.literateprogramming.com/mccabe.pdf [Accessed 5 August 2017].

McConnell, S. 2007*. Technical Debt*. [blog] Available at: http://www.construx.com/10x_Software_Development/Technical_Debt/ [Accessed on 4 April 2017].

McConnell, S. 2013. *Managing Technical Debt*. [pdf] Available at: http://2013.icse-conferences.org/documents/publicity/MTD-WS-McConnell-slides.pdf [Accessed 5 April 2017].

McCormick, R. and James, M. 1988. *Curriculum evaluation in schools.* 2nd ed. London: Routledge.

McFarlan F. 1984. Information Technology changes the way you compete. *Harvard Business Review*, May-June, pp. 99-103.

Merriam, S. and Tisdell, E. 2015. *Qualitative research: A guide to design and implementation.* United States: John Wiley & Sons

Miles, M., and Huberman, A. 1994. *Qualitative data analysis*. 2nd ed. London: Sage

Mingers, J. 2001. “Combining IS Research Methods: Towards a Pluralist Methodology,” Information Systems Research (12:3), pp. 240-259

Mishler, E. 2000. Validation in inquiry-guided research: The role of exemplars in narrative studies. In: B. M. Brizuela, J. P. Stewart, R. G. Carrillo, & J. G. Berger (Eds.), *Acts of inquiry in qualitative research.* pp.119-146. Cambridge, MA: Harvard Educational Review.

Neuendorf, A. 2002. The Content Analysis Guidebook. Thousand Oaks, CA: Sage. p. 10.

Neuman, L.W. 2000. *Social research methods: Qualitative and quantitative approaches*. Harlow: Pearson

Nord, R. 2016. *The Future of Managing Technical Debt*. [online] Software Engineering Institute. Available at: https://insights.sei.cmu.edu/sei_blog/2016/08/the-future-of-managing-technical-debt.html [Accessed 21 Feb. 2017].

Norton, M. 2009. Doc on Dev: Messy Code is not Technical Debt. [online] Available at http://docondev.com/blog/2009/08/messy-code-is-not-technical-debt [Accessed 12 July 2017].

O'Leary, Z. (2014). *The essential guide to doing your research project*. 2nd ed. SAGE Publications Ltd

Oza, N., Hall, T., Rainer, A. and Grey, S. 2006. Trust in software outsourcing relationships: An empirical investigation of Indian software companies. *Information and Software Technology*, 48(5), pp. 345-354.

Patton, Eric, and Appelbaum, S. 2003. The case for case studies in management research. *Management Research News,* 26(5), pp. 60-71.

Paulk, M., Curtis, B., Chrissis, M. and Weber, C. 1993. Capability maturity model, version 1.1. *IEEE Software,* 10(4), pp. 18-27.

Per.gov.ie. (2016). *The Department of Social Expenditure and Reform* [online] Available at: http://www.per.gov.ie/en/ [Accessed 17 January 2017].

Podsakoff, P., MacKenzie, S., Lee, Y. 2003. Common methods biases in behavioural research. *Journal of Applied Psychology*, 88: pp. 879-903

Power, K. 2013. Understanding the impact of technical debt on the capacity and velocity of teams and organizations: viewing team and organization capacity as a portfolio of real options. In: *4th International Workshop on Managing Technical Debt (MTD),* pp. 28–31.

Provalis Research (2016). Montreal: QDA Miner Lite.

Ramakrishnan, S. 2013. *Managing Technical Debt.* [online] Available at: https://www.scrumalliance.org/community/articles/2013/july/managing-technical-debt [Accessed 20 August 2017].

Ramasubbu, N, Kemerer, C, and Woodard, C. 2015. Managing Technical Debt: Insights from Recent Empirical Evidence, *IEEE Software. 32(2)*. pp. 22-25.

Ramasubbu, N. and Kemerer, C. 2013. Towards a model for optimizing technical debt in software products. *4th International Workshop on Managing Technical Debt (MTD).*

Rapley, T. 2007. *Doing conversation, discourse and document analysis.* London: Sage

Reddy, R. 2016. Addressing the Hidden Obstacles to Innovation and Digital Disruption, *Cutter IT Journal.* 29(3), p. 28.

Remenyi, D., Williams,B., Money, A. and Swartz, E. 2003. *Doing Research in Business and Management: An Introduction to Process and Method,* London: Sage publications

Riel, Arthur J. 1996. Chapter 3: Topologies of Action-Oriented Vs. Object-Oriented Applications. *Object-Oriented Design Heuristics*. Boston, Massachusetts: Addison-Wesley.

Ritchie, J. and Lewis, J. 2003. *Qualitative research practice: A guide for social science students and researchers*; Ed. By Jane Ritchie. London: Sage Publications.

Robinson, W. S. 1950. Ecological correlations and the behaviour of individuals. *American Sociological Review,* 15, 351-357.

Robson, C. 2002. *Real World Research: A Resource for Social Scientists and Practitioner-Researchers,* Second Edition, Oxford: Blackwell.

Rooney, D. 2010. Technical debt : challenging the metaphor, *Cutter IT Journal* 23(10), pp. 16–18.

Runeson, P. and Höst, M. 2009. *Empire Software Eng*. [e-journal] 14(131). https://doi.org/10.1007/s10664-008-9102-8.

Sabherwal, R. 1999. The role of trust in outsourced IS development projects. *Communications of the ACM*, 42(2), pp .80-86.

Saunders, M., Lewis, P. and Thornhill, A. 2007. *Research methods for business students.* Harlow, England: Financial Times/Prentice Hall.

Saunders, M., Lewis, P. and Thornhill, A. 2012. *Research Methods for Business Students* 6th edition, Pearson Education Limited.

Schmid, K. 2013. On the limits of the technical debt metaphor some guidance on going beyond, in : *Proceedings of the 4th International Workshop on Managing Technical Debt (MTD'13),* IEEE, San Francisco, CA, USA, 2013, pp.63–66.

Seale, C. 1999. Quality in qualitative research. *Qualitative Inquiry,* 5(4), pp. 465-478.

Seaman, C., Guo, Y., Zazworka, N., Shull, F., Izurieta, C., Cai, Y. and Vetro, A. 2012. Using technical debt data in decision making: Potential decision approaches In: *3rd International Workshop On Managing Technical Debt (MTD),* p. 45

Sharma, T., Suryanarayana, G. and Samarthyam, G. 2015. *Refactoring for Software Design Smells: Managing Technical Debt.* Morgan Kaufmann Publishers, pp.207-208.

Siebra, C., Oliveira, R., Seaman, C., Silva, F. and Santos, A. 2016. Theoretical conceptualization of TD: A practical perspective. *The Journal of Systems & Software,* 120, pp. 219-237.

Silverman, D. 2007. Reply to Charles Briggs: Anthropology, interviewing and communicability. *Current Anthropology* 48(4) pp. 572–573.

Snipes, W, Robinson, B, Guo, Y, & Seaman, C (2012), 'Defining the decision factors for managing defects: A technical debt perspective', 2012 Third International Workshop On Managing Technical Debt (MTD), p. 54, Publisher Provided Full Text Searching File, EBSCOhost, viewed 4 March 2017

Spradley J. 1979. *The ethnographic interview*. Holt, Rinehart & Winston; New York.

Stake, R. E. 1995. *The art of case study research.* Thousand Oaks, CA: Sage

Stenbacka, C. 2001. *Qualitative research requires quality concepts of its own. Management Decision,* 39(7), pp. 551-555.

Strauss, A. and Corbin, J. 1990. *Basics of qualitative research: Grounded theory procedures and techniques.* Newbury Park, CA: Sage Publications, Inc.

Taksande, N. 2011. *Empirical Study on Technical Debt as viewed by Software Practitioners.* Masters. University of Maryland.

Tashakkori, A. and Teddlie, C. 1998. *Mixed methodology: Combining qualitative and quantitative approaches.* Thousand Oaks: Sage.

Taylor, C. 2017. *Irish businesses rank highly for tech spend, survey shows.* [online] The Irish Times. Available at: https://www.irishtimes.com/business/technology/irish-businesses-rank-highly-for-tech-spend-survey-shows-1.3141008 [Accessed 15 August 2017].

The Linux Information Project. 2004. *Source code definition by The Linux Information Project.* [online] Available at: http://www.linfo.org/source_code.html [Accessed 13 Aug. 2017].

Theodoropoulos, T., Hofberg M., Kern, D. 2011, TD From the Stakeholder Perspective, 2nd International Workshop On Managing Technical Debt (MTD), p. 8, Publisher Provided Full Text Searching File, EBSCOhost, viewed 5th March 2017.

Tom, E., Aurum, A. and Vidgen, R. 2013. An exploration of technical debt, *The Journal Of Systems & Software*, 86, pp. 1498-1516.

Tonella, P., Torchiano, M., Du Bois, B. and Systa, T. 2007, Empirical studies in reverse engineering: state of the art and future trends, *Empirical Software Engineering,* 12(5), pp. 551-571.

Transcribe. (n.d.). Mixpanel Mobile Analytics [Software]. Available from https://transcribe.wreally.com/.

Trochim, W. 2006. *Introduction to Validity*. [online] Available at: https://www.socialresearchmethods.net/kb/introval.php [Accessed 14 May 2017].

Walliman, N. 2011. *Your research project: Designing and planning your work*. Sage Publications.

Ward, J. 1987. *Information Systems & Technology Application Portfolio Management - An assessment of matrix based analyses.* Cranfield [Report] / Cranfield school of Management, Cranfield University.

Weber, R. (2005). Basic content analysis. 2nd ed. Newbury Park [etc.]: Sage Publications, p.12.

West, D. 2015 *What happens if robots take the jobs? The impact of emerging technologies on employment and public policy* [pdf] Available at: https://www.brookings.edu/wp-content/uploads/2016/06/robotwork.pdf [Accessed 20 August 2017].

Wiegers, K. 1996. *Creating a software engineering culture*. S.l.: Dorset House Publishing C.

Wilson, J. and Kelling, G. 1982, Broken Windows: The police and neighbourhood safety, *The Atlantic, (*Manhattan institute).

Yin, R. 1989. *Case study research: Design and methods.* Applied Social Research Series, Vol. 5. London: Sage.

Yin, R. 2003. *Case Study Research: Design and Methods,* Sage Publications , Thousand Oaks, California.

Yin, R. 2009. *Case Study Research, Design and Methods.* Sage Publications, Newbury, CA.

Yli-Huumo, J, Maglyas, A and Smolander, K. 2014. *The Sources and Approaches to Management of Technical Debt: A Case Study of Two Product Lines in a Middle-Size Finnish Software Company,* Product-Focused Software Process Improvement: 15th International Conference, PROFES 2014, Helsinki, Finland, December 10-12, 2014. Proceedings, p. 93. Available from: 10.1007/978-3-319-13835-0_7. [Accessed: 20 August 2017].

Yli-Huumo, J., Maglyas, A. and Smolander, K. 2016. How do software development teams manage technical debt? – An empirical study. *Journal of Systems and Software*, 120, pp.195-218. https://doi.org/10.1016/j.jss.2016.05.018.

Yli-Huumo, J., Maglyas, A., and Smolander, K. 2015. The Benefits and Consequences of Workarounds in Software Development Projects, *Software Business: 6th International Conference.* ICSOB 2015, Braga, Portugal, June 10-12, 2015, Proceedings, p. 1. https://doi.org/10.1007/978-3-319-19593-3_1.

Zazworka N., Spinola. R., Vetro, A., Shull, F., Seaman., C. A case study one effectively identifying technical debt, In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE'13),* ACM, Portode Galinhas, Brazil, 2013, pp. 42–47.

Appendices

Appendix A - Role and experience of interview participants

Interviewee ID	Role(s)	Experience in the organisation	Total Experience in IT
P1	IT Business Analyst	1 year	12 years
P2	Software Architect & Team Technical Lead	2 years	16 years
P3	Java Developer & Functional Consultant	2 years	10 years
P4	Infrastructure Project Manager	6 years	13 years
P5	Senior Systems Analyst	3 years	18 years
P6	ERP Support Consultant	2 years	13 years
P7	Technical Team Lead	7 years	10 years
P8	Technical Team Lead	4 years	14 years
P9	Technical Team Lead	2 years	12 years
P10	Java Solution Architect	12 years	12 years
P11	Senior Java Developer	5 years	12 years
P12	Senior Database Developer	3 years	21 years
P13	Technical Team Lead	5 years	14 years
P14	Account Manager	7 years	12 years

Appendix B - Interview participants across roles and years’ experience

Role	Total Experience in IT			Total
	10	**11 - 15**	**16 +**	
Senior IT Business Analyst	-	1	-	1
Java Solution Architect & Technical Team Lead	-	1	1	2
Project Manager	-	1	-	1
Senior Systems Analyst\Developer	1	1	2	4
ERP Support Consultant	1	-	-	1
Technical Team Lead	1	3	-	4
Account Manager	-	1	-	1
Total	3	8	3	14

Appendix C - Participants Full TD Description

Participant ID	**Technical Debt Definition**
P1	*"It’s a decision to make not such a good technical choice, that may impact, or, probably will impact the end user or the application further down the line and will kind of involve some rework in getting that back up to scratch."*
P2	*"Technical debt is like a measure of the effort that you need to undertake to fix the code as it is right now. It is something that you have been dragging along by all the different bad decisions or design decisions that you have taken at first and how those decisions are actually dragged out and bring issues in the future."*
P3	*"Technical debt is known issues that are kind of sitting in the back log waiting to be dealt with and haven’t been assigned to someone and they haven’t been looked at yet or if they have been looked at they haven’t been resolved."*
P4	*"It's a balance between developing new functionality and keeping everything else working so in order to, so you are making pragmatic decisions sometimes to deliver within very tight deadlines and you do something that might not be proper project standards."*
P5	*"It's where you can't move ahead with something because of the amount of resources necessary to build a longer lasting solution. Where you end up hacking something to make it work or allowing it to work and quite often that's fine and oftentimes that's fine as the system is coming to the end of it's life anyway and there is no point investing further time in it when something is working, there is no point in rewriting it completely."*
P6	*"Technical debt is making a change now for a customer to get them through a problem or an issue but in the long term it is not the best approach and it may cost the customer more money in the long run if they don’t actually fix the issue properly."*
P7	*"Technical debt is when you recognise that something in an application is not as good as it could be - it is not causing any business critical issues or it is causing some issues the business is willing to accept the risk and you agree with the business that you are not going to make the changes right now but that you agree that at some time in the future that those issues are going to have to be addressed”.*

P8	*"Technical debt comes out of shortcuts that you take where you need to get something done or usually on a timeline or because of technical difficulties with the right way of doing something you take a shortcut and go a different way. So, with taking a shortcut it might even be a deliberate design decision up front where someone says 'I want you to do it like this' and you know that is not the right way to do it and you stick it on a list of things you would like to do in an ideal world".*
P9	*"It is what makes an exact science un-exact".*
P10	*"Technical debt is the short-term decision that a team takes that eventually accumulates in additional complexity in software systems."*
P11	*"It is a concept that happens when due to poor requirements or a lack of an understanding in the process as well as business pressures especially in IT teams that you have to make a decision to do the easiest way, the fastest way without taking into account the implications of that and what will happen in the future if you do this or it will probably be good for the short term but in the long run this type of decision will take a toll on the project".*
P12	*"It is anything that affects the quality of our code"*
P13	*"It is technology, or aspects of what you're working on that are not quite right and maybe they are holding you back or holding the systems evolution back ... things that probably should be improved on but often aren't".*
P14	*"It is the culmination of the decisions that developers make to do less than optimal practices as they are developing that then affects your ability to enhance or extend a system or maintain the system at a later date".*

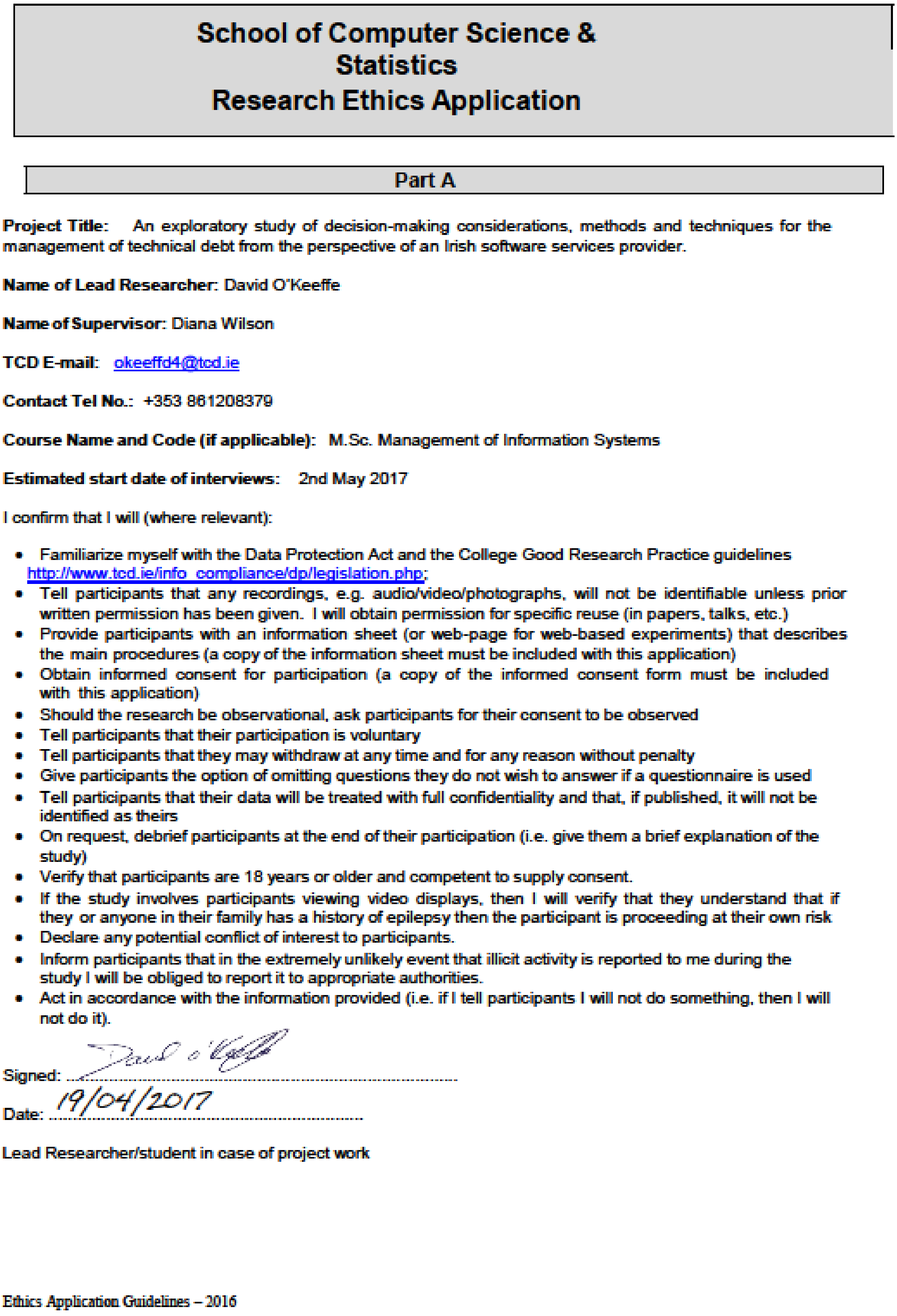
Appendix D - Analogy to describe technical debt

Participant	Text
P1	*‘Carrying additional time and effort forward from one sprint to the other which is a wall of debt’.*
P12	*‘It is like buying a load of things on your credit card and just letting the interest accrue. You got to come back and fix it up. You can't just let it get out of control’.*
P14	*‘It's like trying to change a tyre on a car when it's driving down the motorway at 150km!’.*
P3	*‘It's like the ‘Buy cheap, buy twice advice’.*
P4	*‘If you were running oil rigs and things like that you would be obliged to tell the shareholders about the upkeep and maintenance and the various risks and everything’.*
P8	*‘If somebody said I'm going to throw junk on to the roof of Harrods and I'm going to keep doing that and one day the roof falls in and Harrods must close for a few months they use millions and millions, they would never do that! Because that is not a sensible thing to do and yet businesses because software is abstract and intangible it’s something that runs and people just look at the surface’.*
P9	*‘Students taking out student loans so that they can go to college and maybe earn more in the future’.*

Appendix E – Transcribed word count by participant

Participant	Transcribed Word Count	Obfuscated Word Count
P1	3151	3065
P2	3459	3404
P3	3819	3761
P4	7265	7215
P5	4819	4798
P6	5517	5484
P7	3910	3879
P8	5953	5933
P9	5975	5962
P10	5436	5412
P11	5445	5402
P12	6251	6223
P13	2076	2054
P14	3815	3791
Total	66891	66383
Average	4778	4742

Appendix F - Ethics Application and Supporting Documentation



School of Computer Science & Statistics
Research Ethics Application

Part A

Project Title: An exploratory study of decision-making considerations, methods and techniques for the management of technical debt from the perspective of an Irish software services provider.

Name of Lead Researcher: David O'Keeffe

Name of Supervisor: Diana Wilson

TCD E-mail: okeeffd4@tcd.ie

Contact Tel No.: +353 861208379

Course Name and Code (if applicable): M.Sc. Management of Information Systems

Estimated start date of interviews: 2nd May 2017

I confirm that I will (where relevant):

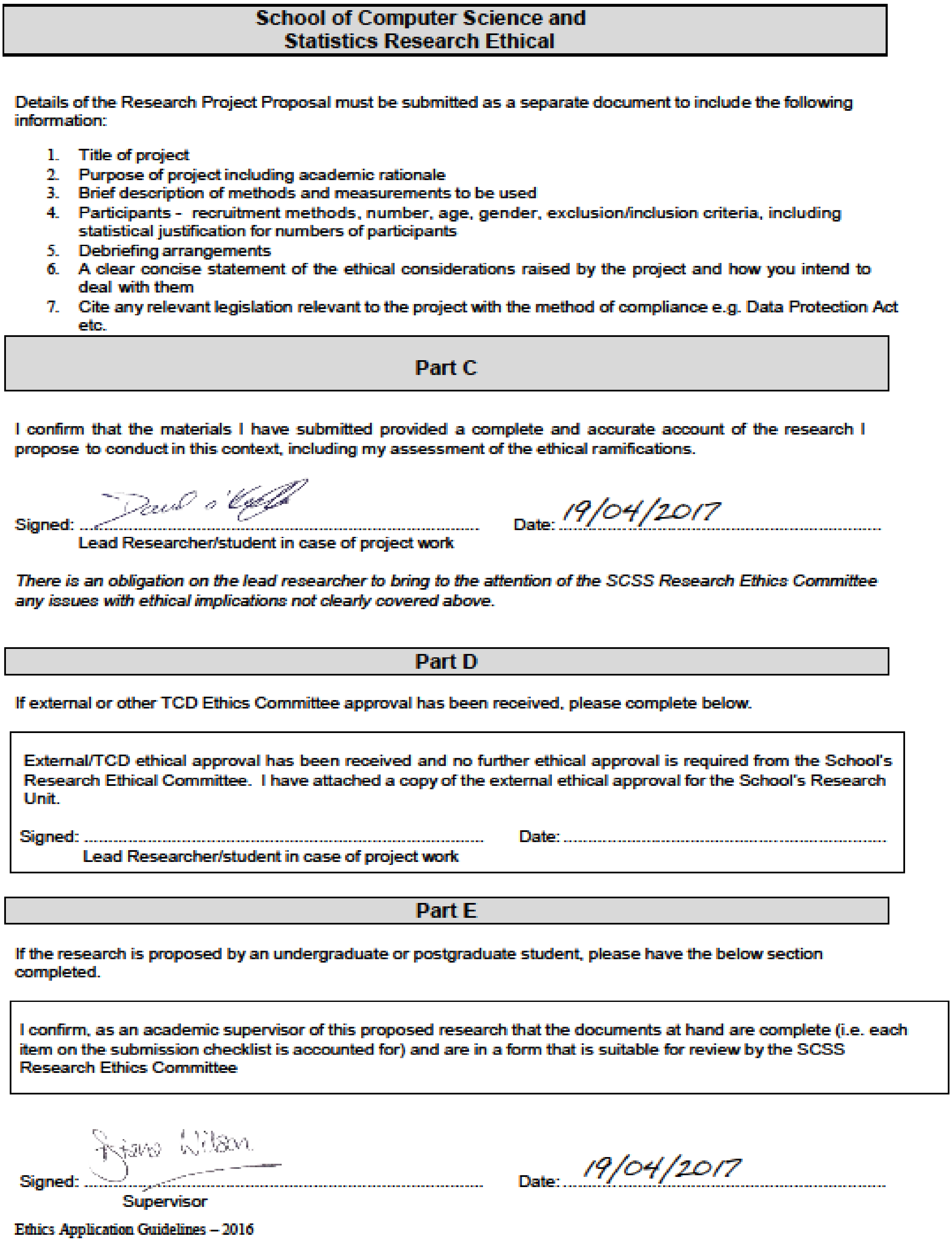
- Familiarize myself with the Data Protection Act and the College Good Research Practice guidelines http://www.tcd.ie/info_compliance/dp/legislation.php;
- Tell participants that any recordings, e.g. audio/video/photographs, will not be identifiable unless prior written permission has been given. I will 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: ....

Date: 19/04/2017 ....

Lead Researcher/student in case of project work

Ethics Application Guidelines – 2016



School of Computer Science and Statistics Research Ethical

Details of the Research Project Proposal must be submitted as a separate document to include the following information:

1. Title of project
2. Purpose of project including academic rationale
3. Brief description of methods and measurements to be used
4. Participants - recruitment methods, number, age, gender, exclusion/inclusion criteria, including statistical justification for numbers of participants
5. Debriefing arrangements
6. A clear concise statement of the ethical considerations raised by the project and how you intend to deal with them
7. Cite any relevant legislation relevant to the project with the method of compliance e.g. Data Protection Act etc.

Part C

I confirm that the materials I have submitted provided a complete and accurate account of the research I propose to conduct in this context, including my assessment of the ethical ramifications.

Signed: .... Date: 19/04/2017 ....
Lead Researcher/student in case of project work

There is an obligation on the lead researcher to bring to the attention of the SCSS Research Ethics Committee any issues with ethical implications not clearly covered above.

Part D

If external or other TCD Ethics Committee approval has been received, please complete below.

External/TCD ethical approval has been received and no further ethical approval is required from the School's Research Ethical Committee. I have attached a copy of the external ethical approval for the School's Research Unit.

Signed: .... Date: ....
Lead Researcher/student in case of project work

Part E

If the research is proposed by an undergraduate or postgraduate student, please have the below section completed.

I confirm, as an academic supervisor of this proposed research that the documents at hand are complete (i.e. each item on the submission checklist is accounted for) and are in a form that is suitable for review by the SCSS Research Ethics Committee

Signed: Fiona Wilson .... Date: 19/04/2017 ....
Supervisor

Ethics Application Guidelines – 2016

Appendix G – Interview Questions

Introduction

- What is your current position or role?
- How many years' experience do you have in the IT Industry? How long have you been with [studied organisation]?
- What have been the types of responsibilities of the positions you've held that relate to software development and\or support?
- Could you briefly describe your project experiences related to industry including the types of software development methodologies used?
- What has been the largest release or project in terms of duration or effort that you have been involved in?

Main Interview

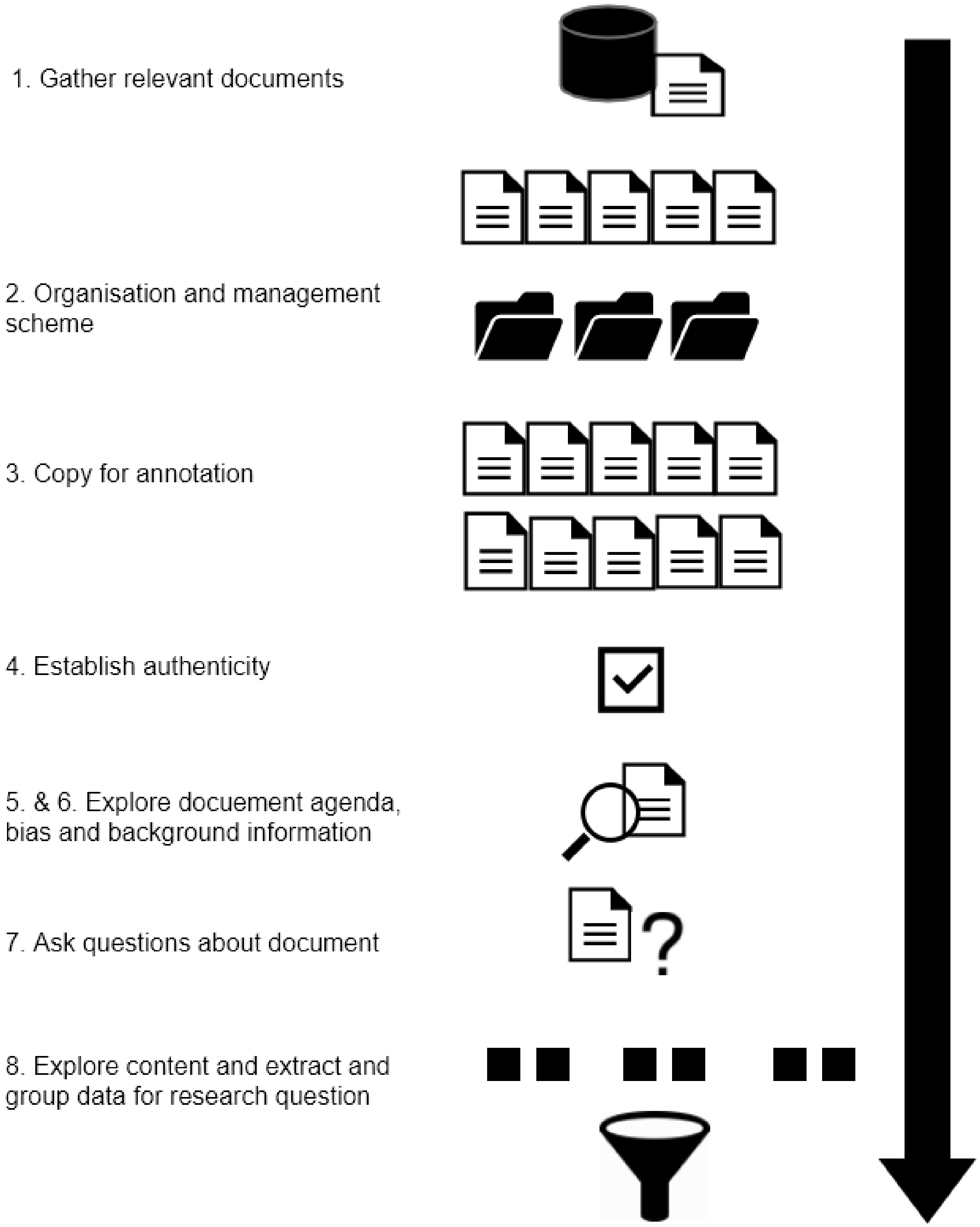
- How would you define technical debt?
- Have you experienced situations where you had to take shortcuts in your projects for some reason and decided to fix them later?
    - If yes, could you give any examples of intentional technical debt in your current project?
        - Why was the intentional technical debt incurred in the mentioned instance?
- What factors do you normally consider when you decide to incur technical debt?
    - How are these factors weighted? Are there any factors more important to the team or the client than others? If so, why is this?
- Has the decision to defer a piece of maintenance ever had any positive or negative impact?
    - Did you learn anything from these examples? Would you take the same shortcuts again? Why or why not?
- Is there anything that impacts on fixing the TD later?
- Have you ever taken shortcuts in development because of pressure from business people or a customer due to dead- lines? Or from other external third parties?
- Have you ever been “forced” to take shortcuts in a situation where business people did not necessarily understand the concept of technical debt and its effects on the project, and you thought it was a bad idea?
- Does the software development methodology being used have an impact on technical debt decisions? Agile or waterfall etc.

- Do you think third party software services providers play a part in technical debt management?
    - How?
- How do you\your team make decisions regarding taking on technical debt in projects?
- How would prospect of future maintenance\support impact your decisions when you are working on a software change?
- Do you have any strategies for managing or reducing the impact of TD?
- Do you use any methods\techniques for technical debt management?
- What, if any, are the estimation techniques you\your team use for technical debt?
- Would the technical debt risk be assessed and normally be assessed up front? If so, how?
- Is the risk of the technical debt an important factor for you?
- How do you manage the gap between technical and non-technical stakeholders when communicating the impact of explicitly contracting technical debt?
- Who is responsible for TDM?
- How do you communicate the technical debt to a business decision maker?
    - How do you represent it?
- Does TDM present any challenges for you?
    - What are they?

Conclusion

- Do you have any ideas on how organisations should approach TDM?
- Do you have any other thoughts, comments, suggestions of what you have learned about technical debt / taking shortcuts in development what you would like to share?

Appendix H – Document Analysis Planning



1. Gather relevant documents

2. Organisation and management scheme

3. Copy for annotation

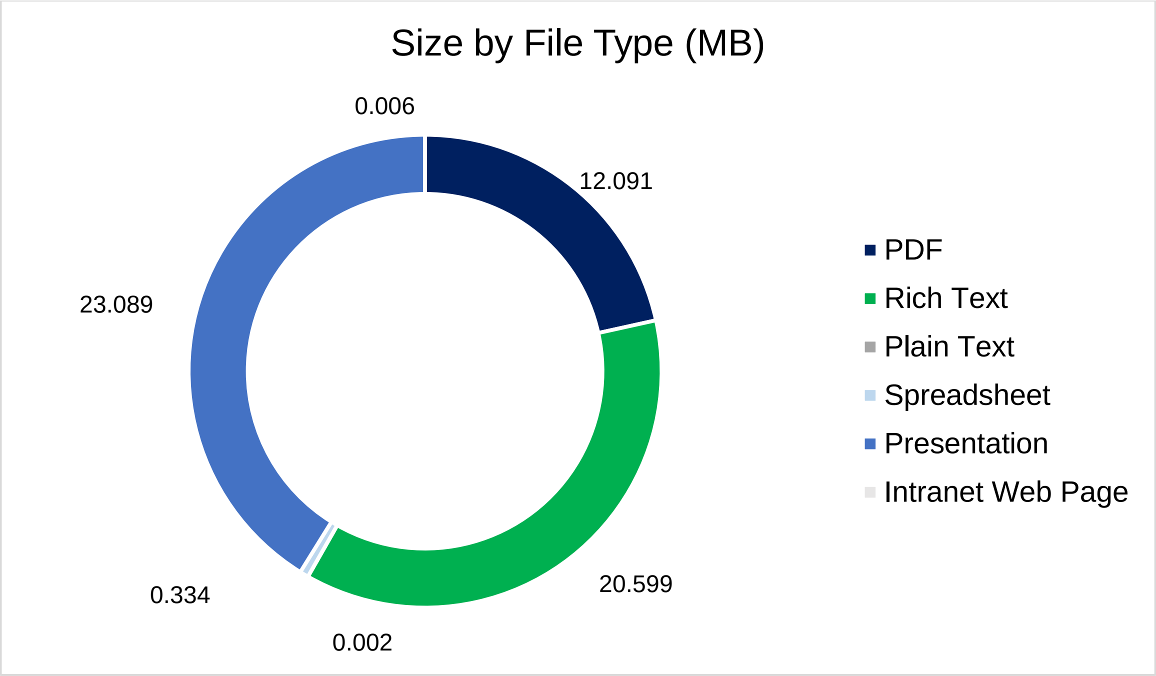
4. Establish authenticity

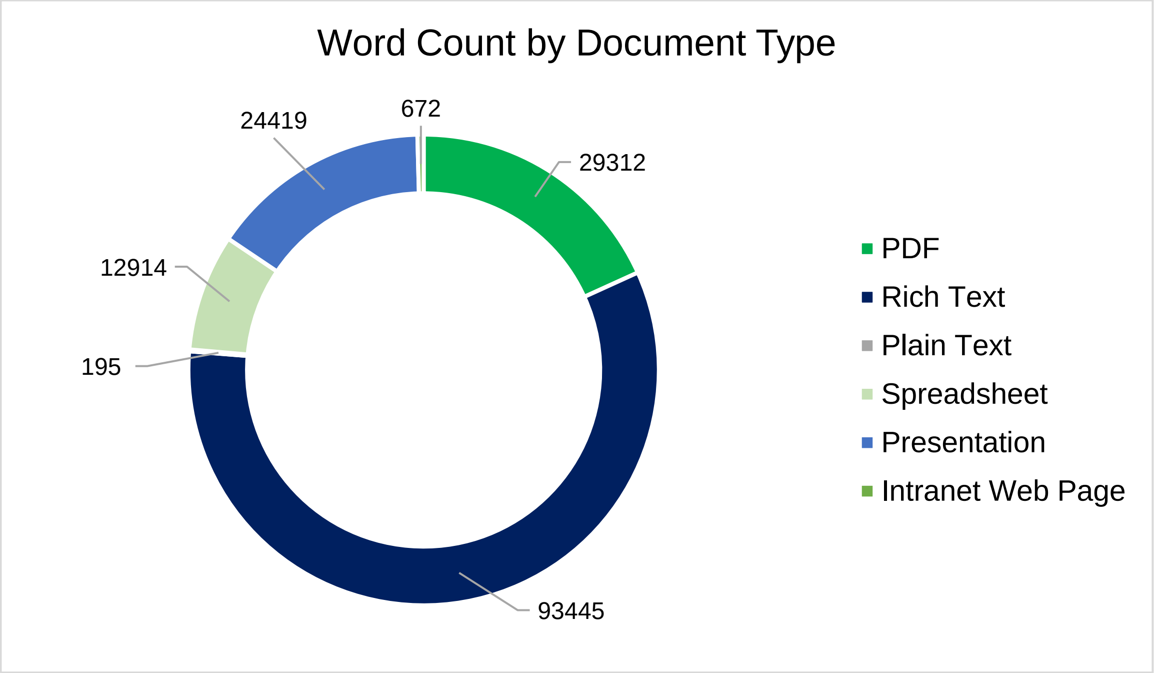
5. & 6. Explore docuement agenda, bias and background information

7. Ask questions about document

8. Explore content and extract and group data for research question

Appendix I – Document Analysis Statistics





File Type	Doc Count	Pages	Rows	Slides	Size	Words
PDF	6	128	-	-	12.091	29312
Rich Text	14	384	-	-	20.599	93445
Plain Text	1	1	-	-	0.002	195
Spreadsheet	3	-	314	-	0.334	12914
Presentation	7	-	-	211	23.089	24419
Intranet Web Page	3	3	-	-	0.006	672
Total	34	516	314	211	56.121	160957

Year	Coun
2017	6
2016	15
2015	7
2014	2
2013	3
2012	1
Total	34