BUSINESS INFORMATION SYSTEMS

University College Cork
Coláiste na hOllscoile Corcaigh

The Project Management Challenges, Benefits, Risks and Limitations of Adopting Agile Methodologies for a Multiphase ERP Program.

Eoghan Toomey B.B.S. (Accounting)

A Thesis Submitted for the Degree of Master of Commerce (MComm) of the National University of Ireland.

Research Supervisors: Prof. Philip O'Reilly & Prof. David Sammon

2021

CHAPTER 1 Introduction 3

1.1 Research Objectives 6

CHAPTER 2. Literature 10

2.1 ERP 10

2.2 Challenges of ERP implementations 12

2.3 ERP Project Management 13

2.4 ERP Implementation Failure 19

2.5 Software Development Methodologies 24

2.5.1 Plan-Driven Software Development Methodology.. 26

2.5.2 Agile Software Development Methodologies.. 30

2.5.3 Waterfall vs. Agile: Which is the Right Development Methodology for an ERP? 41

2.5.4 Scaled Agile Framework® (SAFe).. 46

2.5.5 Project Management Benefits, Risks, Challenges and Limitations of adopting
Agile-At-Scale Methodologies for an ERP Program.. 51

CHAPTER 3. Research Problem 58

3.1 Research Methodology 59

3.2 Participant Observer Research 67

CHAPTER 4. Findings 72

4.1 The Project Management challenges of adopting Agile for a multiphase ERP Program............ 74

4.2 The Project Management benefits and risks of adopting Agile for a multiphase ERP Program 80

4.3 The Project Management lessons learned of adopting Agile for a multiphase ERP Program... 86

4.4 Dell's Agile-At-Scale IT Delivery Framework 92

CHAPTER 5. Conclusions 117

References 122

Abstract

This research presents an insight into the project management challenges, benefits, risks and limitations of adopting an Agile-At-Scale methodology for a multiphase ERP (Enterprise Resource Planning) IT Programme.

As the researcher, my main motivation for undertaking this study was to gain a deeper understanding of how Agile could be leveraged in an ERP Environment. In my work for Dell Technologies, I have been at the forefront of Dell's Global IT Transformation initiatives and have been part of pilot programmes to adopt Agile and Agile-At-Scale across the IT function. While working closely with specialist external IT Consultancy firms, I identified a knowledge gap which raised research objectives relating to the project management challenges, benefits, risks and limitations of adopting an Agile-At-Scale methodology for a multiphase ERP programme. This study aims to address and further examine this knowledge gap.

This study is aimed primarily at both IT and Business Project leaders engaged in large scale, global, multiphase Transformational ERP Programmes.

The primary research method I adopted was that of participant observer. As a fully engaged participant in Dell's IT Transformation journey, my role provided me with the opportunity to attend professional Agile training, participate in the piloting of Agile Transformation Programs and be part of a Core Team whose role was to implement Agile. The Core Team's goal was to provide feedback to the Executive Leadership on the Agile adoption and contrast Agile with the more traditional Waterfall methodologies being used over my twenty years working on Global ERP Programs. To compliment the participant observer research method, I also used case studies, surveys and interviews.

This thesis uniquely illustrates, through the lens of the Agile ERP Quadrant view (Fig 2.18), a new understanding of the challenges, benefits, risks, limitations and lessons learned of Adopting Agile for an ERP Program. (Fig 4.1), builds on the Agile ERP Quadrant and denotes the positive, negative and neutral impacts by magnitude.

The study concludes that the adoption of Agile is not a one size fits all and highlights specific areas associated with ERP Programs that require a more hybrid approach.

This research will provide a detailed study of Dell Technologies Agile-At-Scale journey and will present an Agile Framework which can be adopted for a large-scale Global ERP Implementation.

Chapter One: Introduction

1.0 Introduction to the Study

The purpose of this chapter is to introduce the Waterfall and Agile concepts in an ERP environment. The research objectives are outlined along with the motivation for undertaking the study and the role of the researcher. Finally, an in-depth breakdown of the research plan by chapter is detailed.

When companies begin planning an ERP implementation, one of the first questions that needs to be answered after determining which application you will use, is what approach you will take to govern that ERP implementation. The two main approaches companies tend to use to govern an ERP implementation fall under one of two categories: Waterfall methodology and Agile development.

The Waterfall approach is so named because under this approach, each step is supposed to flow seamlessly to the next, like water cascading over a waterfall. In reality, an ERP implementation is a complex project that doesn't always follow a linear progression. The unexpected sometimes occurs, and requirements can change.

The Agile methodology has started to replace the Waterfall approach on many ERP implementation projects. Like Waterfall, Agile development requires a great deal of requirements gathering early in the project, and these requirements are used to guide the project plan. However, what Agile projects do with this information and how the project is managed through the development and deployment process is somewhat different. Rather than completing all the work in a linear progression prior to testing, Agile divides the project plan into short intervals called sprints. Testing occurs at the end of each sprint, and adjustments are made accordingly, rather than spending a great deal of time doing development for the entire project and only discovering and addressing issues late in the lifecycle.

Enterprise Resource Planning (ERP) systems are complex, expensive and powerful software systems that provide modules to support administrative areas in business management such as marketing, manufacturing, sales, finances, distribution, planning, human resource management, inventory management, project management and e-business. Because these systems are off the-shelf solutions, they require consultants to tailor and implement them based on the company's requirements and business needs [1].

Implementing this type of systems is a strategic and expensive decision to make for any company and despite that a lot of research have been conducted to fill the gap in the area of providing solutions for these challenges, the failure rate in ERP projects is still quite high as a recent report shows. During the last 4 years the average cost of ERP implementations has been roughly \$4.5 million with an average duration of 17.3 months. Of these projects, approximately 55% exceeded their planned budgets, and 66% percent experienced schedule overruns. Post implementation, 53% of organizations achieved less than 50% of the measurable benefits they anticipated from new ERP software [2]

The reason behind the high number of failed ERP implementation projects is that this type of projects has been facing many challenges. A recent case-study [3] showed that the issues and challenges can be categorized into six major themes reengineering (organization and infrastructures), top management commitment, funds, skilled manpower, implementation time and data fill-in [3]. Many of these issues could be a result of lack of agility in ERP implementation projects. An online survey conducted in 2010 on 45 firms that had experience in ERP implementation, showed that 38% of the implementation project used a pure big bang strategy [4].

The implementation process in big bang implementation strategy consists of separated phases which should be performed strictly in order. This approach is quite similar to the waterfall approach to systems development since it does not provide the option of going back to a previous phase or stage. In addition to that, it is quite expensive to perform changes in an ERP system. In contrast agile methodologies consist of iterative and incremental phases that deliver smaller parts of functionality from a series of comparatively short development cycles [5].

The lack of agility in ERP implementation projects might be one of the reasons behind the high failure rate in these projects. A study by West (2011) [4] showed that many IT projects are increasingly using Agile methodologies. Agile software development methods have shown themselves to address several of the challenges that traditional systems development projects have struggled with. About 39% of IT professionals said that they follow an Agile method. Because of the benefits Agile methods are bringing to IT projects, their popularity is growing among IT professionals. Thus, increasing the usage of Agile methodologies in ERP implementation projects has a high potential to smooth many of the current ERP challenges.

The ERP environment faces constant change and reassessment of organizational processes and technology. The Project Management method used with ERP deployments must provide adaptability and agility to support these evolutionary processes and technologies [6].

The use of Agile methods in the ERP domain provides:

• Increased participation by the stakeholders.
• Incremental and iterative delivery of business value.
• Maximum return on assets using a real options decision process.

1.1 Research Objective and Questions

This Thesis aims to investigate three areas.

- The Project Management challenges of adopting Agile Methodologies for a multiphase ERP Program.
- The Project Management benefits and risks of adopting Agile Methodologies for a multiphase ERP Program.
- The Project Management lessons learned of adopting Agile Methodologies for a multiphase ERP Program.

1.1.1 Motivation to undertake this Study

As an IT Program Manager with 20 Years' experience working on Transformational ERP Programs across a number of sectors ranging from Telecoms to Software to High Tech Manufacturing and Services, I have seen first-hand how the traditional Waterfall Methodology of Project Management has evolved into a more Agile approach. Working for Dell Technologies has afforded me the ability to be at the cutting edge of adopting and piloting these new methodologies to an ERP environment**.**

1.1.2 Role of the Researcher

My role at Dell Technologies as a Senior Global Program Management Consultant (PMP/CSM) exposes me to an ever changing and extremely fast paced transformational environment. The recent merger with EMC/VMware, the largest technology merger in history, has presented major opportunities and challenges from an IT Program Management perspective as we work to bring the IT Systems of these major corporations together. My role is to manage these large-scale Global Transformation ERP Programs with annual budgets of $10m+ and Project teams of 300-500 people. In recent years, Dell Technologies is transitioning to a more Agile approach to Project Management, I was part of the Pilot Team for the adoption of Agile Methodologies, so I have an in-depth knowledge of the lessons we have learned.

1.2 Outline of the Research Plan

The plan of research is as follows.

The purpose of **Chapter Two** is to provide a definition and investigation of ERP Programs. It illustrates the challenges of implementing an ERP Program and the reasons for ERP failures as well as the best practices to avoid such failures. Chapter Two will look specifically at the Project Management of ERP Programs and looks at the associated Project Management knowledge areas. This Chapter will reveal a detailed analysis of the Software Development Methodologies and provides a comparison across Waterfall and the many evolving Agile Methodologies, including the Scaled Agile Framework (SAFe). Chapter Two concludes with the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program.

2.1 provides an introduction to the concept of ERP systems, the distinction between ERP systems and other IT systems is very relevant to this thesis as the interdependencies across ERP modules and interlocking systems plays a major part in the understanding of the benefits, risks, challenges and limitations of adopting Agile-At-Scale Methodologies for an ERP Program.

2.2 illustrates the challenges and risks associated with ERP Implementations and reveals that the high degree of complexity and change in ERP Programs requires an effective Project Management Methodology.

2.3 describes the role of Project Management in ERP implementations and emphasizes its impact across the Project Management knowledge areas. It concludes by revealing how it greatly improves the odds of an ERP implementations success.

2.4 describes the most common reasons for ERP implementation failures and illustrates the best practices for avoiding such failures.

2.5 goes into greater depth on the Software Development Methodologies, it compares and contrasts Waterfall and the various Agile Methodologies and asks which is best suited to an ERP Program. This Chapter concludes with a detailed breakdown of the benefits, risks, challenges and limitations or adopting Agile-At-Scale for an ERP Program.

Chapter Three presents the Participant Observer Research in the form of a comprehensive survey across Dell's IT function. The survey reveals Dell's Agile experiences, the areas where Agile is working well and the areas that need improvement.

3.2 presents the detailed findings of an extensive Dell IT survey, specifically on Dell's Agile adoption experiences. It describes the key highlights, opportunities and takeaways.

Chapter 3 concludes that while adopting Agile allows for better traceability and the ability to deliver key requirement to the business partners, it requires significant investment in training and business alignment and introduces challenges across Release Management, Environment Management and Roadmap planning.

Chapter Four describes and interprets the findings of the Dell Survey and looks specifically at Dell's experience in a large multiphase ERP Program. It focuses on the research objectives of the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program. Chapter four will illustrate the research objectives by providing evidence based on Dell's direct experiences and will set the foundation for Chapter Five which will conclude on how best Agile-At-Scale can be adopted in an ERP Program.

Chapter 4 breaks down the challenges, benefits, risks, limitations and lessons learned of Adopting Agile for an ERP Program. It provides concrete examples of Dell's experience across these categories and ties back the findings to the company wide findings outlined in Chapter 3. Chapter 4 concludes that the adoption of Agile is not a one size fits all and highlights specific areas associated with ERP Programs that require a more hybrid approach.

Chapter 4 presents an in-depth analysis of Dell Technologies Agile Adoption and Dell's Agile-At-Scale IT Delivery Framework. The research illustrates the details behind the Scrum Workflow and describes the Project Teams associated roles and responsibilities, the artefacts and the Scrum/Sprint Planning procedures. It reveals the concept of the MBI (Minimum Business Incremental Business Value) and how this is used to form and sequence the Product backlog.

4.4 presents Dell's Scrum Workflow and illustrates the flow from Program Vision, through refining the business objectives, creating the Product backlog, release coordination, the coordination of dependencies and the release plan. It explains how to establish the Sprint backlog, define and accept Sprint goals and complete Sprint planning.

4.4 provides an example and framework for how MBI's can be sequenced and the benefits of such an approach.

4.4 demonstrates how Dell's Agile-At-Scale IT Delivery Framework fits into the wider overall IT delivery mechanism, it also illustrates all of the business and IT artifacts and process flows.

Chapter 2. Literature Review

2.0 Introduction

The purpose of Chapter Two is to provide a definition and investigation of ERP Programs. It illustrates the challenges of implementing an ERP Program and the reasons for ERP failures as well as the best practices to avoid such failures. Chapter Two will look specifically at the Project Management of ERP Programs and looks at the associated Project Management knowledge areas. This Chapter will reveal a detailed analysis of the Software Development Methodologies and provides a comparison across Waterfall and the many evolving Agile Methodologies, including the Scaled Agile Framework (SAFe). Finally, Chapter Two concludes with the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program.

2.1 ERP

Enterprise Resource Planning (ERP) is a software solution that integrates business functions and data into a single system to be shared within a company. While ERP originated from manufacturing and production planning systems used in the manufacturing industry, ERP expanded its scope in the 1990's to other "back-office" functions such as human resources, finance and production planning (Swartz & Orgill, 2001; Nieuwenhuyse, Boeck, Lambrecht, & Vandaele, 2011) [1]. Moreover, in recent years ERP has incorporated other business extensions such as supply chain management and customer relationship management to become more competitive (See Figure 2-1 below).

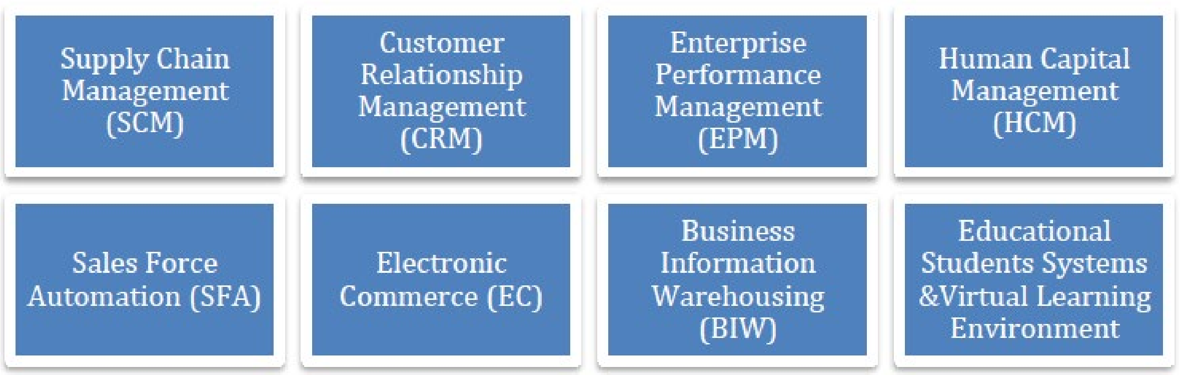

Supply Chain Management (SCM)
Customer Relationship Management (CRM)
Enterprise Performance Management (EPM)
Human Capital Management (HCM)
Sales Force Automation (SFA)
Electronic Commerce (EC)
Business Information Warehousing (BIW)
Educational Students Systems &Virtual Learning Environment

Fig 2-1. ERP Extension (Abbas, 2011)

The major goal of ERP is to increase operating efficiency by improving business processes and decreasing costs (Nah, Lau, & Kuang 2001; Beheshti 2006) [2]. ERP allows different departments with diverse needs to communicate with each other by sharing the same information in a single system. ERP thus increases cooperation and interaction between all business units in an organization on this basis (Harrison, 2004) [3].
Furthermore, ERP standardizes processes and data within an organization with best practices. The company also streamlines data flow between different parts of a business by creating a one-transaction system (Lieber, 1995). As Hitt, Wu, and Zhou (2002) [4] stated, "The standardized and integrated ERP software environment provides a degree of interoperability that was difficult and expensive to achieve with stand-alone, custom-built systems." Standardization and integration of processes and data allows a company to centralize administrative activities, improves ability to deploy new information system functionality, and reduces information system maintenance costs (Siau, 2004) [5].
As a result of its benefits, ERP has become the backbone of business intelligence for organizations by giving managers an integrated view of business processes (Parr & Shanks, 2000; Nash, 2000) [6]. ERP is designed to adapt to new business demands easily. The continuous technological advancement and the increasing complexity of ERP require companies to regularly upgrade their systems. Most ERP vendors provide an opportunity to update procedures and align with perceived best practices to meet changing business needs more quickly (Harrison, 2004) [3].
A significant number of organizations have adopted ERP over the last two decades, and the revenue of the ERP market has grown from \$17.2 billion in 1998 (O'Leary, 2000) to \$39.7 billion in 2011 (Dover, 2012) [7]. Overall, the ERP market size is predicted to reach \$41.69B by 2020 according to Allied Market Research. [15]

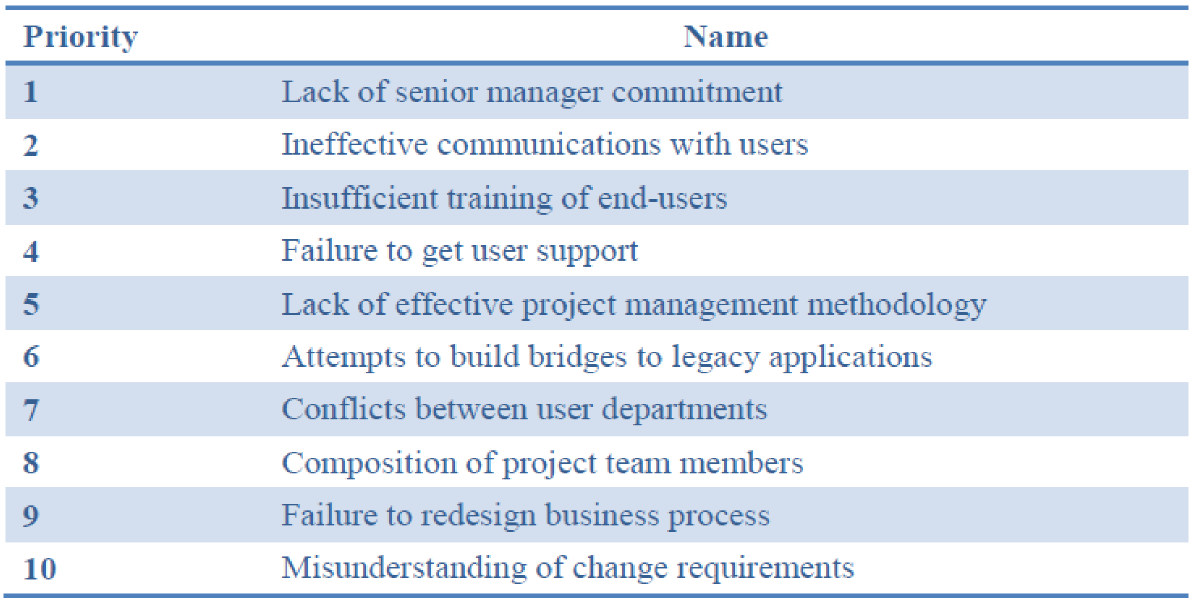
2.2 Challenges of ERP implementations

In spite of ERP's significant growth from the late 1990s to the present day, there are a number of challenges that companies may encounter when implementing ERP.

Dillard and Yuthas (2006) [8] stated that most multinational firms are using ERP and that more small and midsize companies have begun to adopt ERP. Despite ERP's promises to benefit companies and a substantial capital investment, not all ERP implementations have successful outcomes. ERP implementations commonly have delayed an estimated schedule and overrun an initial budget (Ehie & Madsen, 2005; Helo, Anussornnitisarn & Phusavat, 2008) [9].

Furthermore, the literature indicates that ERP implementations have sometimes failed to achieve the organization's targets and desired outcomes. Much of the research reported that the failure of ERP implementations was not caused by the ERP software itself, but rather by a high degree of complexity from the massive changes ERP causes in organizations (Scott & Vessey, 2000; Helo *et al.*, 2008; Maditinos, Chatzoudes & Tsairidis, 2012). [10]

These failures can be explained by the fact that ERP implementation forced companies to follow the principle of 'best practices' in most successful organizations and form appropriate reference models. (Zornada & Velkavrh, 2005) According to Helo *et al.*, (2008) [11], "the major problems of ERP implementation are not technologically related issues such as technological complexity, compatibility, standardization, etc. but mostly [about] organization and human related issues like resistance to change, organizational culture, incompatible business processes, project mismanagement, top management commitment, etc.". Huang et al, Chang, Li and Lin (2004) presented the top ten risk factors causing ERP implementation failure.



Priority	Name
1	Lack of senior manager commitment
2	Ineffective communications with users
3	Insufficient training of end-users
4	Failure to get user support
5	Lack of effective project management methodology
6	Attempts to build bridges to legacy applications
7	Conflicts between user departments
8	Composition of project team members
9	Failure to redesign business process
10	Misunderstanding of change requirements

Table 2.2. Top ten risk factors of ERP (Huang *et al.*, 2004)

These risk factors illustrate various organizational considerations: organization fit, skill mix, project management and control, software system design, user involvement and training, and technology planning.

Since ERP implementation inevitably causes organizational changes, it requires the engagement of senior management from across the organization who is able to resolve conflicts. Without the commitment of senior management, ERP implementation has a high risk of failure.
In other words, due to changes in business processes across an organization, there can be resistance to adopting the ERP system. ERP connects and integrates all business functions within the organization. Therefore, it is critical that management staff be committed, and particularly that they equip employees who are using business functions influenced by ERP with clear channels of communication. Lack of end-user training increases risks by creating confusion and inaccuracy, thereby decreasing user satisfaction and the credibility of the system.

2.3 ERP Project Management

Excellent project management is also needed for successful ERP implementation. (Sammon 2007) [16] Project teams should have clear guidelines to execute ERP implementation from their project objectives and work plan to their resource allocation plan. Without good project management, ERP implementation projects that are large in scale and must take place over longer time periods may end in failure.
Furthermore, the composition of team members plays a crucial role in ERP implementation. ERP integrates diverse business functions across an organization into one single system, necessitating a complex and integrated software package. If a project team does not clearly understand the changes in its organizational structure, strategies, and processes from ERP implementation, it will not be in a position to benefit from ERP's competitive advantage. In order to best implement ERP, project team members should be selected with a balance between members with business experience within the organization and external experts with specialties in ERP.
From the perspective of project management, the iron triangle Fig. 2.3 can illustrate how important it is to balance the three corners of the triangle – scope, schedule and cost. (Lamers, 2002) [13]

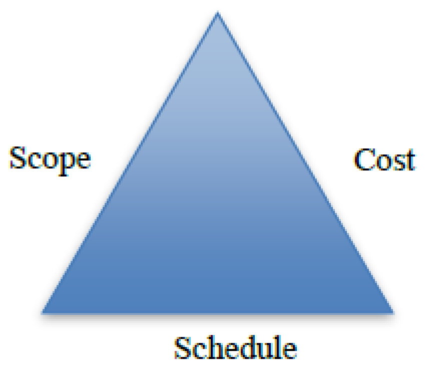


Fig 2.3. The iron triangle of project management (Lamers, 2002)

However, in ERP implementations, both schedule and cost tend to be underestimated, while scope is overestimated (Aiken, 2002). ERP changes the entire organizational environment by reengineering the entire business process; thus, after implementation, it is not easy to revise previous processes. Therefore, ERP implementations need accurate estimation, preparation with a holistic view, and systematic management of the entire implementation process.

2.3.1 Project Management Knowledge Areas for ERP Implementations

The Project Management Institute systematizes the body of knowledge of project management into nine areas: scope management, human resource (HR) management, risk management, communications management, procurement management and integration management in addition to the three constituents mentioned in section 2.3 (Scope, Cost, Schedule). As a project manager becomes more sophisticated in managing these areas, the processes used to manage a project become more consistent and systematic that can contribute to a higher rate of project success.

The different project management areas have presented difficulties to contemporary management of IT projects. A large-scale project managed at different locations, in different time zones, and by different users can create many difficulties. These are also applicable to ERP implementations because an ERP system is typically large-scale, cuts across functional boundaries, and often has heterogeneous stakeholders. This is especially so in a multinational company where business units are on different continents. In these situations, decoupling the large-scale software project into flexible and manageable modules can be a challenge, and cross-functional coordination is one of the most important issues in ERP implementations. Excellence in scope, time, cost, risk, and communication management is essential in meeting this challenge.

Agile development techniques such as rapid application development can induce higher risks and poorer quality than the traditional development method. Quality and risk management of products and processes are crucial to the success of Agile development methods. For ERP implementations in particular, in-house expertise is often lacking, and companies often turn to external consultants in implementing the system, but the outsourcing of jobs does not transfer the ultimate management responsibility for their successful completion. Poor management of outsourcing responsibilities can increase risks and create integration problems across products and processes.

The techniques of procurement and integration management can help IT managers succeed in the outsourcing activity. In addition, an organization needs to avoid project management problems such as "estimate to please" and establishment of subjective and immeasurable objectives. Unrealistic cost estimates and lack of objective benchmarks can contribute to escalating costs, and cost management is an important skill in the face of this challenge.

The importance of project management cannot be emphasized enough, particularly in the development of large-scale software projects. Chen's study [14] adopts the project management areas of expertise to assess the project management practices of the ERP implementation because these areas and practices are widely accepted throughout the project management profession. In fact, these same areas have been codified in the

IEEE Standard 1490-(2003) [ ], which states that the areas and practices are generally accepted, and "generally accepted means that the knowledge and practices described are applicable to most projects most of the time, and that there is widespread consensus about their value and usefulness" . Given that the first three project management areas (i.e., budget, schedule, and quality) already have obvious implications for project success, Chen's study focuses on six other process-oriented project management knowledge areas: *scope*, *HR*, *risk*, *communications*, *procurement*, and *integration*.

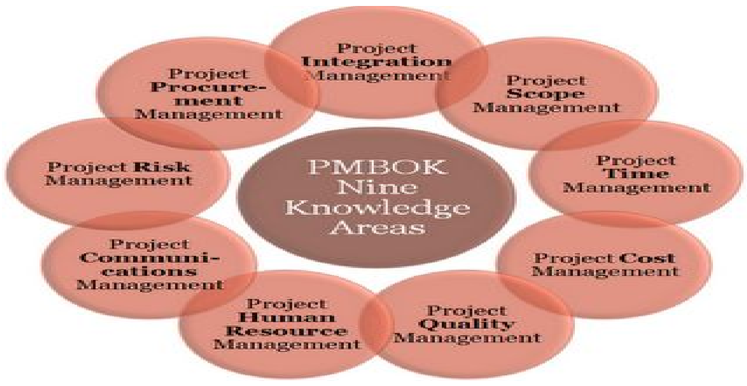

Project Integration Management
Project Scope Management
Project Time Management
Project Cost Management
Project Quality Management
Project Human Resource Management
Project Communi-cations Management
Project Risk Management
Project Procure-ment Management
PMBOK Nine Knowledge Areas

Fig 2.4 PMBOK nine knowledge areas. Chew C et al (2009).

For practitioners, it is important to recognize that stakeholders at the project, business unit, and corporate levels often have divergent interests. IIED [17] An enterprise system can impact these users in different ways and create conflicts among these stakeholders.
What Chen's Study demonstrated is that it is critical to manage these impacts and conflicts by incorporating project management practices in the implementation process (i.e., communications, scope, risk, HRs, procurement, and integration management).

In terms of ***communications***, the first phase of Chen's case study showed that the presence of conflict and resentment created symptoms such as hostility, jealousy, poor communication, frustration, and low morale. The lack of an open forum to involve users in the system implementation process can create paralyses in effective communication, goals alignment, trust, and poor system design between management and IS.
Thus, it is important for project managers to manage the communication process and create a forum in which stakeholders can order priorities and discuss issues. Managing the conflict between business and IS throughout a system development cycle is imperative to the successful delivery of an IS project. User participation has been an effective mechanism to lessen conflict, thereby improving system development outcomes.

In terms of ***scope*** management, many authors have cautioned that customization would likely increase the cost and risks of ERP implementation and the difficulty for upgrades and migration to future releases. However, some amount of customization will always be necessary to meet specific business requirements, especially, in a multinational company with different regional requirements. To capitalize on business opportunities, changing system requirements is a viable option from a managerial perspective, but this represents a great economic cost to any company that trades system functionalities for business agility.

The conflict between the need to meet business needs and the need to control system complexity causes tension between management and IS professionals, and the pressure to resolve the conflict creates a sense of obligation in the system implementer to change system requirements to meet business needs. This, in turn, reinforces an unspoken commitment to adopt the "change" option, even though there are viable alternatives
(e.g., maintenance, off-the-shelf package, or no change).

Creeping requirements can be especially destructive because of their implicit nature, which can mean that their negative impacts are never fully and explicitly recognized, acknowledged, or addressed. Any changes made to honour creeping requirements will be interpreted as a reinforcement of an earlier promise or commitment—whether or not that is the intent of the MIS department.

As a result, MIS can be kept from committing their limited resources to what matters most to enterprise projects, such as reliability, functionality, and training. The chain effect of disagreement and interference during the system requirements acquisition can affect project outcomes.

Project managers can consider a two-pronged approach to manage scope. First, to avoid entering a competing mode with management, a top-down policy on scope can be put into practice (e.g., keeping 85% of business processes common). Second, to facilitate the implementation of such a policy, a bottom-up process involving Super Users and functional areas can be adopted (e.g., forming a prioritization committee). A prioritization committee can serve as a successful scope management vehicle because it can lower the extent of user resistance by involving users across different areas. Conflicts of interest are avoided by improving the degree of transparency in the decision-making process. This case affirms the importance the of scope management vehicle in the development of an enterprise system, and scope planning and definition skills can minimize scope creep problems and channel-limited resource to key issues.

Risk management is important to an IT project, especially one that spans the enterprise. External (e.g., new business models and entrants) and internal (e.g., project size, duration, structure, complexity, and outsourcing) aspects of task, process, or environment can increase the likelihood of unfavourable project outcome, and these aspects represent risks to the project.

Thus, project managers can consider measuring the risk of an ERP project as an important part of risk management, and to the extent possible, a firm should adopt a formal method of assessing risks. Once identified, different categories of risks can be managed with specific action strategies, and different forms of risk control process can be adopted to tailor risk management to specific contexts.

In managing *HRs*, it should be recognized that in-house employees tend to have a lower level of readiness than vendors in implementing an enterprise system. The shortage of critical skills and knowledge in most companies and high turnover rate of IT professionals pose additional challenges. However, these HR issues do not have to be an inhibitor of a successful implementation of an enterprise system. To facilitate knowledge transfer, a company can pair in-house employees with vendors based on similarity in work values, norms, and problem-solving approaches. The idea is to support ERP implementation with a knowledge management mind-set that can facilitate the knowledge generation, transfer, and absorption process between internal and external stakeholders.
In-house employees can solve problems more efficiently and effectively after acquiring system-related skills and knowledge.
The complementary support of a knowledge management system can further the success rate of ERP implementation.

In ***procurement*** management, managing partners should be the responsibility of the adopting company, instead of that of the vendor. What this means is that project managers should develop a list of performance metrics for vendors, work out how to measure them, and obtain regular performance measurements. If there is a deviation from benchmark, project managers should assume a hands-on role to track the issue and bring it to closure, instead of relying on vendors themselves to address the issue. Overall, the adopting company needs to keep track of the progress of the vendor–client relationship and take corrective actions if necessary, and a well-managed partnership can incrementally transfer vendor’s knowledge and skills to in-house employees. In addition, the cultural fit between clients and vendors is indispensable for the long-term success of ERP project.

Finally, ***integration*** management is the mechanism that directs all stakeholders at the project, business unit, and corporate levels toward the same direction. Firms contemplating ERP deployments are recommended to have not only a prioritization committee, but also an *empowered* prioritization committee that is authorized to make binding decisions and creates concerted efforts in accomplishing business goals. Setting expectations at the onset of the project would also be useful, i.e., SME's would be expected to give up some of their local processes in order to conform to the 85% policy. In addition, at the project level, it is suggested that for some time after system deployment, those in-house employees who have worked on development also work side by side with the helpdesk support staff. This way, system knowledge can be transferred to the helpdesk and the eventual integration of the ERP system into the organization can be facilitated.

The inclusion of project management skills can greatly improve the odds of ERP implementation success. Both Chen's [14] and Sammon's (Sammon 2007) [16] studies affirm this proposition by presenting evidence for progress from phase 1 to phase 2 where the company leveraged six important project management areas—scope, HR, risk, communications, procurement, and integration management. The importance of these project management skills is often underestimated.

2.4 ERP Implementation Failures

ERP implementation failure occurs when the project does not meet one or more of its key goals, which can include: Goeun Seo (2013). [5]

- On-time implementation
- On- or under-budget implementation costs
- Minimal disruption to business operations
- Improved organizational efficiency
- Reduced operating costs
- Increased sales or revenue

Software Advice [1] examined 22 high-profile ERP implementation projects from the past decade that were dubbed "failures" by software consultants and industry publications. They looked at the common factors behind the implementation failures in their sample, culled from news reports, blog posts, Securities and Exchange Commission (SEC) filings and court dockets.

One critical thing to understand is that **ERP implementation rarely fails because of the software itself**. Indeed, only 18 percent of the failures we examined were due to buggy software (and of those that were, many were a result of the organization's own hefty customizations).

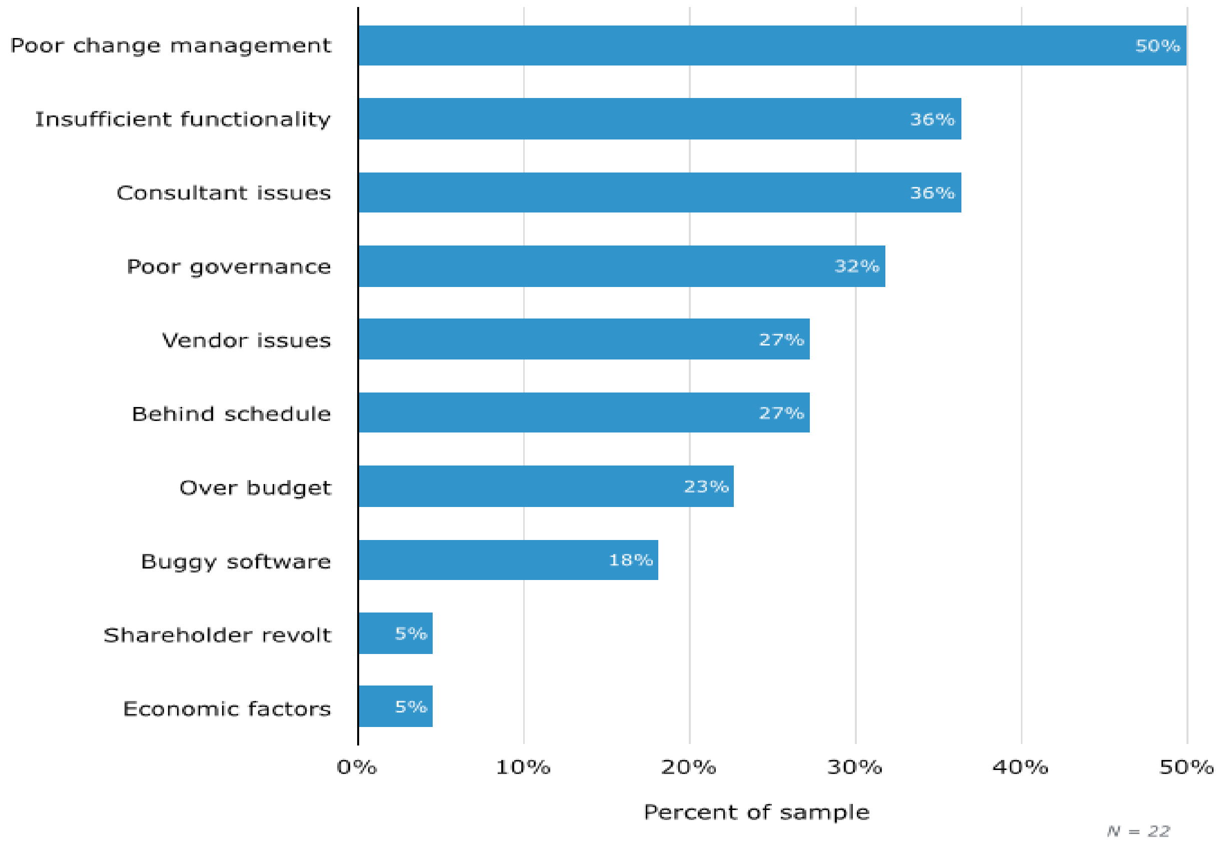


Fig. 2.5 Most Common Reasons for ERP Implementation Failure

Poor change management—which includes inadequate training and executive planning—was a driving factor in half of the implementation failures in our sample. Indeed, it's not enough to merely train employees to use the new system. It is also critical for managers to understand and explain how adopting the system will impact staff's core responsibilities—while communicating the benefits the new software will bring.

In 36 percent of the failures examined by *Software Advice*,[1] the problem stemmed from the organization's **functional requirements for the software not being met**.

In such situations, the organization, its implementation consultants and the ERP vendor all share the blame: Typically, it's a combination of:

- The organization not doing its due diligence in researching the system
- The consultants not fully understanding their client's needs, and
- The vendor over-hyping the system's capabilities

Consider what happened when one manufacturing firm didn't do its homework. Back in 2011, Group Manufacturing Services, a plastics and metal manufacturer, began implementing a new ERP system.3 From the get-go, the implementation was a mess—largely because the company took it on without the assistance of the vendor's implementation services.

During the implementation, the company realized that the system would not support a critical function out of the box: According to PC World, the platform could not support the manufacturing firm's quoting system without a customization that could cost as much as $24,000. This prompted the company to renege on its contract and file a lawsuit against the vendor.
Ultimately, the court dismissed the case, and the company was back at square one—presumably with lighter pockets. Had the company researched the system's capabilities in depth ahead of time, the whole situation might have been avoided.

Too often, an organization's in-house IT team is not given a voice during the initial selection and early planning phases of ERP software selection, despite having the most in-depth knowledge in the company about the technical requirements for its IT infrastructure.

Implementing ERP must be viewed and undertaken as a new business endeavour and a team mission, not just a software installation. Companies must involve all employees, and unconditionally and completely sell them on the concept of ERP for it to be a success. A successful implementation means involving, supervising, recognizing, and retaining those who have worked or will work closely with the system. Without a team attitude and total backing by everyone involved, an ERP implementation will end in less than an ideal situation. Barker 2003 [6]

2.4.1Best Practices for Avoiding ERP Implementation Failure

To ensure a smooth implementation, here are some best practices to keep in mind: Somers and Nelson (2001) [7]

Practice proper change management. It's frustrating for employees to come into work one day only to find that their workflows have been dramatically changed. Begin training employees on the new system early and consider how the new system will often force them to learn new processes or tools.

Further, make sure to always address any problems or concerns workers might have. In some cases, it's a good idea for upper management to "shadow" employees when they are completing tasks that will be directly impacted by implementation, in order to better understand how their workflows will change.

Don't rush it. Implementations take time, and unrealistic time frames only create more problems. Many factors can determine an implementation time frame—but in general, a small to midsize company can expect the process to last anywhere from six months to two years.

"Trust your implementation teams when they give you a time frame," Lincoln says. "You are paying for their expertise, so believe them. Trying to force a project to deliver within an unrealistic time frame is a recipe for disaster."

Engage and involve your IT team. There's a reason why you're in the executive suite while your IT manager is in the server room: You're best at running a business, and he's best at dealing with your company's technology needs.

Unless you're particularly knowledgeable about the technical intricacies involved with implementing a new ERP system, respect the fact that your IT team are the experts, and defer to their judgment when possible.

Take the time to listen to their concerns, and ensure they have the support and resources they need from other departments to execute on the project. During the selection and implementation process, solicit their feedback and make sure that there are no lingering issues concerning data migration or training that could boil over halfway through the implementation.

Know what you need and know what you're buying. It's critical to ensure that the vendor you're buying from can provide the functionality you need. While vendors generally have the common sense to not deliberately mislead clients, critical technical requirements can fall to the wayside during negotiations—especially if a firm's decision-makers are not independently confirming that the vendor will be able to provide all features and functions necessary.

In addition to demonstrating potential new systems, decision-makers should seek outside, third-party advice from current users and/or from consultants.

In section 2.4, we have looked at the most common reasons for ERP implementation failure and the best practices for avoiding such failures. Section 2.5 will look at Software Development Methodologies and how from a Project Management perspective, Agile can mitigate or accentuate such challenges for an ERP Program.

2.5 Software Development Methodologies

Selecting an appropriate software development approach is as crucial to a project success as applying project management best practices. In order to deliver better, faster and cheaper software products, software practitioners and academics have suggested many approaches such as Waterfall, Rational Unified Process and Agile. These three approaches are the most frequently used approaches for software development [1] [2]. In particular, the Agile approaches have become popular in the IT industry due to its ability to handle a high degree of uncertainty in software development project. Rather than relying on the process, Agile puts the emphasis on people and communication by following the light-but-sufficient rules of project behaviour [3]. It has also been proven to provide high productivity and speed up the development process beyond conventional development approach i.e. Waterfall [4].

The idea of Agile emerged from experienced practitioners who proposed the lightweight approaches for software development. They published their common idea in 2001 in the form of the Agile manifesto. The manifesto asserts that Agile software development should focus on four core values [5]):

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The Agile manifesto was introduced after various versions of Agile software development were defined. Scrum and Dynamic System Development Method (DSDM) were among the oldest Agile approaches created in the early-1990s. Extreme Programming (XP) was introduced in the late 1990s and become the most popular approach of Agile software development at that time [6] [7]. The Agile approach constitutes a set of practices for software development that have been proved effective by experienced practitioners. For example, XP is comprised of major practices such as planning game, pair programming, refactoring, 40-hour week, and more [7]. Scrum requires certain project management practices in various phases such as sprint planning meeting, sprint review meeting, and creating/controlling the product backlog [8].

There is no single Agile development approach that ideally fits all project contexts without adaptation due to the uniqueness of software project. Every project is different whether in its subject area, development team or project size. Because of this, adapting Agile practices according to project circumstances is a must. For example, a large development team cannot be as agile as a small one. However, the team can exploit the use of the Agile approaches to be light and effective, as long as they can creatively adjust the practices to the situation [3].

The Chaos reports published by The Standish Group from 1994-2009 [14] indicated that many software development projects failed. In 2009, 24 % of projects were categorized as "failed" which mean that the projects were cancelled prior to completion or delivery and never used. 44 % of the projects were defined as" challenged" which indicated that the projects were late, over budget and/or produced fewer functions and features than required. Only 32 % of projects were reported as "successful projects", which mean they were delivered on time, on budget, and with required functions and features (see Figure 2.6) [14]. Although the actual numbers and the validity of the data are still debated among experts, this figure is typically accepted as a description of the actual situation of software development nowadays [15].

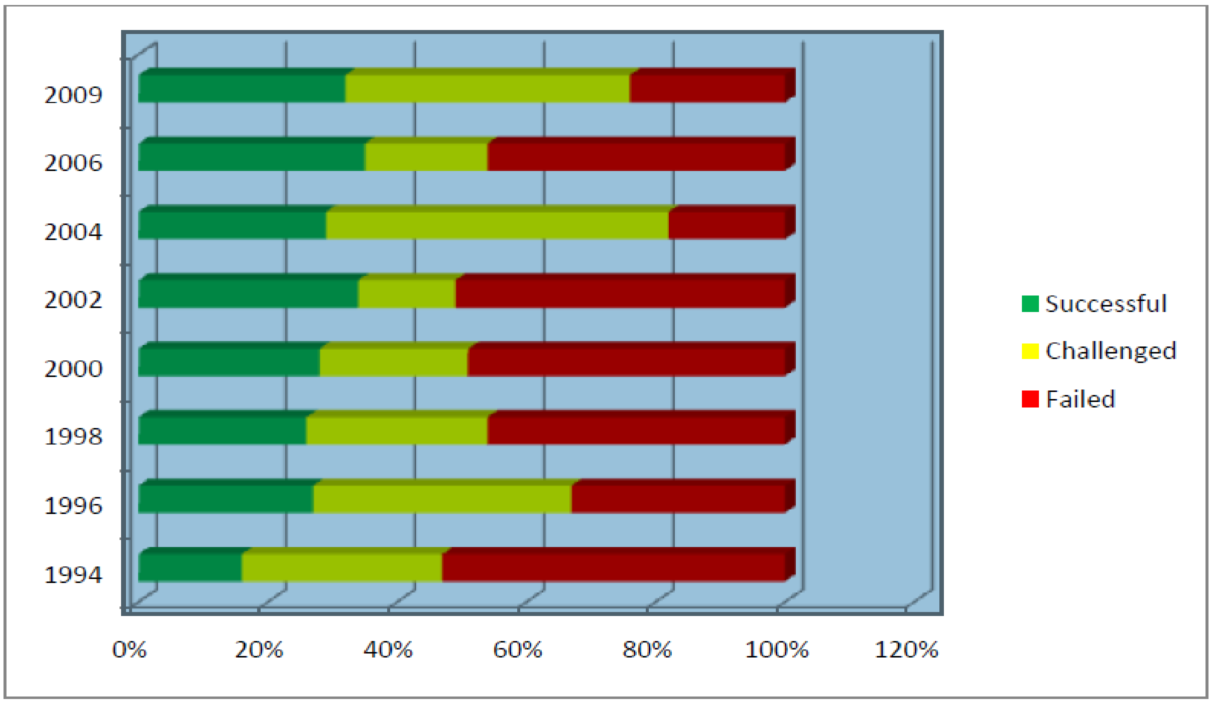


Fig 2.6 CHAOS report chart. Adapted from [14]

The cause of late, over budget and fewer functions/features delivered projects were not explained explicitly. However, by a survey of 365 IT executive managers, several main project success factors were discovered in this Chaos report, namely: user involvement, executive management support, clear statement of requirements, proper planning, realistic expectations, smaller project milestones and competent staff [16]. In the present, software practitioners are still looking for better ways to minimize project failure. One of the most important ways is to find better software development methodology which can lead to more successful software development projects. There are two main categories of software development methodologies, namely Plan-Driven Software Development Methodology and Agile methodology. Both methodologies will be briefly described in the following section.

2.5.1 Plan-Driven Software Development Methodology

There had been many plan-driven software methodologies developed since the invention of computer programming language. As indicated from their category name, these software methodologies put emphasis on planning everything from the beginning to the end of a project. These methodologies are well suited for well understood problems with well-defined output from the beginning [11]. Several well-known plan-driven methodologies such as Waterfall, Rational Unified Process (RUP), V-Model, and Spiral Model will be presented in the following section, before going on to examine Agile methodologies in 2.5.2.

• Waterfall

The Waterfall methodology generally used to refer to common use of software development practice, which has a figure like a waterfall (see Figure 2.7).The “waterfall” terminology was never proposed by Royce although people often refer the model to his paper [17]. In this consecutive model, each phase will be activated when its preceding phase has been completed.

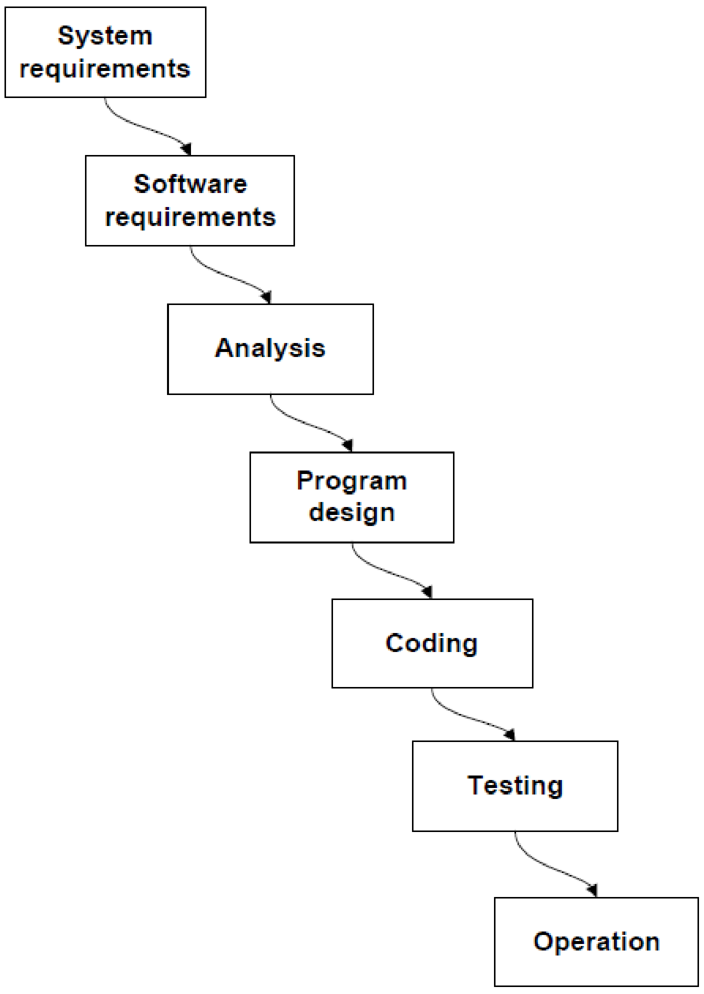


Fig 2.7 Waterfall methodology (adapted from [17]

● Rational Unified Process (RUP)

The Rational Unified Process (RUP) was developed by Rational Software Corporation in 2003. This methodology is categorized as an iterative enhancement model that is adaptable to the organization and project specific context. RUP lifecycle consists of four phases, which are [18]:

● Inception: determine the business case for the system and define the project scope

● Elaboration: analyse the problem domain, set up the architectural foundation, develop the project plan and mitigate the key risk items.

● Construction: build the software components and features. Integrate them into the product and test it thoroughly.

● Transition: deploy the system to the end user. Handle all issues by correcting problems, developing new releases, and finishing postponed features.

Each phase can be broken down into several iterations which resulted in a release of an executable product. The produced system will grow incrementally from iteration to iteration and finally will become the final system.

There are six core process workflows, namely: business modelling, requirements, analysis and design, implementation, test, and deployment. In addition, there are three core supporting workflows, namely: configuration and change management, project management, and environment [18].

● V-Model

The V-Model is used as the standard methodology in German defence and federal administration software development projects. The initial version of V-Model was developed in 1986 and has been evolved into V-Model 92, V-Model 97 and V-Model XT [19]. The V-Model 97 methodology can be seen in Figure 2.8. Verification steps are performed to ensure that in each phase the developed deliverables satisfy the requirements. In validation steps, this model relates each phase of development stage with its corresponding testing phase to ensure the system correctness [20].

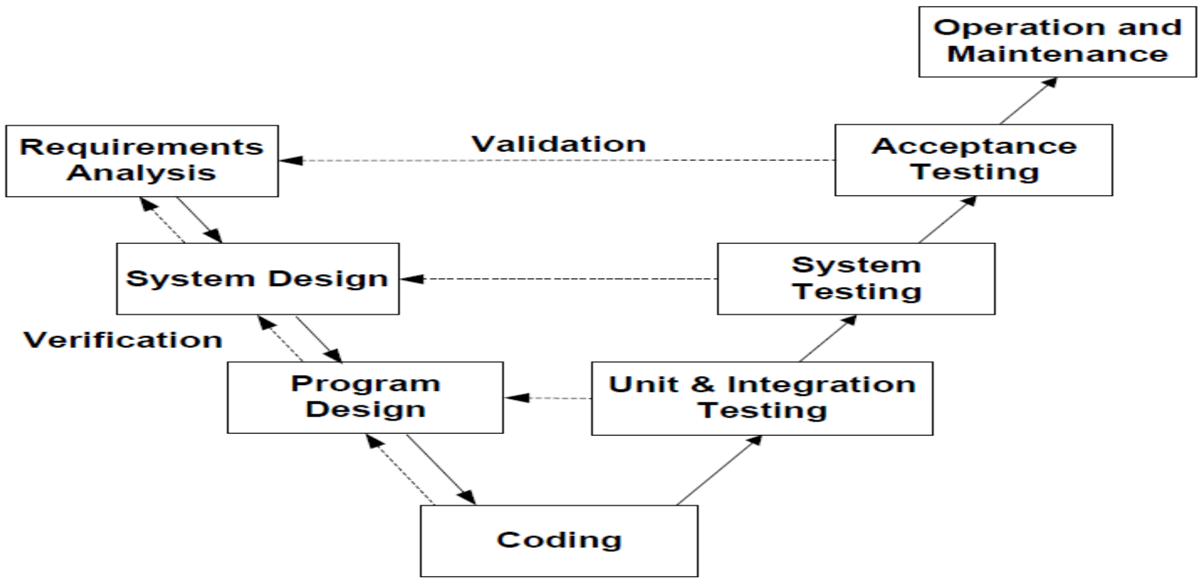


Fig 2.8 V-Model 97 methodology (adapted from [20]

● Spiral Model

Barry Boehm introduced the Spiral model in 1986.[21]. The model is intended for large and complicated software development project. It combines waterfall, rapid prototyping, iterative development, and risk-driven approach to guide the software development process. The Spiral model consists of four cycle's i.e.

● Determine the objectives, alternatives and constraints

● Evaluate the alternatives, identify and resolve risks

● Develop and test the software

● Plan the next iteration

It assumes that each cycle progresses through the same sequence of steps which allows for selection of a process model or termination of the project based on the identified risk. If the risk is relatively low, a waterfall approach could be chosen, otherwise incremental release of software should be performed. Each cycle is completed by product stakeholder review [21].

2.5.2 Agile Software Development Methodologies

In contrast to the plan-driven methodology which emphasizes planning, Agile software development methodology put emphasis on people, communication, adaptability to change, and iterative and incremental development. There are many variants of Agile methodologies. However, only four major Agile methodologies will be presented in this
Thesis, namely Extreme Programming, Scrum, Lean, Scaled Agile Framework®, or SAFe®. None of these Agile methodologies as described by their originator is perfect for every organization. Any methodology may be a good starting point, but software developers will need to adapt the practices to fit the unique circumstances of their organization, individuals, and industry [22].

• Extreme Programming (XP)

Extreme Programming or XP is one of initial Agile approaches that has been proposed after the problem of long development cycle of traditional development models. The idea was developed by Kent Beck and Ward Cunningham in the late 1990s [3]. The starting point of the XP is doing the simplest things to get the job done. It uses the practices that have been proved to be effective in software development. XP aims to address the specific needs of software development performed by small teams facing vague and changing requirements, which are hard to handle by conventional software development.
After several trials and adjustments of practices, XP was proposed as a discipline to help people develop high quality software by following key values and practices [7]. The four values are communication, simplicity, feedback and courage. These values are realized through a set of individual 12 practices taken from Beck’s book as follows [7]:

1. **Planning game**

Planning the next releases features is performed by having developers estimate the effort needed for the customer stories implementation. Then the customers decide about the scope and release time. This emphasizes the close interaction between customers and developers.

2. **Small releases**

Release the software often to the customer with small incremental versions. New version of products is released at least monthly or can be even daily.

3. **Metaphor**

The metaphor is a simple story shared between customers and developers of how the system works.

4. **Simple design**

Design the simplest solution that is workable at that time and constantly evolves to add needed flexibility. Useless complexity and unessential code should be removed.

5. **Testing**

Test driven development is the key which means the developers write unit tests before the production code. The unit tests must run perfectly for development to continue and be kept running at all times. Customers write the functional tests to test the stories.

6. **Refactoring**

Restructure the system without changing its behaviour by removing duplication complexity from code, improving communication, simplifying, and adding flexibility.

7. **Pair programming**

Two programmers write all production code together at a single computer. One writes the code and, at the same time, another reviews the code for correctness and clarity.

8. **Collective ownership**

Every developer owns the code. Therefore, they can change any part code in the system at any time.

9. **Continuous integration**

Build and integrate the system several times a day whenever the task is completed.

10. **40-hour week**

Work no more than 40 hours a week as a rule. Never work overtime for two consecutive weeks.

11. **On-site customer**

Include a real customer that can work with the development team and is available full-time to help defines the system and answers questions.

12. **Coding standards**

Developers write all codes in accordance with rules.

Ideally, XP project starts with a short development phase, followed by a long period of simultaneous improvement and maintenance, and lastly retirement when the project no longer makes sense. XP project life cycle consists of five phases, prior to final release (Death Phase) as shown in Figure 2.9

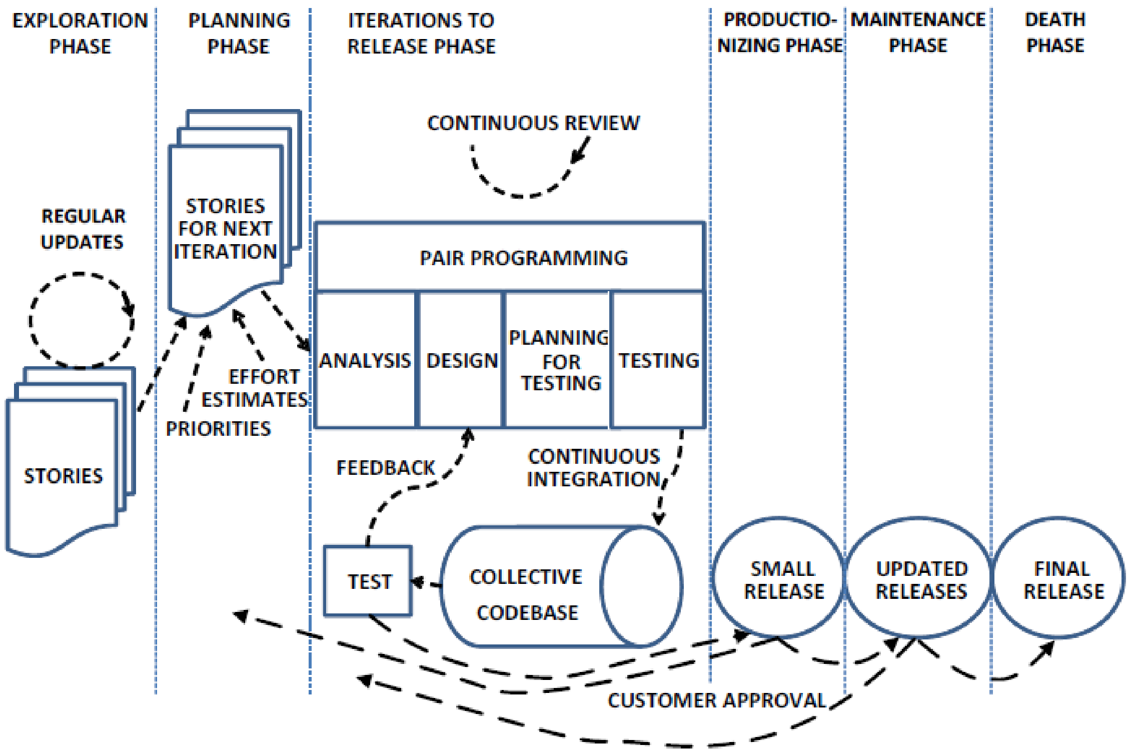


Fig 2.9 XP Phases [23]

Each **XP Phase** is described below according to Beck [7]:

1. Exploration Phase

This is the pre-production phase where the customer writes down the story cards containing the features of the system they expected to have in the first release.

Concurrently, the programmers familiarize themselves with technology and tools planned to be used in the development. The project teams explore the possibilities for the system architecture by building prototypes of a system and testing them in a different way. This phase is done when the customer is confident that he/she can write good enough story cards to be implemented. On the other side, programmers are sure that they are acquainted with the practices, technology, and can confidently estimate the effort required by the story. This phase normally takes a few weeks to a few months.

2. Planning Phase

The story cards are prioritized, and the scope of the first small release is defined by the customer with the agreement from the developers. The developers estimate efforts needed to implement each story card and then the schedule is agreed upon.

The period of each release is normally around two months but can be up to six months for the first release. The planning phase itself lasts a couple of days.

3. Iteration to Release Phase

There are several iterations before the first release and each of which takes one to four weeks to implement. As a general rule, customer decides the stories to be implemented in the first iteration that will build the architecture of the whole system.

Customer also selects each iteration stories. Customer creates functional tests, which will be run at the end of iteration. When the last iteration is reached and finished, the system is prepared for production.

4. Productionizing Phase

This phase entails extra testing of the system performance before releasing to the customer. Change management is needed as new changes may be identified and decisions have to be made. The requirements or ideas that are decided to be postponed will be documented for later implementation during the maintenance phase.

5. Maintenance Phase

After releasing the system to the customer for use, the project must maintain the running system and make sure it works fine while also working on new iterations.
This is what maintenance phase is about. In general, extra efforts are required to support customers as well as incorporate new people into the team and change the team structure.

● Scrum

Scrum is a simple and adaptive framework used for managing software development project. It helps in organizing the team and getting work done productively with higher quality [24]. Scrum was introduced by Jeff Sutherland at Easel Corporation in 1993 and then formalized for the software industry by Ken Schwaber in 1995. The term "Scrum" was originally derived from the Scrum formation in Rugby. This term was identified by
Takeuchi and Nonaka in their works when reviewing best business practices for building new product in the Japanese automobile in 1986. It was used as the foundation of team building [8] [24].
Even though Scrum was originally proposed for managing product development projects, it has been used largely for software project management counted as an Agile software development approach. By using Scrum, a team can deliver software to the customer faster. More energy focus and transparency will be added to the project planning and implementation. The following things can be achieved by Scrum implementation [24]:
● Individual objectives aligned with corporate objectives
● A culture driven by performance
● Shareholder value creation
● Stable and consistent communication of performance
The Scrum framework contains three roles, three artefacts and four ceremonies. It is designed to deliver functional software in Sprints or about 30 days iterations [24] [25].

Scrum Roles

All management responsibilities in a project are divided among these three roles [8] [24] [25]:

1. The **Product Owner** is responsible for exemplifying the interests of all stakeholders in the project by defining the product features documented as the product requirements. He or she is responsible for the profitability of the product (ROI) as well as creating release plan. The list of requirements is called a Product Backlog which the Product Owner will use to ensure that the functionality is produced as prioritized according to the market value. However, the Product Owner can change features and its priority during each sprint and he or she is the one who accepts or rejects work outcome.

2. The **Scrum Master** is a facilitator for Scrum project. He or she is responsible for the project success by ensuring that the team is fully functional, productive and Scrum process is followed correctly. Scrum Master also helps the Product Owner selects the most valuable Product Backlog and helps the team to turn the backlog into functionality. Besides, he or she supports close collaboration across all roles and functions as well as protects them from external interference.

3. The **Team** is self-organizing and cross functional team which contains seven plus/minus two members. They are responsible for developing the functionality of each iteration and each project as a whole. They select the Sprint goal, specify work results and have the right to do everything within the project to reach the Sprint goal. The team manages itself and its works.

Scrum Artefacts

Scrum brings up three new artefacts, which are used throughout the Scrum process as following [24] [25]:

1. Product Backlog

It is a list of the system requirements being developed by the development teams while its content and prioritization are the responsibility of the product owner. In the project planning, the product backlog is used as an initial estimation of the requirements. It exists as long as the product exists and grows as the product grows. The Product backlog is constantly changed to identify what can make the product valuable and competitive.

2. Sprint Backlog

Sprint backlog describes the work that a team selects from the product backlog to be implemented in the Sprint. It is a real-time picture of the work that the team plans to accomplish during the sprint. In general, the work is divided into individual tasks that will take roughly four to sixteen hours to finish. The Sprint backlog can be changed only by Product Owner.

3. Burn Down Chart

Burn down Chart is used as a tool to help the development team to successfully complete a Sprint on time by delivering working and shippable software product.
It shows the remaining tasks to be done in the Sprint backlog.

Scrum Ceremonies

The Scrum framework contains four ceremonies, which are Sprint Planning, Sprint Review, Daily Scrum Meeting, and Sprint Retrospective meeting [24] [25]. Following figure describes these Scrum ceremonies in a Scrum process.

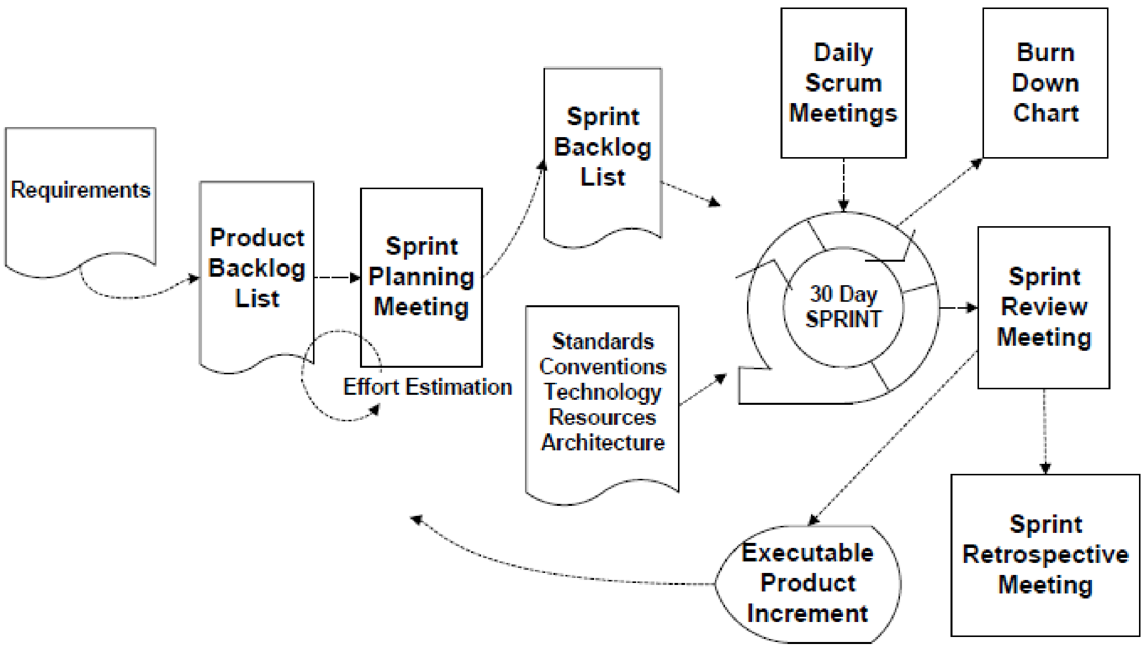


Fig 2.10 Scrum processes (adapted from [23].

All works are performed in a Sprint which is an iteration of 30 days. Each Sprint starts with a Sprint planning meeting. This is the activity where the Product Owner works together with the team on the tasks need to be done in the next Sprint. The tasks are prioritized and selected from the Product Backlog.

Sprint Planning meetings are normally time-boxed and last about eight hours or less.

There are two parts: the first half, about four hours, are for the Product Owner to present the Product Backlog with high priority to the team and the team queries about the detail and intentions of the Product Backlog [25]. After that, the team selects Product Backlog that they think they can implement and ship to the customer by the end of the Sprint. For the second half of the Sprint Planning meeting, the team reforms the Sprint and creates a tentative plan to start the Sprint. The tasks decided to be implemented as plan are put in a Sprint Backlog and appear as the Sprint evolves [24].

The second ceremony that occurs every day is called a Daily Scrum meeting. It allows team members to get together for about 15 minutes so that they can synchronize all the works and schedule any future meetings if necessary [25]. At the Daily Scrum meeting, each member answers three questions about the project: *What have been done since the last Daily Scrum meeting? What do you plan to do until the next Daily Scrum meeting? What obstacles do you face on during your works?*

The third ceremony is called a Sprint Review meeting. This time-boxed meeting lasting about four-hours is for the team to present to the product owner and other stakeholders about the functionalities developed during the Sprint [24] [25]. The purpose of this informal meeting is to get people together so that they help the team decides what to do next. After the Sprint review meeting, the Scrum Master conducts a Sprint Retrospective meeting [25]. This meeting is for the team to reflect and improve its development process to make it more effective for the next Sprint

- **Lean**

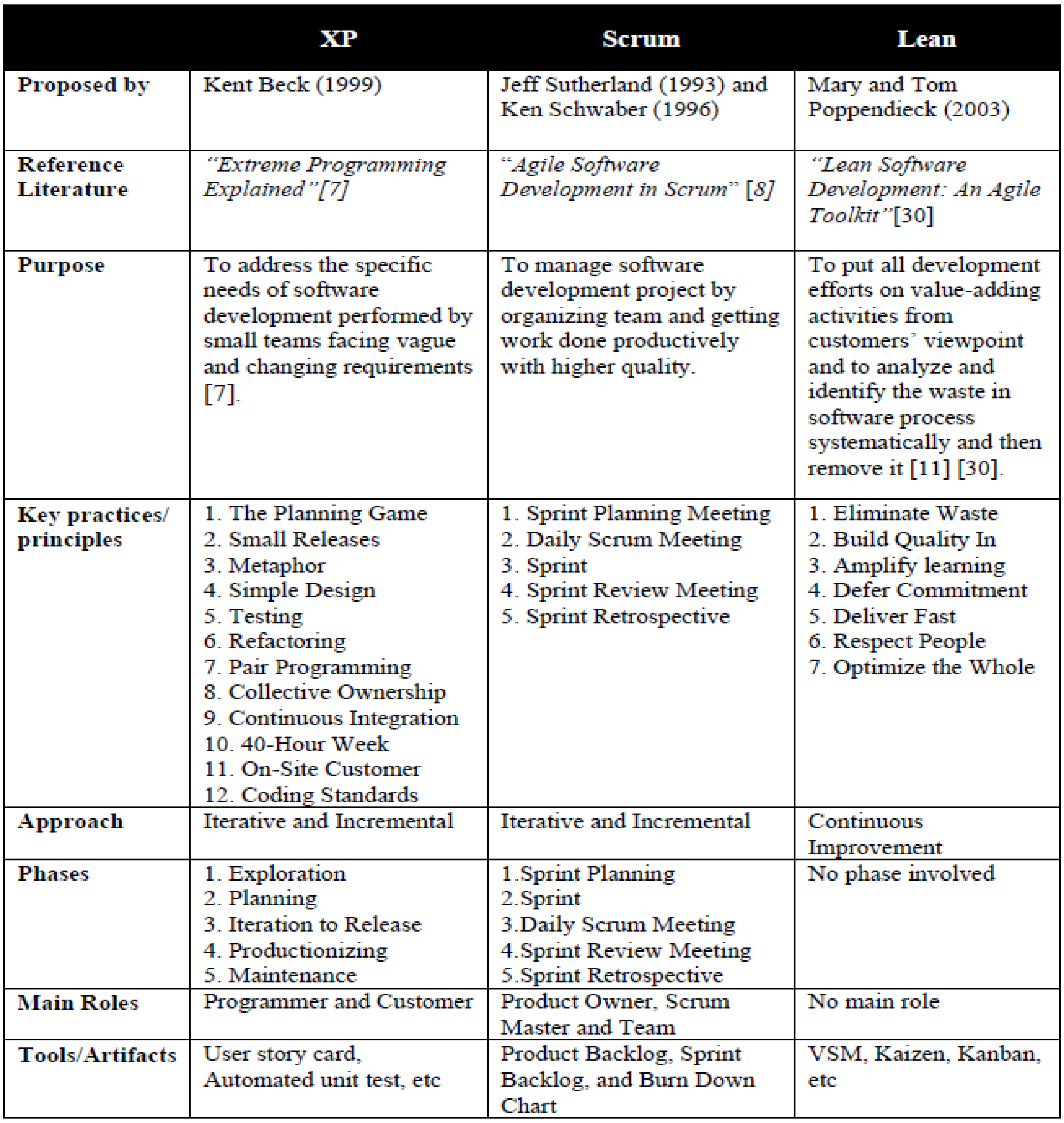
Since the success of Lean concept in Toyota Production System which made Toyota produce high-quality cars with the lowest cost and shortest time, principles of Lean were then brought into other areas including Software Development domain. In particular, the attention got significantly higher after a book on Lean Software Development by Mary and Tom Poppendieck was published [30]. The key ideas behind Lean are to put all development efforts on the value-adding activities from the customers‟ viewpoint and to analyse and identify the waste in software process systematically and then remove it [11] [30]. The seven principles of Lean are described in the following table [31] [32].

Principle	Description
Eliminate Waste	Eliminate everything in software development process that does not contribute to the value for the customer e.g. partially done work, extra features, extra process, handovers (e.g. documentation), task switching, delays, and defects.
Build Quality In	Software quality should be built in as early as possible and not late by fixing the defects that testing found.
Amplify learning	Processes and practices in companies should support learning. Learning here includes getting a better understanding of customer needs, good testing strategies, and so on.
Defer Commitment	A commitment should be delayed as late as possible for irreversible decisions. For instance, a complex architectural decision might require some experimentation and therefore should not be committed early.
Deliver Fast	Create short product development cycle time by minimizing the time from receiving a request for a feature to the delivery of the feature.
Respect People	Poppendieck gave three principles that were used in the context of Toyota case. First, entrepreneurial leadership, which means people that led by managers who trust and respect them are more likely to become good leaders themselves. Second, expert technical workforce, which means that successful companies should make sure that the necessary expertise for achieving a task is within the teams. Third, responsibility-based planning and control, which means management should trust their teams and not tell them how to get the job done. Moreover, it is important to provide the teams with reasonable and realistic goals.
Optimize the Whole	When improving the process of software development the whole value-stream needs to be considered end to end.

Table 2.11 Seven principles of Lean software development [31]

Lean software development is closely related to Agile software development approaches. Lean itself is a mind-set, a way of thinking about how to deliver value to customer more quickly by finding and eliminating waste (the impediments to quality and productivity) [33]. Lean software development, which Poppendieck defined as Agile toolkit, is slightly different from their equivalents in Agile software development, but they are parallel [30]. When applying Lean software development principle, it is quite common to select a lightweight Agile software approach as a starting point and begin applying Lean from there. Lean tools include Value Stream Mapping (VSM), Kaizen (continuous improvement), Kanban (a signalling system used to signal the need for an item, typically using things like index cards, coloured golf balls, or empty carts), etc. [33].

Summary comparison of main Agile software development approaches (XP, Scrum, and Lean) can be found in the following figure.



	XP	Scrum	Lean
Proposed by	Kent Beck (1999)	Jeff Sutherland (1993) and Ken Schwaber (1996)	Mary and Tom Poppendieck (2003)
Reference Literature	*"Extreme Programming Explained" [7]*	*"Agile Software Development in Scrum"* [8]	*"Lean Software Development: An Agile Toolkit"* [30]
Purpose	To address the specific needs of software development performed by small teams facing vague and changing requirements [7].	To manage software development project by organizing team and getting work done productively with higher quality.	To put all development efforts on value-adding activities from customers' viewpoint and to analyze and identify the waste in software process systematically and then remove it [11] [30].
Key practices/ principles	1. The Planning Game 2. Small Releases 3. Metaphor 4. Simple Design 5. Testing 6. Refactoring 7. Pair Programming 8. Collective Ownership 9. Continuous Integration 10. 40-Hour Week 11. On-Site Customer 12. Coding Standards	1. Sprint Planning Meeting 2. Daily Scrum Meeting 3. Sprint 4. Sprint Review Meeting 5. Sprint Retrospective	1. Eliminate Waste 2. Build Quality In 3. Amplify learning 4. Defer Commitment 5. Deliver Fast 6. Respect People 7. Optimize the Whole
Approach	Iterative and Incremental	Iterative and Incremental	Continuous Improvement
Phases	1. Exploration 2. Planning 3. Iteration to Release 4. Productionizing 5. Maintenance	1.Sprint Planning 2.Sprint 3.Daily Scrum Meeting 4.Sprint Review Meeting 5.Sprint Retrospective	No phase involved
Main Roles	Programmer and Customer	Product Owner, Scrum Master and Team	No main role
Tools/Artifacts	User story card, Automated unit test, etc	Product Backlog, Sprint Backlog, and Burn Down Chart	VSM, Kaizen, Kanban, etc

Table 2.12 Summary comparison of main Agile software development approaches

2.5.3 Waterfall vs. Agile: Which is the Right Development Methodology for an ERP Project?

One of the first decisions IT Project Managers face for their project implementations is "Which development methodology should we use?" This is a topic that gets a lot of discussion (and often heated debate). But, before deciding which is more appropriate, it is essentially important to provide a little background on both. Lotz 2013] [40]

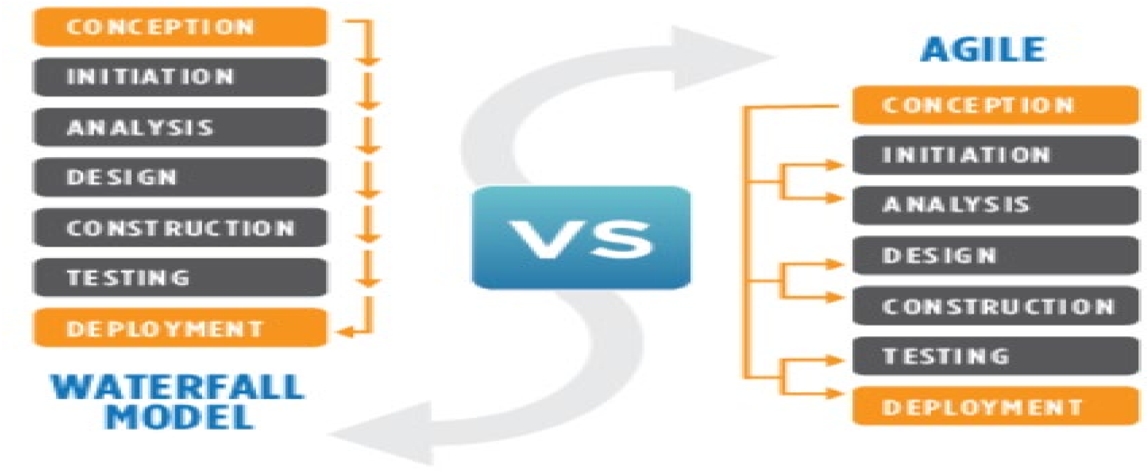


Fig 2.13 Waterfall model v. Agile

Waterfall

A classically linear and sequential approach to software design and systems development, each waterfall stage is assigned to a separate team to ensure greater project and deadline control, important for on-time project delivery. A linear approach means a stage by stage approach for product building, e.g.

1. The project team first **Analyses**, then determining and prioritising business requirements / needs.
2. Next, in the **Design phase** business requirements are translated into IT solutions, and a decision taken about which underlying technology i.e. COBOL, Java or Visual Basic, etc. etc. is to be used.
3. Once processes are defined and online layouts built, code **Implementation** takes place.
4. The next stage of data conversion evolves into a fully **Tested** solution for implementation and testing for evaluation by the end-user.
5. The last and final stage involves **Evaluation** and **Maintenance,** with the latter ensuring everything runs smoothly.

However, in case a glitch should result, changing the software is not only a practical impossibility, but means one has to go right back to the beginning and start developing new code, all over again.

Agile

It is a low over-head method that emphasizes values and principles rather than processes. Working in cycles i.e. a week, a month, etc., project priorities are re-evaluated and at the end of each cycle.

Four principles that constitute Agile methods are: [40]

1. The reigning supreme of individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over plan follow-throughs.

To synopsise the difference between the two, one can say the classic waterfall method stands for predictability, while Agile methodology spells adaptability.

Agile methods are good at reducing overheads, such as, rationale, justification, documentation and meetings, keeping them as low as is possible. And, that is why Agile methods benefit small teams with constantly changing requirements, rather more than larger projects.

Agile, based on empirical rather than defined methods (Waterfall) is all about light manoeuvrability and sufficiency for facilitating future development. By defined methods what one means is that one plans first and then enforces these plans. However, Agile methods involve planning what one wants and then adapting these plans to the results.

Extreme Programming (XP) is an excellent example of Agile methodology i.e.:

1. Communication between customers and other team members;
2. Simple, clean designs.
3. Feedback given on Day 1 of software testing.
4. Early delivery and implementation of suggested changes.

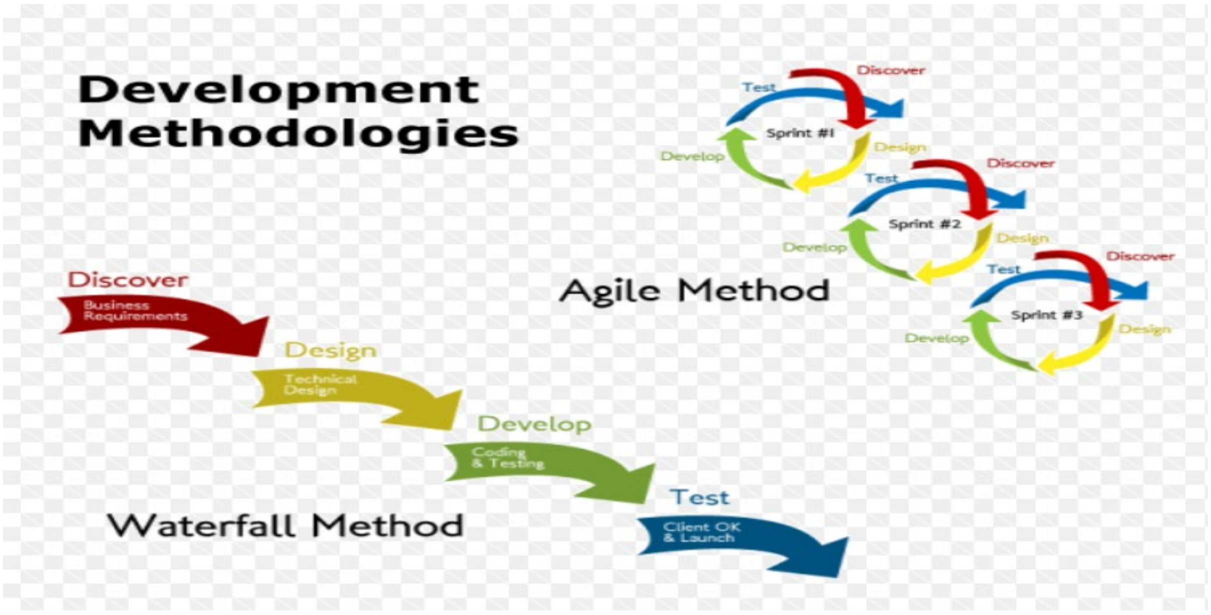


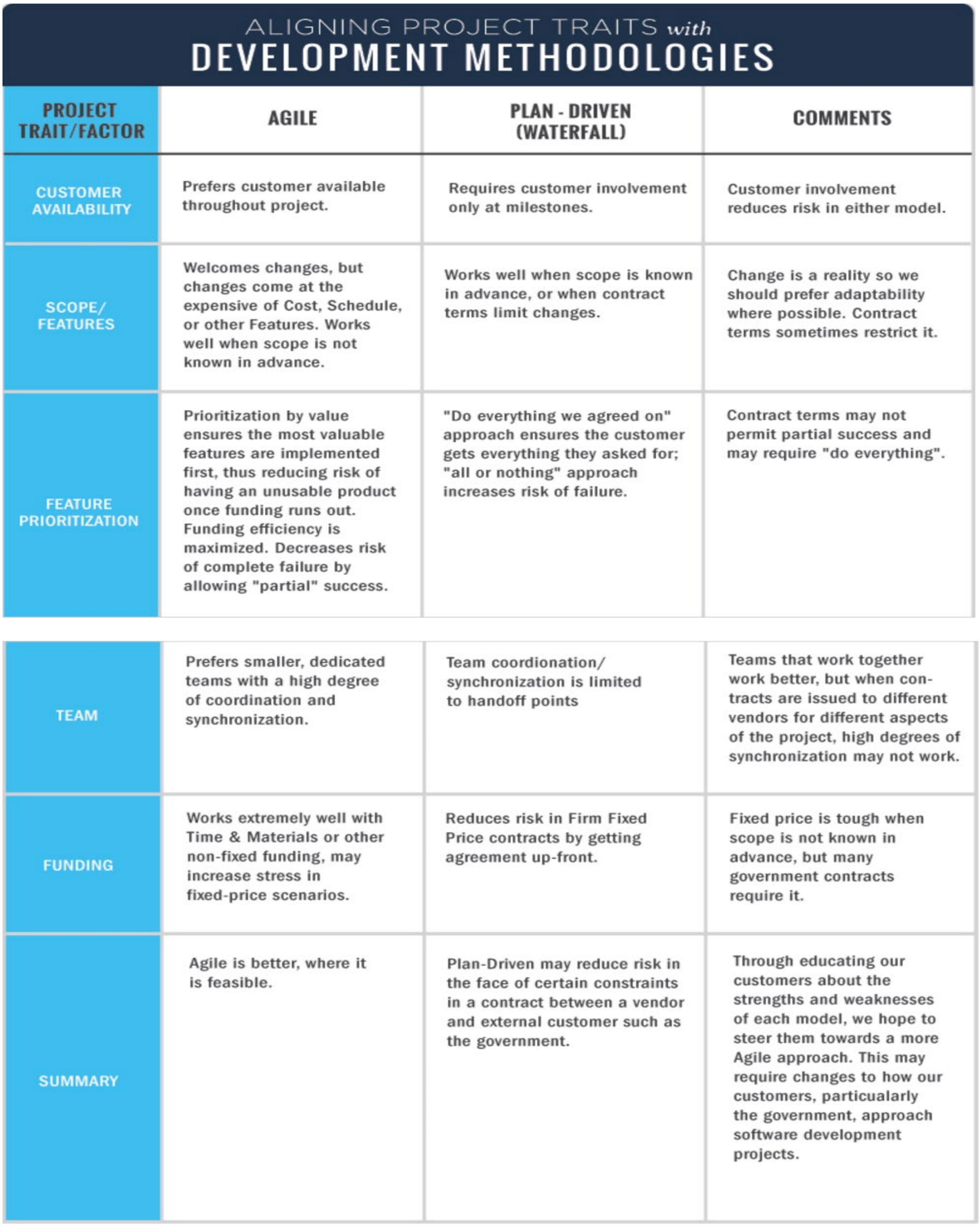
Fig 2.14 Development Methodologies

Agile methodology means cutting down the big picture into puzzle size bits, fitting them together when the time is right e.g. design, coding and testing bits. So, while there are reasons to support both the waterfall and agile methods, however, a closer look clarifies why many software and web design firms make the more appropriate choice of employing Agile methodology.

The following points enumerates the raison d'être for choosing Agile methodology over the Waterfall method.

1. Once a stage is completed in the **Waterfall method,** there is no going back, since most software designed and implemented under the waterfall method is hard to change according to time and user needs. The problem can only be fixed by going back and designing an entirely new system, a very costly and inefficient method. Whereas, **Agile methods** adapt to change, as at the end of each stage, the logical programme, designed to cope and adapt to new ideas from the outset, allows changes to be made easily. With Agile, changes can be made if necessary without getting the entire programme rewritten. This approach not only reduces overheads, it also helps in the upgrading of programmes.

2. Another **Agile method** advantage is one has a launchable product at the end of each tested stage. This ensures bugs are caught and eliminated in the development cycle, and the product is double tested again after the first bug elimination. This is not possible for the **Waterfall method,** since the product is tested only at the very end, which means any bugs found results in the entire programme having to be re-written.
3. **Agile's** modular nature means employing better suited object-oriented designs and programmes, which means one always has a working model for timely release even when it does not always entirely match customer specifications. Whereas, there is only one main release in the waterfall method and any problems or delays mean highly dissatisfied customers.
4. Agile methods allow for specification changes as per end-user's requirements, spelling customer satisfaction. As already mentioned, this is not possible when the waterfall method is employed, since any changes to be made means the project has to be started all over again.
5. However, both methods do allow for a sort of departmentalization e.g. in waterfall departmentalization is done at each stage. As for Agile, each coding module can be delegated to separate groups. This allows for several parts of the project to be done at the same time, though departmentalization is more effectively used in Agile methodologies.



ALIGNING PROJECT TRAITS *with*

DEVELOPMENT METHODOLOGIES

PROJECT TRAIT/FACTOR	AGILE	PLAN - DRIVEN (WATERFALL)	COMMENTS
CUSTOMER AVAILABILITY	Prefers customer available throughout project.	Requires customer involvement only at milestones.	Customer involvement reduces risk in either model.
SCOPE/ FEATURES	Welcomes changes, but changes come at the expensive of Cost, Schedule, or other Features. Works well when scope is not known in advance.	Works well when scope is known in advance, or when contract terms limit changes.	Change is a reality so we should prefer adaptability where possible. Contract terms sometimes restrict it.
FEATURE PRIORITIZATION	Prioritization by value ensures the most valuable features are implemented first, thus reducing risk of having an unusable product once funding runs out. Funding efficiency is maximized. Decreases risk of complete failure by allowing "partial" success.	"Do everything we agreed on" approach ensures the customer gets everything they asked for; "all or nothing" approach increases risk of failure.	Contract terms may not permit partial success and may require "do everything".
TEAM	Prefers smaller, dedicated teams with a high degree of coordination and synchronization.	Team coordination/ synchronization is limited to handoff points	Teams that work together work better, but when con-tracts are issued to different vendors for different aspects of the project, high degrees of synchronization may not work.
FUNDING	Works extremely well with Time & Materials or other non-fixed funding, may increase stress in fixed-price scenarios.	Reduces risk in Firm Fixed Price contracts by getting agreement up-front.	Fixed price is tough when scope is not known in advance, but many government contracts require it.
SUMMARY	Agile is better, where it is feasible.	Plan-Driven may reduce risk in the face of certain constraints in a contract between a vendor and external customer such as the government.	Through educating our customers about the strengths and weaknesses of each model, we hope to steer them towards a more Agile approach. This may require changes to how our customers, particualarly the government, approach software development projects.

Table 2.15. Factors to consider when choosing which methodology to use.

2.5.4 Scaled Agile Framework® (SAFe)

The Scaled Agile Framework®, or SAFe®, [41] provides a recipe for adopting Agile at enterprise scale.

As Scrum is to the Agile team, SAFe is to the Agile enterprise.

SAFe tackles the tough issues – architecture, integration, funding, governance and roles at scale. It is field-tested and enterprise friendly.

SAFe is the brainchild of Dean Leffingwell

SAFe is based on Lean and Agile principles

There are three levels in SAFe:
* Team
* Program
* Portfolio

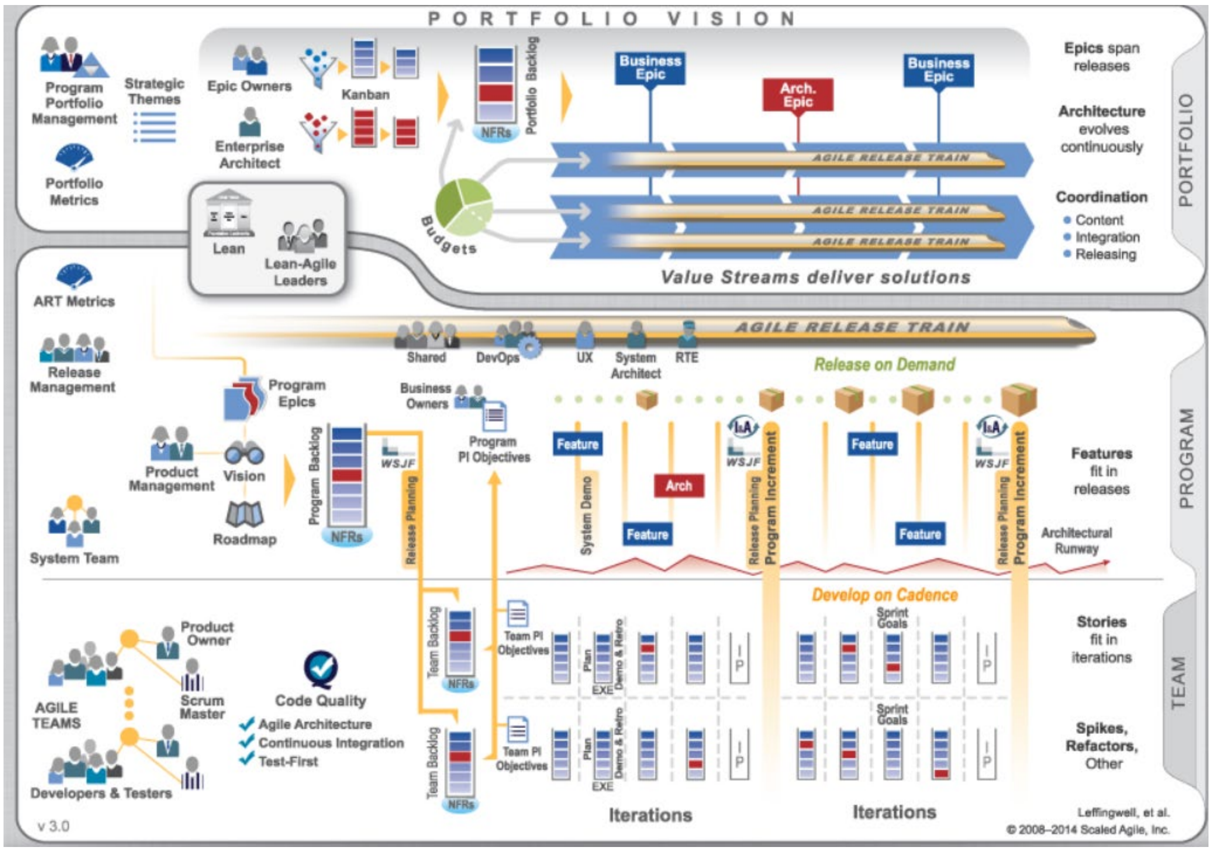


Fig 2.16 SAFe Portfolio Vision [41]

At the Team Level:

- Scrum with XP engineering practices are used.
- Define/Build/Test (DBT) teams deliver working, fully tested software every two weeks. There are five to nine members of each team.

At the Program Level:

- SAFe defines an Agile Release Train (ART). As iteration is to team, train is to program.
- The ART (or train) is the primary vehicle for value delivery at the program level. It delivers a value stream for the organization.
- SAFe is three letter acronym (TLA) heaven – DBT, ART, RTE, PSI, NFR, RMT and I&A!
- Between 5 and 10 teams work together on a train. They synchronize their release boundaries and their iteration boundaries.
- Every 10 weeks (5 iterations) a train delivers a Potentially Shippable Increment (PSI). A demo and inspect and adapt sessions are held. Planning begins for the next PSI.
- PSIs provide a steady cadence for the development cycle. They are separate from the concept of market releases, which can happen more or less frequently and on a different schedule.

New program level roles are defined
* System Team
* Product Manager
* System Architect
* Release Train Engineer (RTE)
* UX and Shared Resources (e.g., security, DBA)
* Release Management Team

In IT/PMI environments the Program Manager or Senior Project Manager might fill one of two roles. If they have deep domain expertise, they are likely to fill the Product Manager role. If they have strong people management skills and understand the logistics of release, they often become the Release Train Engineer.

SAFe defines a Scaled Agilist (SA) certification program for executives, managers, architects and change agents responsible for leading SAFe implementations.

SAFe makes a distinction between content (what the system does) and design (how the system does it). There is separate "authority" for content and design.
The Product Manager (Program Manager) has content authority at the program level. She defines and prioritizes the program backlog.
SAFe defines an artifact hierarchy of Epics – Features – User Stories. The program backlog is a prioritized list of features. Features can originate at the Program level, or they can derive from Epics defined at the Portfolio level. Features decompose to User Stories which flow to Team-level backlogs.
Features are prioritized based on Don Reinersten's Weighted Shortest Job First (WSJF) economic decision framework.
The System Architect has design authority at the program level. He collaborates day to day with the teams, ensuring that Non-Functional Requirements (NFRs) are met. He works with the enterprise architect at the portfolio level to ensure that there is sufficient architectural runway to support upcoming user and business needs.
The UX Designer(s) provides UI design, UX guidelines and design elements for the teams. In a similar manner, shared specialists provide services such as security, performance and database administration across the teams.

The Release Train Engineer (RTE) is the Uber-ScrumMaster.
The Release Management Team is a cross-functional team - with representation from marketing, dev, quality, ops and deployment – that approves frequent releases of quality solutions to customers.

At the Portfolio Level:

PPM has a central role in Strategy, Investment Funding, Program Management and Governance.

Investment Themes drive budget allocations.

Themes are done as part of the budgeting process with a lifespan of 6-12 months.

Portfolio philosophy is centralized strategy with local execution.

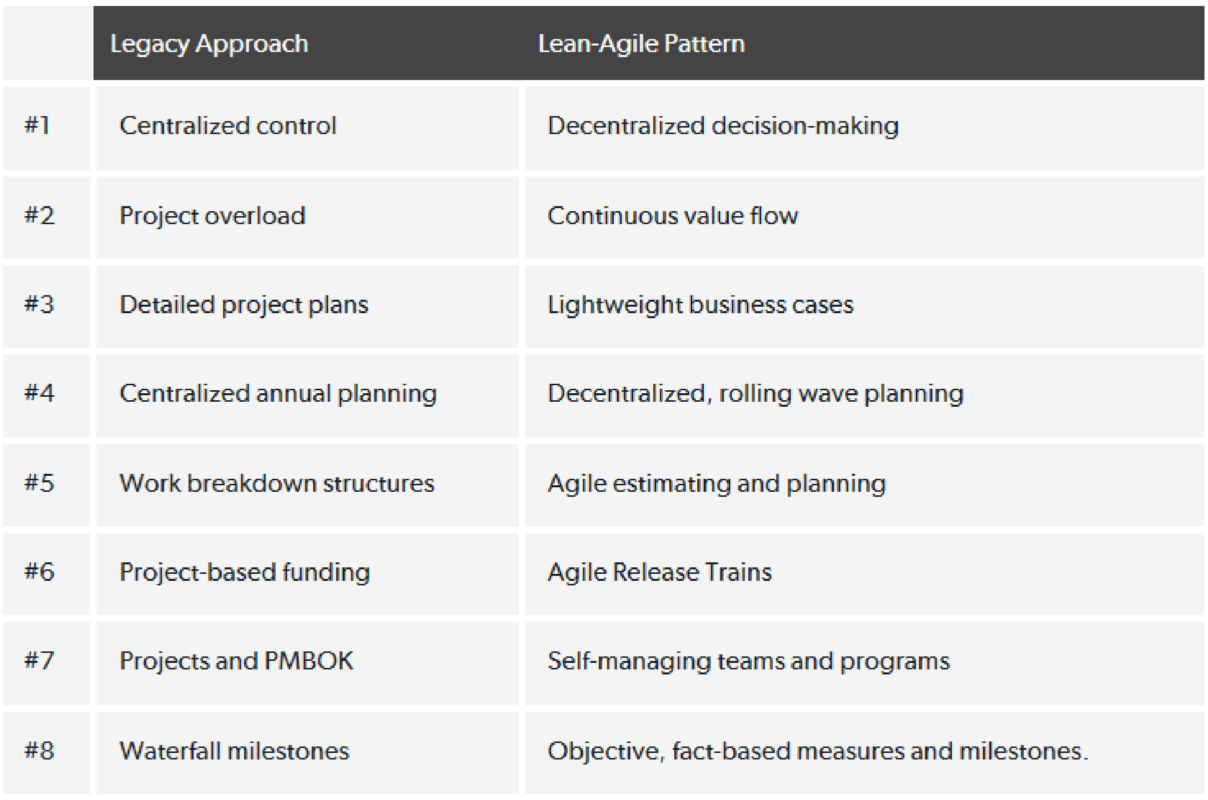
Epics define large development initiatives that encapsulate the new development necessary to realize the benefits of investment themes.

There are business epics (customer-facing) and architectural epics (technology solutions).

Business and architectural epics are managed in parallel Kanban systems.

Objective metrics support IT governance and continuous improvement.

Enterprise architecture is a first-class citizen. The concept of Intentional Architecture provides a set of planned initiatives to enhance solution design, performance, security and usability.



	Legacy Approach	Lean-Agile Pattern
#1	Centralized control	Decentralized decision-making
#2	Project overload	Continuous value flow
#3	Detailed project plans	Lightweight business cases
#4	Centralized annual planning	Decentralized, rolling wave planning
#5	Work breakdown structures	Agile estimating and planning
#6	Project-based funding	Agile Release Trains
#7	Projects and PMBOK	Self-managing teams and programs
#8	Waterfall milestones	Objective, fact-based measures and milestones.

able above reproduced with permission of Leffingwell LLC and Scaled Agile Inc.

Table 2.17 SAFe patterns provide a transformation roadmap [41}

Adoption

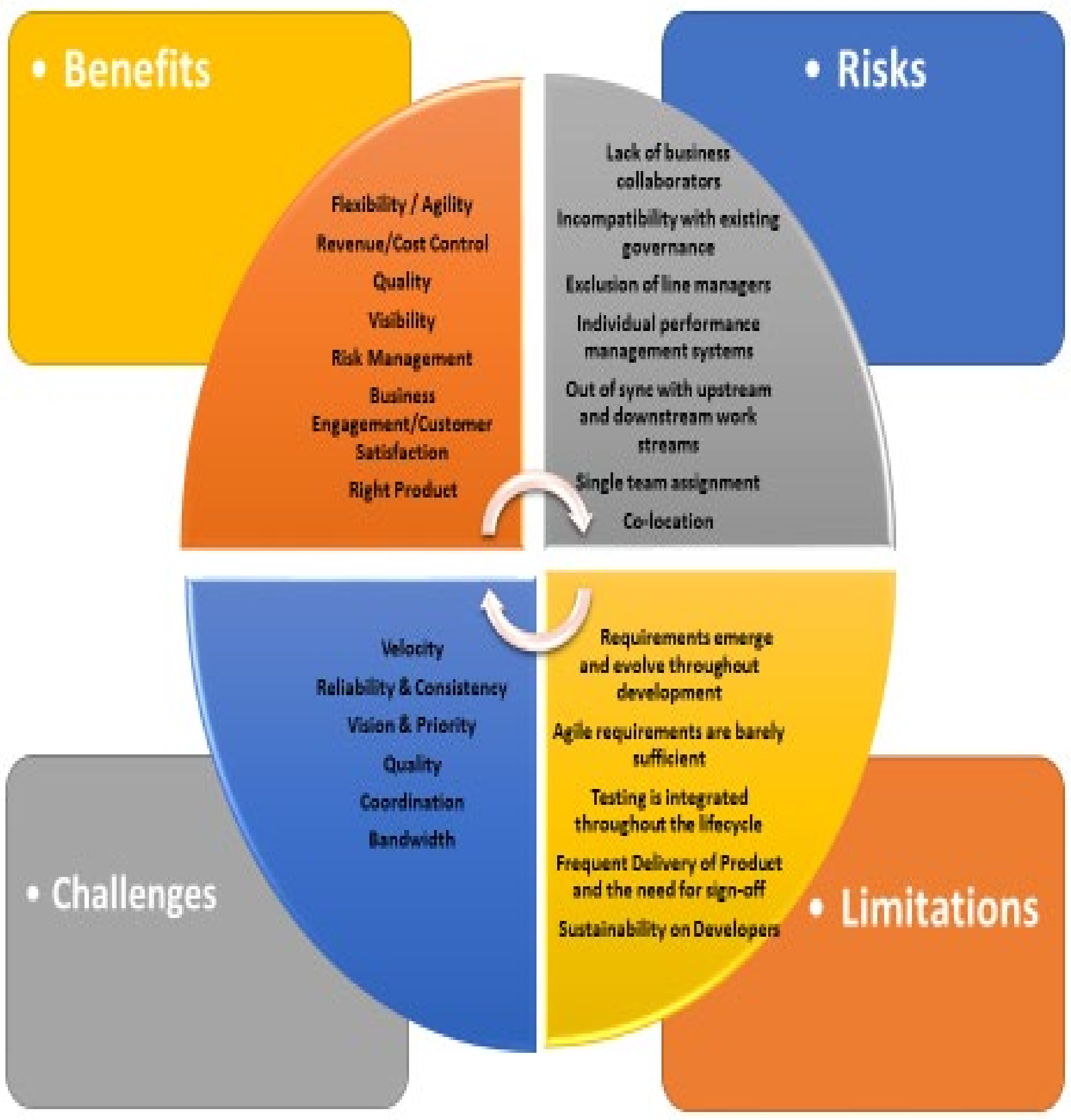
Adoption focuses on identifying a value stream. A value stream is a sequence of activities intended to produce a consistent set of deliverables of value to customers. Value streams are realized via an Agile Release Train (ART).

SAFe poses questions to help identify value streams (ARTs):

* What program might adopt the new process the fastest?
* Which executives are ready for a transition?
* What are the geographical locations and how are the team members distributed?
* What programs are the most challenged, or represent the biggest opportunities?

When you identify a value stream, you go "All In" and "All at Once" for that train.

2.5.5 Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program.



• Benefits
Flexibility / Agility
Revenue/Cost Control
Quality
Visibility
Risk Management
Business Engagement/Customer Satisfaction
Right Product
• Risks
Lack of business collaborators
Incompatibility with existing governance
Exclusion of line managers
Individual performance management systems
Out of sync with upstream and downstream work streams
Single team assignment
Co-location
Velocity
Reliability & Consistency
Vision & Priority
Quality
Coordination
Bandwidth
• Challenges
Requirements emerge and evolve throughout development
Agile requirements are barely sufficient
Testing is integrated throughout the lifecycle
Frequent Delivery of Product and the need for sign-off
Sustainability on Developers
• Limitations

Fig 2.18 Benefits, Risks, Challenges, and Limitations of Agile in an ERP Program

Benefits

As outlined by Nah, F. F., Lau, J. L., & Kuang, J. (2001) [2}. Critical factors for successful implementation of enterprise systems. *Business Process Management Journal and* Fergal Carton, Frederic Adam and David Sammon Business Information Systems, University College Cork, Cork, Ireland (2007) in "Project management: a case study of a successful ERP implementation" [16]

- **Revenue/Cost Control**
  The iterative nature of agile development means features are delivered incrementally, enabling some benefits to be realised early as the product continues to develop. The scope of the product and its features are variable, rather than the cost.
- **Quality**
  A key principle of agile development is that testing is integrated throughout the lifecycle, enabling regular inspection of the working product as it develops. This allows the product owner to make adjustments if necessary and gives the product team early sight of any quality issues.
- **Visibility**
  Agile development principles encourage active 'user' involvement throughout the product's development and a very cooperative collaborative approach. This provides excellent visibility for key stakeholders, both of the project's progress and of the product itself, which in turn helps to ensure that expectations are effectively managed.
- **Risk Management**
  Small incremental releases made visible to the product owner and product team through its development help to identify any issues early and make it easier to respond to change. The clear visibility in agile development helps to ensure that any necessary decisions can be taken at the earliest possible opportunity, while there's still time to make a material difference to the outcome
- **Flexibility / Agility**
  In traditional development projects, we write a big spec up-front and then tell business owners how expensive it is to change anything, particularly as the project goes on. In fear of scope creep and a never-ending project, we resist changes and put people through a change control committee to keep them to the essential minimum. Agile development principles are different. In agile development, change is accepted. *In fact, it's expected*. Because the one

thing that's certain in life is change. Instead the timescale is fixed and requirements emerge and evolve as the product is developed. Of course, for this to work, it's imperative to have an actively involved stakeholder who understands this concept and makes the necessary trade-off decisions, trading existing scope for new.

- **Business Engagement/Customer Satisfaction**
  The active involvement of a user representative and/or product owner, the high visibility of the product and progress, and the flexibility to change when change is needed, create much better business engagement and customer satisfaction. This is an important benefit that can create much more positive and enduring working relationships.
- **Right Product**
  The ability for agile development requirements to emerge and evolve, and the ability to embrace change (with the appropriate trade-offs), the team build the right product. It's all too common in more traditional projects to deliver a "successful" project in IT terms and find that the product is not what was expected, needed or hoped for. In agile development, the emphasis is absolutely on building the right product.

Risks

As outlined by Huang, S., Chang, I., Li, S., & Lin, M. (2004).[12] in Assessing risk in ERP projects: Identify and prioritize the factors. *Industrial Management & Data Systems*

- **Lack of available business collaborators**
  Each team needs a single point of contact business representative (a.k.a. sponsor or customer). Known as the "product owner" in Scrum, this person needs to frequently engage with the team and have the authority to specify, prioritize, and accept the team's work. Not properly fulfilling this role will curtail the efficiency of your Agile initiative, even kill it.
- **Incompatibility with existing governance policies**
  Quality gate sign-off policies. Documents such as project plans, requirements, and design documents may not be amendable after formal sign-off from upper management (without formal change requests). However, Agile won't work if teams and their product owners are not empowered to make changes to tasks, estimates, priority, and design.

- **Exclusion of supervisors and line managers**
  When teams are empowered, supervisors sometimes lose job responsibilities. Yet too often organizations make no provisions for addressing these changes in line manager function and status. This can result in active and (less addressable) passive resistance to the Agile transformation from an important and influential constituency.
- **Individual performance management systems**
  The Agile value is commitment to the team, where all “win” or “lose” together. This is diametrically opposed to "Individual" performance ranking meritocracy systems currently in place across organisations
- **Out of phase with upstream and downstream work streams**
  The Agile team is somewhere in the middle of a larger flow of work or value stream. It increases frequency of inflow and outflow to every two weeks, maybe even less. If the cadence of work upstream and downstream from the team does not also change, then delays and log jams can occur.
- **Single team assignment**
  People who are simultaneously committed to more than one “Team” are unable to make meaningful delivery commitments because of inevitable conflicting team imperatives. If members can’t make meaningful commitments, Agile won’t work. This problem grows in magnitude as the program grows in size and more teams are involved.
- **Co-location**
  Collaboration thrives when teams are in intimate proximity. Maybe an exceptional team can overcome this challenge, but not the many teams of a larger Agile transformation. Once teams get established and members have effective working relationships, the team has a greater chance of surviving if some members are dispersed. Also, an Agile program involving many teams does not need to be completely co-located in order to be successful. What is vital is that small groups of people intimately work together and develop the trust and mutual understanding necessary for effective collaboration. It will be extremely difficult for small groups to achieve this state if they are never afforded the opportunity to work in a shared, physical space

Challenges

As outlined by Maditinos, D., Chatzoudes, D., & Tsairidis, C. (2012) [10] in their Factors affecting ERP system implementation effectiveness. *Journal of Enterprise Information Management*

- **Velocity**
  Whether real or only perceived, the velocity challenge indicates that "the business" does not believe that they are getting adequate throughput / value from the technology team.
- **Reliability & Consistency**
  This challenge is typically shown through Agile teams that consistently over-commit and miss their Sprint goals and deliverables
- **Vision & Priority**
  An Agile team that lacks proper business or technical vision and priority is constantly changing directions for each Sprint, or worse yet within a single Sprint
- **Quality**
  Teams that have a quality challenge "finish" their sprints and deliverables, but accumulate a significant amount of technical debt in the form of defects and poor design choices
- **Coordination**
  A coordination challenge is present when individual teams are reasonably high-performing, but there are miscommunications and inefficiencies between Scrum teams (e.g., missed dependencies, tremendous overhead, integration issues, environment issues, etc.)
- **Bandwidth**
  All teams are resource constrained in some form, but a team with only this challenge is otherwise in great shape

Limitations

As outlined by Kelly Waters 17 March 2007 *"All about Agile" [42]*

- **Requirements emerge and evolve throughout development**
  This creates the very meaning of agile – flexibility. Flexibility to change course as needed and to ensure delivery of the right product. There are two big flip sides to this principle though. One is the potential for scope creep, which can create the risk of ever-lasting projects. The other is that there is much less predictability, at the start of the project and during, about what the project is actually going to deliver. This can make it harder to define a business case for the project, and harder to negotiate fixed price projects. Without the maturity of a strong and clear vision, and the discipline of fixing timescales and trading scope, this is potentially very dangerous.
- **Agile requirements are barely sufficient**
  Requirements are clarified just in time for development and can be documented in much less detail due to the timeliness of conversations. However, this can mean less information available to new starters in the team about features and how they should work. It can also create potential misunderstandings if the teamwork and communication aren't at their best, and difficulties for team members (especially testers) that are used to everything being defined up front. The belief in agile is that it's quicker to refactor the product along the way than to try to define everything completely up front, which arguably is impossible.
- **Testing is integrated throughout the lifecycle**
  This helps to ensure quality throughout the project without the need for a lengthy and unpredictable test phase at the end of the project. However, it does imply that testers are needed throughout the project and this effectively increases the cost of resources on the project. This does have the effect of reducing some very significant risks, which have proven through research to cause many projects to fail. The cost of a long and unpredictable test phase can cause huge unexpected costs when a project over-runs. However, there is an additional cost to the project to adopt continuous testing throughout.

- **Frequent Delivery of Product and the need for sign-off**
  The users or product owner needs to be ready and available for prompt testing of the features as they are delivered and throughout the entire duration of the project. This can be quite time-consuming but helps drastically to ensure a quality product that meets user expectations.
- **Sustainability on Developers**
  Common feedback is that agile development is rather intense for developers. The need to really *complete* each feature 100% within each iteration, and the relentlessness of iterations, can be mentally quite tiring so it's important to find a sustainable pace for the team.

In summary the purpose of Chapter Two is to provide a definition and investigation of ERP Programs. It illustrated the challenges of implementing an ERP Program and the reasons for ERP failures as well as the best practices to avoid such failures.

Chapter Two looked specifically at the Project Management of ERP Programs and looks at the associated Project Management knowledge areas. This Chapter revealed a detailed analysis of the Software Development Methodologies and provided a comparison across Waterfall and the many evolving Agile Methodologies. Chapter Two described the role of Project Management in ERP implementations and emphasizes its impact across the Project Management knowledge areas. It concludes by revealing how it greatly improves the odds of an ERP implementations success. Chapter Two describes the most common reasons for ERP implementation failures and illustrates the best practices for avoiding such failures.

Chapter Two goes into great depth on the Software Development Methodologies, it compares Waterfall and the various Agile Methodologies and asks which is best suited to an ERP Program. This Chapter concludes with a detailed breakdown of the benefits, risks, challenges and limitations or adopting Agile-At-Scale for an ERP Program.

Chapter 3. Research Problem

3.0 Introduction

The purpose of this Chapter is to present the Participant Observer Research in the form of a comprehensive survey across Dell's IT function. The survey reveals Dell's Agile experiences, the areas where Agile is working well and the areas that need improvement.

This Thesis aims to investigate three areas.

- The Project Management challenges of adopting Agile Methodologies for a multiphase ERP Program.
- The Project Management benefits and risks of adopting Agile Methodologies for a multiphase ERP Program.
- The Project Management limitations and lessons learned of adopting Agile Methodologies for a multiphase ERP Program.

One of the challenges encountered by ERP Projects trying to move to an Agile methodology is how does an Agile team deal with interlocks, especially when those interlocks are using the Waterfall methodology and want their requirements and commitments a year plus in advance? What will make the Agile team seamlessly interlock with its Waterfall counterparts? My experience of working on an Agile ERP Program that had 100+ interlocks, almost all of them following the Waterfall methodology, is that you cannot be 100% agile. Our Agile Product Owners (POs) focused on the current sprint and perhaps a few sprints out. However, we were sent much farther out in terms of requirement development to engage with the business and the interlock teams. We had a situation where we were documenting requirements a year and a half in advance of when they would likely be delivered! "That's not Agile" you may say. And you're right. It's a blend of methodologies. This Hybrid Agile approach resulted in many risks and challenges but also provided many benefits. As part of this Thesis, I intend outlining these.

We have now been challenged to adopt an Agile-At-Scale approach for our latest Phase of this multiphase ERP Program. This adds a whole new dimension as Agile-At-Scale requires interlock teams coming together to produce MBI's (Minimum Business Increments), the minimum amount of incremental business value that can be built, deployed, and consumed by our business partners. From a Project Management perspective, this adds a whole new layer of challenges and risks as to how the Agile-At-Scale teams can be managed and how the end-to-end scope delivery will be managed across numerous interlock teams. This Thesis will outline how to address these challenges, risks and what benefits can be gained from Agile-At-Scale in an ERP Program.

3.1 Research Methodology

3.1.1 Qualitative research involves the use of qualitative data, such as interviews, documents, and participant observation, to understand and explain social phenomena. Qualitative researchers can be found in many disciplines and fields, using a variety of approaches, methods and techniques.
In Information Systems, there has been a general shift in IS research away from technological to managerial and organizational issues, hence an increasing interest in the application of qualitative research methods
Research methods can be classified in various ways, however one of the most common distinctions is between qualitative and quantitative research methods.

Quantitative research methods were originally developed in the natural sciences to study natural phenomena. Examples of quantitative methods now well accepted in the social sciences include survey methods, laboratory experiments, formal methods (e.g. econometrics) and numerical methods such as mathematical modelling.

Qualitative research methods were developed in the social sciences to enable researchers to study social and cultural phenomena. Examples of qualitative methods are action research, case study research and ethnography. Qualitative data sources include observation and participant observation (fieldwork), interviews and questionnaires, documents and texts, and the researcher's impressions and reactions.

Qualitative research methods are designed to help researchers understand people and the social and cultural contexts within which they live. Kaplan and Maxwell (1994) [1] argue that the goal of understanding a phenomenon from the point of view of the participants and its particular social and institutional context is largely lost when textual data are quantified.

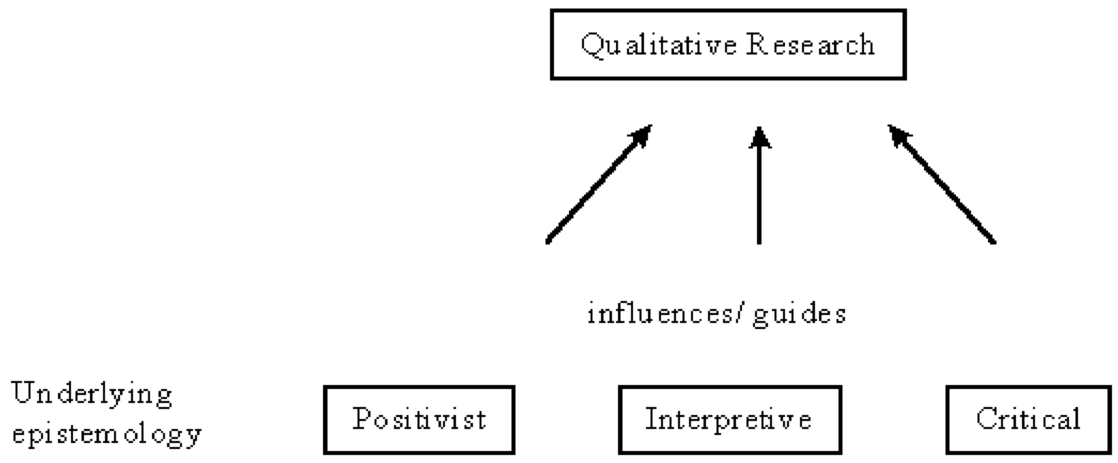


Fig 3.1 Underlying Philosophical assumptions

Qualitative research can be positivist, interpretive, or critical (see Figure 3.1). It follows from this that the choice of a specific qualitative research method (such as the case study method) is independent of the underlying philosophical position adopted. For example, case study research can be **positivist** (Yin, 1994) [2], **interpretive** Walsham, (1993) [8], or **critical**, just as action research can be positivist Clark, (1972) [3], interpretive Elden and Chisholm, (1993) [4] or critical Carr and Kemmis, (1986) [5].. These three philosophical perspectives are discussed below.

Positivist Research

Positivists generally assume that reality is objectively given and can be described by measurable properties which are independent of the observer (researcher) and his or her instruments. Positivist studies generally attempt to test theory, in an attempt to increase the predictive understanding of phenomena. In line with this Orlikowski and Baroudi (1991, p.5)[6] classified IS research as positivist if there was evidence of formal propositions, quantifiable measures of variables, hypothesis testing, and the drawing of inferences about a phenomenon from the sample to a stated population.

Interpretive Research

Interpretive researchers start out with the assumption that access to reality (given or socially constructed) is only through social constructions such as language, consciousness and shared meanings. The philosophical base of interpretive research is hermeneutics and phenomenology Boland, (1985) [7]. Interpretive studies generally attempt to understand phenomena through the meanings that people assign to them and interpretive methods of research in IS are "aimed at producing an understanding of the context of the information system, and the process whereby the information system influences and is influenced by the context" Walsham (1993), p. 4-5).[8] Interpretive research does not predefine dependent and independent variables, but focuses on the full complexity of human sense making as the situation emerges Kaplan and Maxwell, (1994) [1].

Critical Research

Critical researchers assume that social reality is historically constituted and that it is produced and reproduced by people. Although people can consciously act to change their social and economic circumstances, critical researchers recognize that their ability to do so is constrained by various forms of social, cultural and political domination. The main task of critical research is seen as being one of social critique, whereby the restrictive and alienating conditions of the status quo are brought to light. Critical research focuses on the oppositions, conflicts and contradictions in contemporary society, and seeks to be emancipatory i.e. it should help to eliminate the causes of alienation and domination.

3.1.2 Research Method is a strategy of inquiry which moves from the underlying philosophical assumptions to research design and data collection. The choice of research method influences the way in which the researcher collects data. Specific research methods also imply different skills, assumptions and research practices. For the purpose of this study, I adopted the following research methods that will be discussed here,

- Participant Observer
- Survey
- Case Study Research
- Interviews

3.1.2.1 Participant Observer Participant Observation is where the researcher immerses himself / herself in the subject being studied. It is argued that this technique allows the researcher to gain a deeper insight into the subject than he / she would otherwise be allowed with other data collection methods Blum, (1952) [9]. This method is particularly relevant to research topics which involve interpersonal group process.

According to Michael Quinn Patton (1986) [10] there are three possible roles for a participant observer

- Full participant observation
- Partial participant observation
- Onlooker: observation as outsider.

Also, according to Quinn Patton (1986) [10] there are three ways to portray the role of participant observer to other members of the study

- Overt Observations – subjects know that Observations are being made and who the observer is.
- Observer role is known to some but not to others.
- Covert observation – subjects do not know the Observations are being made or that there is an observer.

For the purposes of this study, the researcher portrayed the role of participant observer through overt observation.

Participant observation involves the researcher's involvement in a variety of activities over an extended period. In the researcher's case, over twenty years working as an IT professional, specializing in ERP Implementations. This experience enabled the researcher to observe the evolution of IT methodologies and to participate in a wide spectrum of IT roles which facilitates a better understanding of those methodologies and their associated benefits, risks, limitations, challenges.

This qualitative research method is a widely used methodology in many disciplines. Its aim is to gain a close and intimate familiarity with a given group of individuals (such as a religious, occupational, sub cultural group, or a particular community) and their practices through an intensive involvement with people in their cultural environment, usually over an extended period of time. The method originated in the field research of social anthropologists, especially Bronisław Malinowski [11] and his students in Britain, the students of Franz Boas in the United States, and in the later urban research of the Chicago School of Sociology.

Participant observation requires the researcher to be a subjective participant in the sense that they use knowledge gained through personal involvement with the research subjects to interact with and gain further access to the group. This component supplies a dimension of information that is lacking in survey data.

Participant observation research also requires the researcher to aim to be an objective observer and record everything that he or she has seen, not letting feelings and emotions influence their observations and findings.

The strengths of participant observation include the depth of knowledge that it allows the researcher to obtain, and the perspective of knowledge of social problems and phenomena generated from the level of the everyday lives of those experiencing them. Many consider this an egalitarian research method because it centres the experiences, perspectives, and knowledge of those studied.

3.1.2.2 Survey

As a member of Dell Technologies IT Project Management Office (PMO), I have been part of a pilot team set up to promote the adoption of Agile. Six months into the pilot, we conducted an Agile Survey in August 2016 to discover what is working well and what areas need more focus.

The Survey was sent to ~4000 legacy Dell IT team members, was open for 2 weeks and was anonymous.

There were 369 responses. Outlined below are the response distribution by role and by location.

Full details of the survey's findings are presented and analyzed in section 3.2 – Participant Observer Research.

3.1.2.3 Case Study Research

Data collection and analysis techniques of a quantitative (concerned with words and meaning) and qualitative (concerning numbers and measures) can be used in case study research Yin, (1994) [2]. Case studies typically make use of both quantitative and qualitative data collection and analysis methods Darke et al., (1998) [13]. As quantitative methods are not as well established as their qualitative counterparts, the volume and variety of data collected may result in data analysis being difficult and time consuming. This section will discuss data collection and data analysis in more detail.

Data Collection

There are three main principles of data collection Yin, (1994) [2]:

1. **Use multiple sources of evidence**

   This can help to reduce any researcher biases in any collection and analysis of data Miles & Huberman, (1984) [12]. If there are multiple sources of information, providing multiple measures of the same phenomenon, the case study findings are strengthened considerably Darke et al., (1998) [13]. If a researcher can triangulate converging findings from different sources the construct validity is increased considerably Eisenhardt, (19890 [14]; Yin, (19940 [2]; Maimbo & Pervan, (2005) [15].

2. **Create a case study database** Darke et al., (1998) [13]

   This should be a database separate from the final reports to be written and should contain Yin, (1994) [2]:

   - Case study notes
   - Case study documents
   - Tabular materials
   - Narratives

   It is important to use a case study database to allow cross referencing and citation of relevant evidence Darke et al., (1998) [13]. This is especially beneficial when pursuing qualitative data collection methods as the case study database may provide for some statistical information based on the qualitative data collected.

3. **Maintain a chain of evidence** Darke et al., (1998) [13]

   The case study protocol should be used to maintain the link between the initial case study questions and the case study procedure Yin, (1994) [2]. Maintaining a chain of evidence allows for checking against the database at a later date.

Data collection should be facilitated by allowing the participants to familiarize themselves with the research instrument in advance Maimbo & Pervan, (2005) [15]. This should result in the participants being more comfortable with the process which arguably provides higher quality data. Analysis of data can be made easier if data collection is overlapped as it gives the researcher more flexibility Miles & Huberman, (19840 [12]; Eisenhardt, (1989) [14]; Yin, (1994) [2].

Case studies typically have a number of different data collection methods Eisenhardt, (1989) [14].

- Participant Observations (assuming a role in the situation & getting an inside view of the events) (Eisenhardt, (19890 [14]; Marshall and Rossman,(1999) [17];Yin,(2003) [2]
- Documentation (letters, agendas, progress reports) Eisenhardt, (1989);[14] Marshall and Rossman,(19990 [17]; Yin, (2003) [2]; Maimbo and Pervan,(2005) [15], Archiva (Records (Service records, organizational charts, budgets etc.) Yin, (2003) [2]
- Interviews (typically open-ended, but also focused, structured & surveys are possible) Eisenhardt, (1989) 14]; Marshall and Rossman, (1999) [17]; Yin, (2003) [2]; Maimbo and Pervan, (2005) [15]
- Questionnaires Eisenhardt, (1989) [14]; Maimbo and Pervan, (2005) [15]
- Direct Observations (formal or casual; useful to have multiple observers) Eisenhardt, (1989) [14]; Marshall and Rossman, (1999) [17]; Yin, (2003) [2]
- Physical Artefacts Yin, (2003) [2]

It is argued that the increased number and variety of data collection techniques available to the researcher is a direct consequence of the increased number and complexity of research methodologies Faulkner, (1982) [18].

3.1.2.4 Interviews

Within this research study, the interviews conducted were either face to face or over the phone. Some of the Dell Technologies interviewees were in the United States so these interviews were conducted face to face when the researcher was on site and over the phone at all other times.

Interviews are essential sources of information for case study research Yin, (1994) [2] and because of their interactive nature, can be a key data collection method for the interpretivist researcher Walsham, (1995) [8]. Interviewing is one of the most popular data collection techniques among qualitative researchers Bodgan and Biklen, (1982) [16]; Marshall and Rossman, (1989) [17]. One possible reason for this is that interviewing maximises the possibility of encountering unexpected data Becker and Geer, (1982) [19] while at the same time capturing the contextual complexity Benbasat et al., (1987) [20]. Interviewing is described as a fundamental technique associated with qualitative research Marshall and Rossman, (1989) [17]

3.2 Participant Observer Research

- In August 2016, Dell IT conducted an Agile Survey to discover what is working well and what areas need more focus.
- The Survey was sent to ~4000 legacy Dell IT team members, was open for 2 weeks, and was anonymous.
- There were 369 responses. Outlined below are the response distribution by role and by location.
- Through the survey, 42 new SDLC "requests for help" engagements are being addressed.

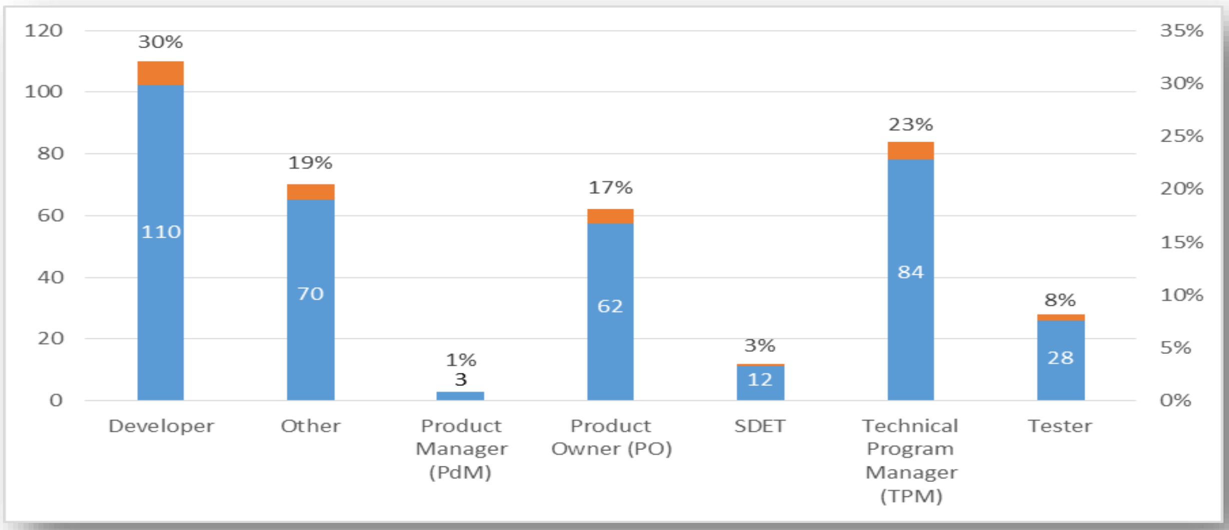


Fig 3.2 Respondents Roles

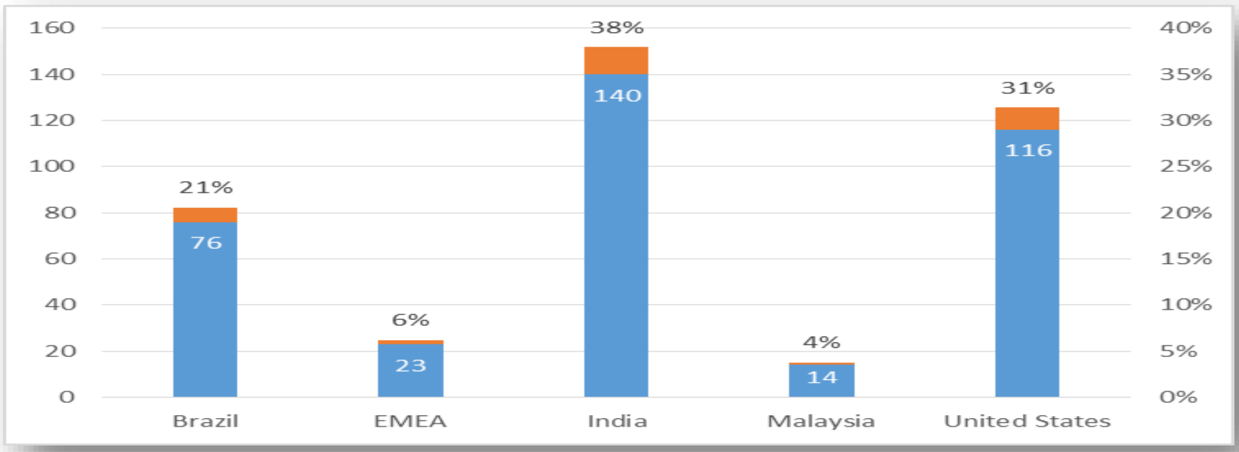


Fig 3.3 Respondents Location

Key Takeaways

The investment in training and coaching is paying off, this is evident in the Sprint statistics with a higher % of User Stories "Ready for Dev" at the start of the Sprint.

Resources (i.e., the SDLC site, Agile Mentors, Chatter, etc.) are readily available when questions arise about Agile within legacy Dell IT.

There is still work to be done to ensure team members and business partners see the benefits of our Agile transformation.

The majority said they have received sufficient training to perform their roles, but some expressed the need for more training on specific topics

Highlights:

Team members understand how to perform their roles (79%)

People know where to go to find information on the Agile methodologies (77%)

Opportunities:

Some team members don't feel that the Agile Transformation has improved how they work (38%)

The full benefit of Agile is not being realized by our business partners (34%) or by IT (32%)

Additional training may be needed (31%)

General Agile Questions	Agree or Strongly Agree	Disagree or Strongly Disagree
1 - I think we (Dell IT) have benefitted by going Agile.	68%	32% ▼
2 - I think our business partners have benefitted by going Agile.	66%	34% ▼
3 - The Agile Transformation has improved the way I do my work.	62%	38% ▼
4 - I know how to perform my role using Agile.	79% ▲	21%
5 - I have received sufficient training to enable me to do my job using the Agile methodologies.	69%	31% ▼
6 - I know where to obtain information about the Agile methodologies.	77% ▲	23%

Table 3.4. Response to qn. 1 – 6

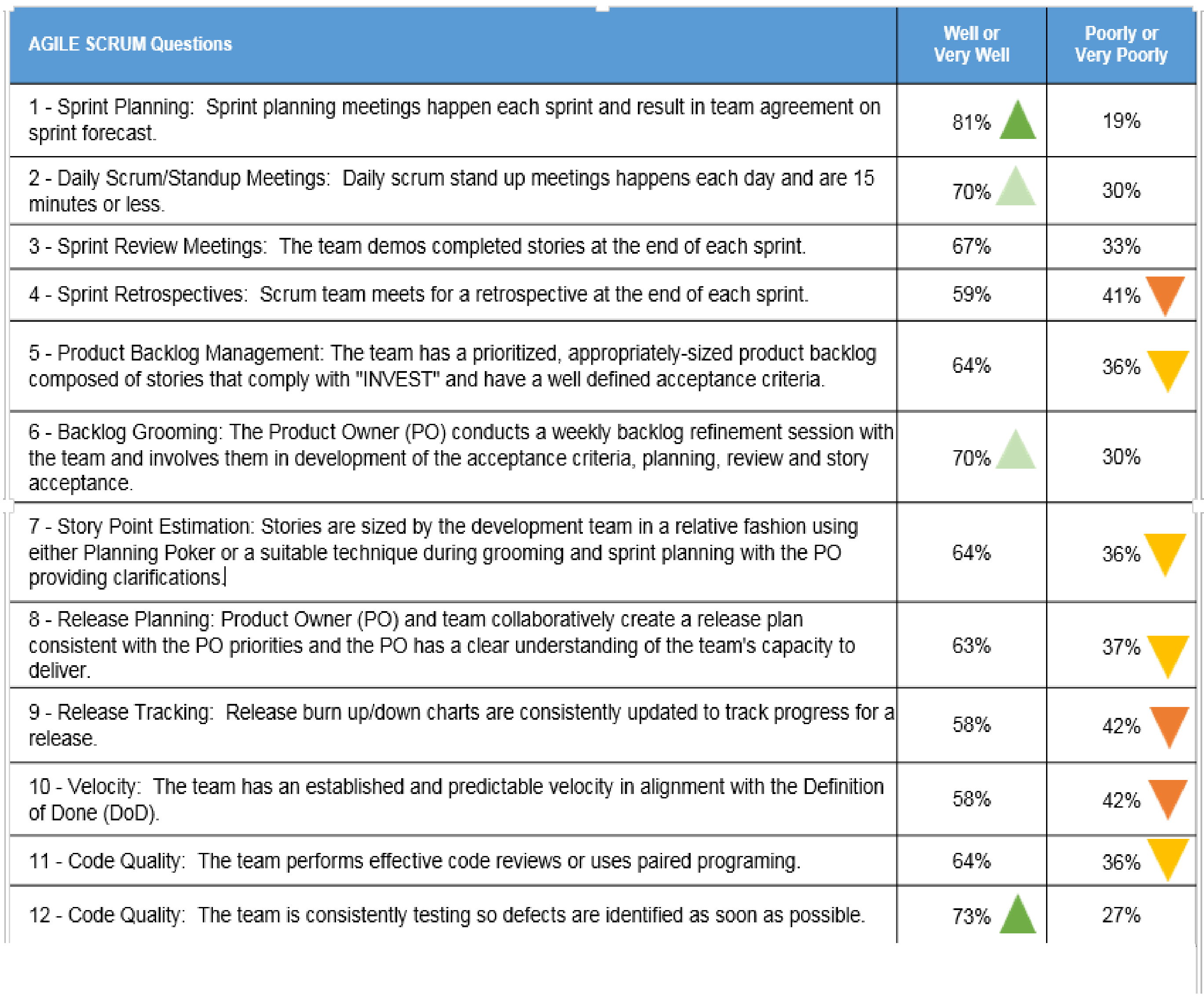
From those using Agile Scrum

Highlights:

- Sprint Planning (81%) and Testing (77%) are being performed well or very well.

Opportunities:

- Some teams are struggling to establish velocity (42%)
- Tracking burn ups/down for a release is inconsistent (42%)
- Teams aren't consistently conducting Sprint Retrospectives (41%)



AGILE SCRUM Questions	Well or Very Well	Poorly or Very Poorly
1 - Sprint Planning: Sprint planning meetings happen each sprint and result in team agreement on sprint forecast.	81% ▲	19%
2 - Daily Scrum/Standup Meetings: Daily scrum stand up meetings happens each day and are 15 minutes or less.	70% ▲	30%
3 - Sprint Review Meetings: The team demos completed stories at the end of each sprint.	67%	33%
4 - Sprint Retrospectives: Scrum team meets for a retrospective at the end of each sprint.	59%	41% ▼
5 - Product Backlog Management: The team has a prioritized, appropriately-sized product backlog composed of stories that comply with "INVEST" and have a well defined acceptance criteria.	64%	36% ▼
6 - Backlog Grooming: The Product Owner (PO) conducts a weekly backlog refinement session with the team and involves them in development of the acceptance criteria, planning, review and story acceptance.	70% ▲	30%
7 - Story Point Estimation: Stories are sized by the development team in a relative fashion using either Planning Poker or a suitable technique during grooming and sprint planning with the PO providing clarifications.	64%	36% ▼
8 - Release Planning: Product Owner (PO) and team collaboratively create a release plan consistent with the PO priorities and the PO has a clear understanding of the team's capacity to deliver.	63%	37% ▼
9 - Release Tracking: Release burn up/down charts are consistently updated to track progress for a release.	58%	42% ▼
10 - Velocity: The team has an established and predictable velocity in alignment with the Definition of Done (DoD).	58%	42% ▼
11 - Code Quality: The team performs effective code reviews or uses paired programing.	64%	36% ▼
12 - Code Quality: The team is consistently testing so defects are identified as soon as possible.	73% ▲	27%

Table 3.5 Response to Agile scrum questions

To gage the adoption of the Agile At Scale concepts:

35% of the respondents are using the Agile at Scale concepts.

Those using Agile at Scale report the following benefits:

- Better traceability
- Flexibility to respond to changing business needs and priorities
- Using consistent terminology: DoD, MBI
- Well managed alignment with interlocks
- Hard dates are respected and delivered when possible
- Able to deliver key requirements to the Business

Area for improvement as identified from 215 distinct comments.

- Culture
  - Command and Control-style management still exists
  - Too many meetings
- Methodology
  - Not all projects are a good fit for Agile
  - We are not agile. We are “water-scrum-fall”
- Business Agile alignment
  - Our business partners still think and work in a waterfall manner
  - Lack of business partner engagement
- Roadmap Planning & Funding Processes
  - These processes are still project-based and hinder Agile delivery
- Training & Coaching
  - We are not mature enough to not have Agile coaches; Mentors are often too busy to help
  - More training is needed: Product Owners, General Scrum practices
- Release Management
  - Release management processes are not aligned with Agile delivery
- Environment Management
  - Environment availability is hindering Agile delivery
- Roles & Responsibilities
  - Role of the BSA, Development Lead, Architect in Agile are not well defined
  - Combining the TPM with Scrum Master is not working

Chapter Three presents the Participant Observer Research in the form of a comprehensive survey across Dell’s IT function. The survey reveals Dell’s Agile experiences, the areas where Agile is working well and the areas that need improvement. It presents the detailed findings of an extensive Dell IT survey, specifically on Dell’s Agile adoption experiences. It describes the key highlights, opportunities and takeaways.

Chapter 3 concludes that while adopting Agile allows for better traceability and the ability to deliver key requirement to the business partners, it requires significant investment in training and business alignment and introduces challenges across Release Management, Environment Management and Roadmap planning.

Chapter 4. Findings

4.0 Introduction

The purpose of this Chapter is to describe and interprets the findings of the Dell Survey and looks specifically at Dell's experience in a large multiphase ERP Program. It focuses on the research objectives of the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program. Chapter four will illustrate the research objectives by providing evidence based on Dell's direct experiences and will set the foundation for Chapter Five which will conclude on how best Agile-At-Scale can be adopted in an ERP Program.

Chapter 4 breaks down the challenges, benefits, risks, limitations and lessons learned of Adopting Agile for an ERP Program. It also provides concrete examples of Dell's experience across these categories and ties back the findings to the company wide findings outlined at the end of Chapter 4.

Chapter 4 presents an in-depth analysis of Dell Technologies Agile Adoption and Dell's Agile-At-Scale IT Delivery Framework. The research illustrates the details behind the Scrum Workflow and describes the Project Teams associated roles and responsibilities, the artefacts and the Scrum/Sprint Planning procedures. It reveals the concept of the MBI (Minimum Business Incremental Business Value) and how this is used to form and sequence the Product backlog

This Chapter aims to investigate three areas

- The Project Management **challenges** of adopting Agile Methodologies for a multiphase ERP Program.
- The Project Management **benefits and risks** of adopting Agile Methodologies for a multiphase ERP Program.
- The Project Management **limitations and lessons learned** of adopting Agile Methodologies for a multiphase ERP Program.

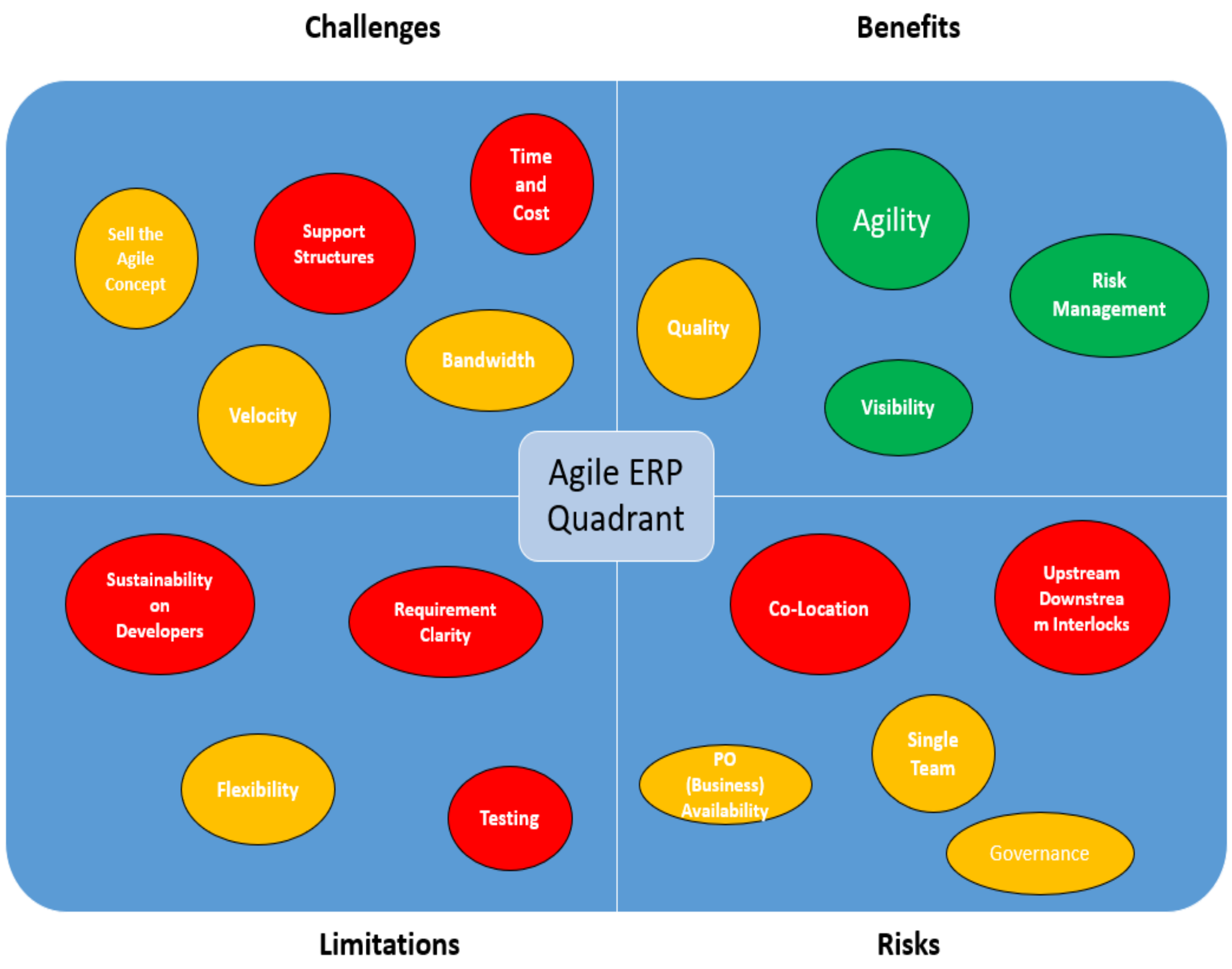


Fig 4.1 Agile ERP Quadrant – Red denotes a negative impact, green positive and amber neutral.

Based on the findings from Chapters Three and Four, and my experience as a participant observer across multiple ERP global implementations,

I have compiled the Agile ERP Quadrant.

The Agile ERP Quadrant illustrates the four key areas undertaken for this research exercise, the challenges, benefits, risks and limitations of adopting Agile in a large multiphase ERP Program.

Red denotes a negative impact on the Program, green a positive impact and amber neutral.

The size of the bubble signifies the magnitude of the impact, the large the bubble, the higher the impact.

Sections 4.1, 4.2, 4.3 will go into the details behind each of these categories.

4.1 The Project Management challenges of adopting Agile Methodologies for a multiphase ERP Program.

In this section I will identify the key challenges across the following categories

- Time and Cost
- Support Structures
- Selling the concept of Agile
- Velocity
- Reliability and Consistency
- Quality
- Coordination
- Bandwidth

Time and Cost

Based on Dell's experience, moving from a Waterfall Methodology to being Agile and maturing to Agile-At-Scale requires a very significant investment in training and coaching. Supporting the entire IT organization to facilitate them in becoming trained and certified in Scrum techniques is a major undertaking. Dell have invested heavily in this endeavor and this is evident in the extremely high percentage of people who responded very positively when asked

"I know how to perform my role using Agile" (79%)

"I have received sufficient training to enable me to do my job using Agile Methodologies" (69%)

"I know where to obtain information about Agile Methodologies" (77%)

When you consider the typical cost of becoming a Certified Agile Scrum Master (CSM) is over $1000 and given the fact that Dell IT has 6000+ employees, it is a major financial investment rolling Agile training and certification out to the entire Dell IT function. In 2016, Dell merged with EMC, as part of that deal, Dell took ownership of Pivotal. Dell are now beginning to adopt the Pivotal Cloud Foundry (PCF) approach to digital transformation, pair programming is fundamental to the PCF Agile Development process. Pivotal certification can take up to six weeks to complete so rolling it out to an entire IT organisation, on the scale of Dell, is not financially viable.

Support Structures

In order for the transition from Waterfall to Agile to be effective, as seen in Dell, the organization needs to invest in support structures. Dedicated expert resources need to be made available as Mentors and their role is to provide continuous development and training across all teams. Detailed educational material needs to be maintained on the SDLC sites and regular updates are required across the social media platforms, such as Chatter, etc.

Dell have initiated the "Dell Digital Way" to support Dell's Transformation journey from Waterfall to Agile and ultimately to using Pivotal technology Agile PCF, Pivotal Methodology (Pair Programming). This is a dedicated support and training team which focuses on three core components across Dell IT, people, process and technology. When we think about people, it is about balanced teams (autonomous, collaborative groups with a variety of cross-disciplinary IT skills). Process is shifting from Project delivery to being much more Product focused. Enabling fast delivery and continuous value. Technology, modernizing Dell's full stack, harnessing PCF or Cloud Foundry. Pivotal will become even more pervasive in Dell's future.

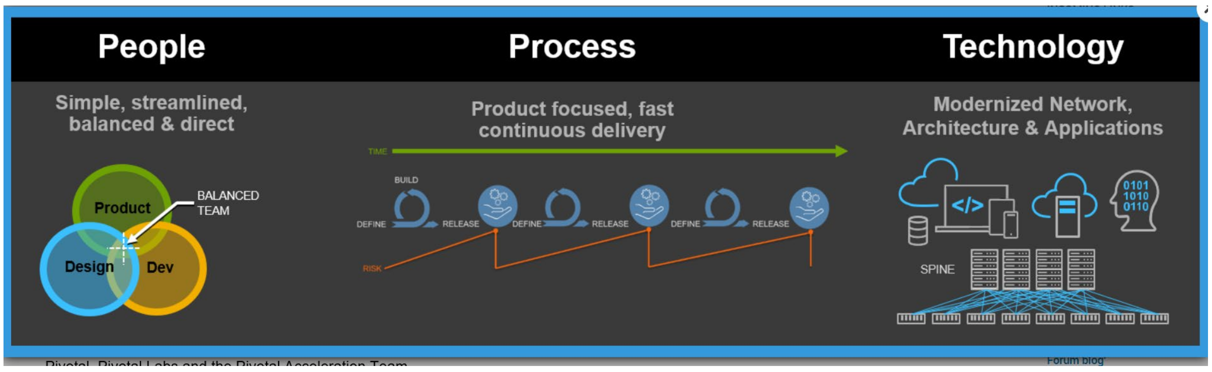
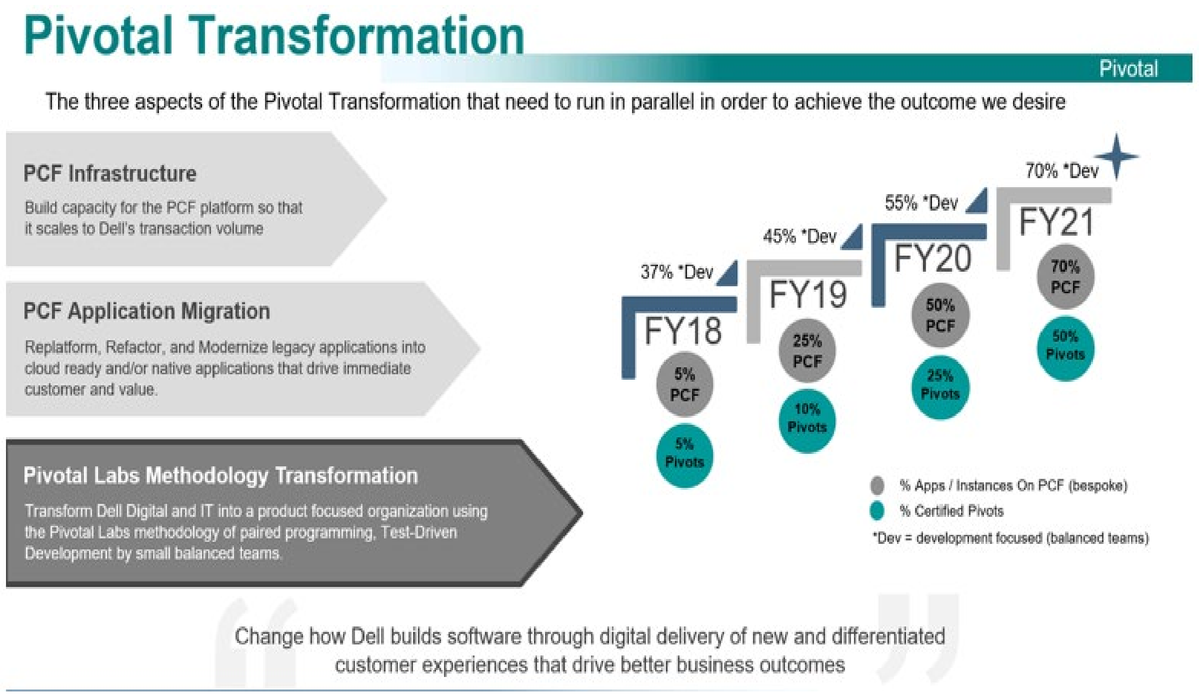

People
Simple, streamlined, balanced & direct
Product
BALANCED TEAM
Design
Dev
Process
Product focused, fast continuous delivery
TIME
BUILD
DEFINE
RELEASE
RISK
Technology
Modernized Network, Architecture & Applications
</>
0101 1010 0110
SPINE

Fig 4.2 Dell Digital Way

Selling the Concept of Agile

There is a lot of work and communication required to ensure IT team members and business partners see the benefits of the Agile PCF (Pivotal) transformation. As Dell is very much at the early stages of its Agile PCF (Pivotal) journey, constant communication and clear goals are required to reinforce the push for adoption, success stories, as well as areas for improvement.

As illustrated below, Dell's Agile PCF (Pivotal) transformation is on a multi-year journey which has seen it go from 5% of its Development teams adopting the Pivotal Labs methodology of Agile paired programming to aiming to have 70% by 2021.



Pivotal Transformation

The three aspects of the Pivotal Transformation that need to run in parallel in order to achieve the outcome we desire

PCF Infrastructure

Build capacity for the PCF platform so that it scales to Dell's transaction volume

PCF Application Migration

Replatform, Refactor, and Modernize legacy applications into cloud ready and/or native applications that drive immediate customer and value.

Pivotal Labs Methodology Transformation

Transform Dell Digital and IT into a product focused organization using the Pivotal Labs methodology of paired programming, Test-Driven Development by small balanced teams.

Change how Dell builds software through digital delivery of new and differentiated customer experiences that drive better business outcomes

Fig 4.3 Pivotal Transformation

Velocity

Whether real or only perceived, the velocity challenge indicates that "the business" does not believe that they are getting adequate throughput / value from the technology team.

This is certainly a challenging concept at Dell and this can be seen by the very low proportion of responds who agreed that

"The team has an established and predictable velocity in alignment with the definition of done (DoD)" (58%)

Dell are addressing the perception gap between what the business expect V's what IT deliver by including the business partners at critical points during the process.

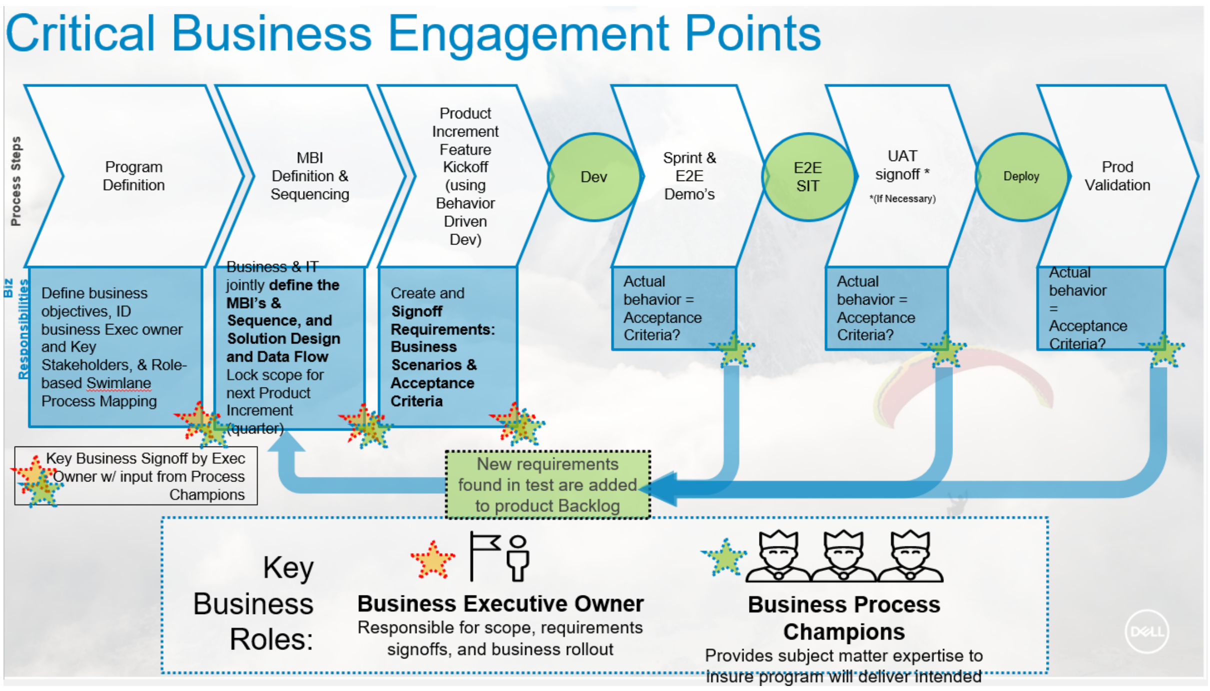

Critical Business Engagement Points
Process Steps
Program Definition
MBI Definition & Sequencing
Product Increment Feature Kickoff (using Behavior Driven Dev)
Dev
Sprint & E2E Demo's
E2E SIT
UAT signoff * *(If Necessary)
Deploy
Prod Validation
Biz Responsibilities
Define business objectives, ID business Exec owner and Key Stakeholders, & Role-based Swimlane Process Mapping
Business & IT jointly define the MBI's & Sequence, and Solution Design and Data Flow Lock scope for next Product Increment (quarter)
Create and Signoff Requirements: Business Scenarios & Acceptance Criteria
Actual behavior = Acceptance Criteria?
Actual behavior = Acceptance Criteria?
Actual behavior = Acceptance Criteria?
Key Business Signoff by Exec Owner w/ input from Process Champions
New requirements found in test are added to product Backlog
Key Business Roles:
Business Executive Owner
Responsible for scope, requirements signoffs, and business rollout
Business Process Champions
Provides subject matter expertise to insure program will deliver intended

Fig 4.4 Critical Business Engagement Points

<u>Reliability & Consistency</u>

This challenge is typically shown through Agile teams that consistently over-commit and miss their Sprint goals and deliverables. This is particularly challenging in a multiphase ERP Program where different Business Units are at different phases in the Development and test cycles. Spikes in sprints result in missed deadlines and that has a knock on impact on the next business unit's deployment dates and release windows.

This was very evident during Dell's Transformation ERP Program for North America and Canada.

The Front Office Order Entry applications did not have adequate resources to support focusing on two major regions concurrently, the Back Office Order Management, Fulfilment and Finance applications were staffed to support both regions but were not able to progress without the upstream interlocks.

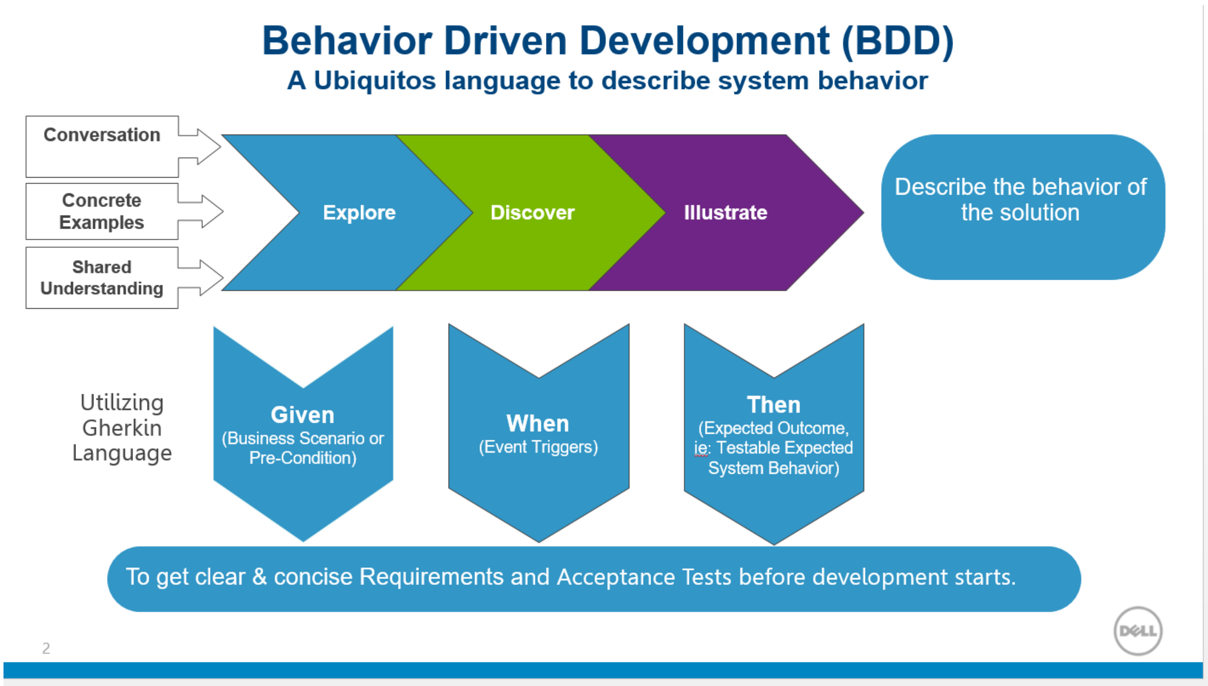
<u>Quality</u>

Teams that have a quality challenge "finish" their sprints and deliverables but accumulate a significant amount of technical debt in the form of defects and poor design choices. Under the Waterfall Methodology this lack of quality is evident as at testing gate exit review, with Agile this can be right up to Production launch.

Dell is addressing this lack of quality by adopting Behaviour Driven Development (BDD). BDD is a common language and framework for capturing requirements and successfully managing and changing system and process behaviour.

BDD is an extension of TDD (Test Driven Development) where:

- Tests are written in plain descriptive english – 'Gherkin' Given – When - Then
- Tests are explained as behavior of application and are user focused
- BDD frameworks such as Cucumber (java), SpecFlow (.net) or Jbehave (java) can convert BDD into functional test code. It acts as a "bridge" between Business & Technical Language



Behavior Driven Development (BDD)
A Ubiquitos language to describe system behavior
Conversation
Concrete Examples
Shared Understanding
Explore
Discover
Illustrate
Describe the behavior of the solution
Utilizing Gherkin Language
Given (Business Scenario or Pre-Condition)
When (Event Triggers)
Then (Expected Outcome, ie: Testable Expected System Behavior)
To get clear & concise Requirements and Acceptance Tests before development starts.
2

Fig 4.5 Behavior Driven development

Coordination

A coordination challenge is present when individual teams are reasonably high-performing, but there are miscommunications and inefficiencies between Scrum teams (e.g., missed dependencies, tremendous overhead, integration issues, environment issues, etc.)

This has been very apparent in Dell's multiphase ERP Program, there are 300+ Developers across 35+ Feature Teams, representing 80+ Interlock dependencies. The coordination required has been one of the key challenges we have encountered.

Bandwidth

When dealing with a Multiphase ERP Program, “fire fights” in the upcoming launch tends to suck in all available resources and this has a direct impact on the next business unit and its velocity and progress.

Key Takeaways 4.1 - The Project Management challenges of adopting Agile Methodologies for a multiphase ERP Program clearly require very significant investment in training, a well-resourced Support and Communications Project Management Office and clearly defined achievable goals. This requires a transformational culture change across both the IT and Business Teams.

4.2 The Project Management benefits and risks of adopting Agile Methodologies for a multiphase ERP Program.

In this section I will identify the key benefits and risks across the following categories

- Quality
- Visibility
- Risk management
- Flexibility/Agility
- Lack of available business collaborators
- Incompatibility with existing governance policies
- Individual performance management systems
- Out of phase with upstream and downstream work streams
- Single team assignment
- Co-location

4.2.1 Benefits

Quality

A key principle of Agile development is that testing is integrated throughout the lifecycle, enabling regular inspection of the working product as it develops. This allows the product owner to make adjustments if necessary and gives the product team early sight of any quality issues.

From my experience of working as IT Program Manager on large scale ERP Programs for 18 years, this principle failed when interlock dependencies became apparent. Feature Teams can address code quality issues, but it is not until end to end testing has been completed that the true quality can be assessed. In my experience, Agile leads to a significant increase in defects being detected late in the test cycle, during end to end test.

As discussed in section 4.1, Behaviour Driven Development (BDD) is a positive initiative which can increase collaboration across the IT Project Team, the business users and test teams.

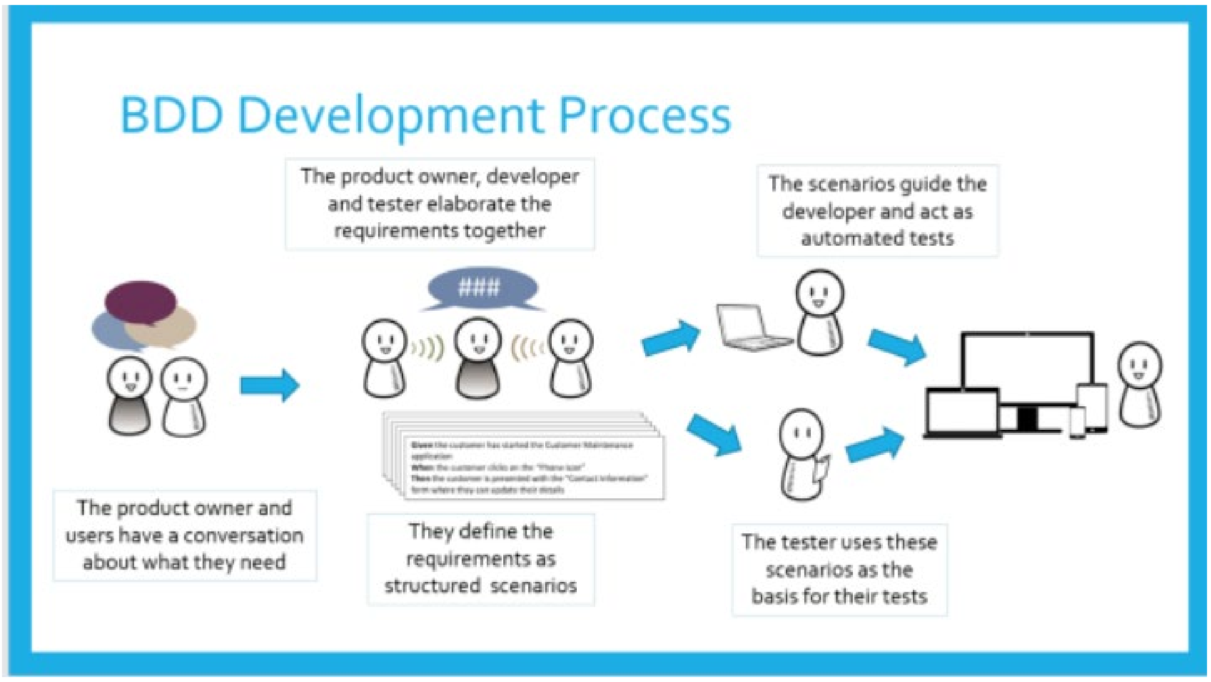

BDD Development Process
The product owner, developer and tester elaborate the requirements together
The scenarios guide the developer and act as automated tests
###
The product owner and users have a conversation about what they need
They define the requirements as structured scenarios
The tester uses these scenarios as the basis for their tests

Fig 4.6 BDD Development Process

<u>Visibility</u>

Agile development principles encourage active 'user' involvement throughout the product's development and a very cooperative collaborative approach. This provides excellent visibility for key stakeholders, both of the project's progress and of the product itself, which in turn helps to ensure that expectations are effectively managed. This was achieved by holding bi-weekly scrum review calls and is certainly a benefit from the Business Partners perspective. See survey results below.

"Sprint Review Meeting: The team demo's completed story points at the end of each sprint" (67%)

Under the Waterfall Methodology, the Business Partners will have very limited exposure to the IT teams during the Development and early Development and Integration test phases. With Agile the by-weekly meetings drive collaboration and an early understanding of what is going to be delivered.

Risk Management

Small incremental releases made visible to the product owner and product team through its development help to identify any issues early and make it easier to respond to change. The clear visibility in Agile development helps to ensure that any necessary decisions can be taken at the earliest possible opportunity, while there's still time to make a material difference to the outcome.
In a fast paced, ever evolving Technology Company like Dell, there is a huge appetite to be able to break deliverables down to a minimum (see Minimum Business Increment in section 3). Where this becomes a challenge is when an ERP Program does not allow for delivery of small increments due to the interdependencies on 100+ interlocking applications and the need to deliver core modules simultaneously.

Flexibility / Agility

In traditional development projects, we write a large spec up-front and then tell business owners how expensive it is to change anything, particularly as the project goes on. In fear of scope creep and a never-ending project, we resist changes and put people through a change control committee to keep them to the essential minimum. Agile development principles are different. In agile development, change is accepted. *In fact, it's expected*. Because the one thing that's certain in life is change. Instead the timescale is fixed and requirements emerge and evolve as the product is developed. Of course, for this to work, it's imperative to have an actively involved stakeholder who understands this concept and makes the necessary trade-off decisions, trading existing scope for new.
The challenge with this benefit in a multiphase ERP Program is that there is still a "Big Bang" deployment required across many of the applications and modules, this greatly limits the trade-off decisions. What I have seen in Dell's experience is that trade-offs result in deferred scope which need to be addressed in future releases, this has led to the Program running 18 months beyond what was envisioned.

4.2.2 Risks

Lack of available business collaborators

Each team needs a single point of contact business representative (a.k.a. sponsor or customer). known as the “product owner” in Scrum. This person needs to frequently engage with the team and have the authority to specify, prioritize, and accept the team’s work.
The PO can often become a bottleneck in a multiphase ERP Program as they are focused on the next release and this leads to delays for the next Business Unit. Raising a “Spike” in the backlog to address this is the most effective approach but in times of severe resource constraints this is not always possible.

Incompatibility with existing governance policies

Quality gate sign-off policies. Documents such as project plans, requirements, and design documents may not be amendable after formal sign-off from upper management (without formal change requests). However, Agile won’t work if teams and their product owners are not empowered to make changes to tasks, estimates, priority, and design.
Agile takes away the rigidity brought about by the strict sign-off policies in a Waterfall world, but the danger with Agile is that you lose controls, governance and audit artifacts.
Without the rigid gate exit reviews associated with Waterfall, you lose the review mechanisms that forced the requirements to be detailed, the architectural documentation to be peer reviewed and the software requirements to be approved before moving into the Development effort. In a Program where the changes are to an established platform and the size of the effort and interlock dependencies are minimal, this is acceptable, that is not the case for large scale ERP Programs.

Individual performance management systems

The Agile value is commitment to the team, where all “win” or “lose” together. This is diametrically opposed to "Individual" performance ranking meritocracy systems currently in place across organisations.

Out of phase with upstream and downstream work streams

The Agile team is somewhere in the middle of a larger flow of work or value stream. It increases frequency of inflow and outflow to every two weeks, maybe even less. If the cadence of work upstream and downstream from the team does not also change, then delays and log jams can occur.
At Dell this has been one of the most challenging elements of adopting Agile in an ERP environment, trying to align 80+ Interlocking Applications and understanding the

dependencies from a development, test and deployment perspective adds major risk to the Program.

Single team assignment

People who are simultaneously committed to more than one "Team" are unable to make meaningful delivery commitments because of inevitable conflicting team imperatives. If members can't make meaningful commitments, Agile won't work. This problem grows in magnitude as the program grows in size and more teams are involved.

Ring fencing resources is difficult to achieve as key resources are regularly pulled into urgent support or "firefighting" scenarios.

Co-location

Collaboration thrives when teams are in intimate proximity. Maybe an exceptional team can overcome this challenge, but not the many teams of a larger Agile transformation. Once teams get established and members have effective working relationships, the team has a greater chance of surviving if some members are dispersed. Also, an Agile program involving many teams does not need to be completely co-located in order to be successful. What is vital is that small groups of people intimately work together and develop the trust and mutual understanding necessary for effective collaboration. It will be extremely difficult for small groups to achieve this state if they are never afforded the opportunity to work in a shared, physical space.

When you are dealing with IT teams of greater than 300 people, as we do in Dell's ERP Programs. Co-location is not entirely possible. Dell invests in having teams co-locate during End to End Test cycles and deployment windows but have all teams Co-located at all times is not feasible.

Pair Programming is an Agile (Pivotal) concept Dell are now piloting, it requires two Developers to share a computer with the aim of helping each other achieve a daily task. This requires co-location and is causing major challenges to Development Managers who may have multiple resources spread across different locations.

Key Takeaways 4.2 - The Project Management benefits and risks of adopting Agile Methodologies for a multiphase ERP Program illustrate the clear benefits gained by the collaboration between the IT Project Team, Business Users and Testers. The Agile approach provides for much needed visibility for the business partners and the small incremental releases lead to a far higher level of risk management. The flexibility provided by Agile is a clear benefit however, this is not always achievable in a large ERP Program due to interdependencies across interlocks.

Agile can create risks caused by resource constraints and loss of controls, governance and audit artifacts. Interlock alignment is a very significant risk along with the Co-location feasibility.

4.3 The Project Management limitations and lessons learned of adopting Agile Methodologies for a multiphase ERP Program.

In this section I will identify the key limitations and lessons learned across the following categories

- Requirements emerge and evolve throughout development
- Agile requirements are barely sufficient
- Testing is integrated throughout the lifecycle
- Frequent Delivery of Product, the need for sign-off and the Sustainability on Developers

4.3.1 Limitations

Requirements emerge and evolve throughout development

This creates the very meaning of Agile – flexibility. Flexibility to change course as needed and to ensure delivery of the right product. There are two big flip sides to this principle though. One is the potential for "scope creep", which can create the risk of ever-lasting projects. The other is that there is much less predictability, at the start of the project and during, about what the project is going to deliver. This can make it harder to define a business case for the project, and harder to negotiate fixed price projects. Without the maturity of a strong and clear vision, and the discipline of fixing timescales and trading scope, this is potentially very dangerous.

In the course of Dell's multiphase ERP Program, the lack of predictability has resulted in a vast increase in Change Requests and tests defects, this has led to deployment delays and scope being moved into the next Release window which in turn is putting future software releases at risk.

Agile requirements are barely sufficient

Requirements are clarified just in time for development and can be documented in much less detail due to the timeliness of conversations. However, this can mean less information available to new starters in the team about features and how they should work. It can also create potential misunderstandings if the teamwork and communication aren't at their best, and difficulties for team members (especially testers) that are used to everything being defined up front. The belief in Agile is that it's quicker to refactor the product along the way than to try to define everything completely up front, which arguably is impossible.

We have had cases during Dell's ERP Program whereby interlocking teams have created their entire solution based on the assumption that an upstream system would adopt a certain approach. Under Agile, it was not apparent that this assumption was incorrect until after the downstream system had completed their Development and testing effort. It only became apparent during full end to end testing and the impact was a change request to completely redesign the solution for a number of applications. In a Waterfall approach, this would have been found in the design phase, long before any Development had been completed.

Testing is integrated throughout the lifecycle

This helps to ensure quality throughout the project without the need for a lengthy and unpredictable test phase at the end of the project. However, it does imply that testers are needed throughout the project and this effectively increases the cost of resources on the project. This does have the effect of reducing some very significant risks. That have been proven through research to cause many projects to fail. The cost of a long and unpredictable test phase can cause huge unexpected costs when a project over-runs. However, there is an additional cost to the project to adopt continuous testing throughout.

During the Dell ERP Program, I have seen a three-fold increase in the cost of testing under the Agile approach, continuous testing and a significant increase in co-location has been the main driver of this increase.

Frequent Delivery of Product, the need for sign-off and the Sustainability on Developers

The users or product owner needs to be ready and available for prompt testing of the features as they are delivered and throughout the entire duration of the project. This can be quite time-consuming but helps drastically to ensure a quality product that meets user expectations.

Common feedback is that agile development is rather intense for developers. The need to really *complete* each feature 100% within each iteration, and the relentlessness of iterations, can be mentally quite tiring so it's important to find a sustainable pace for the team.

Under Dell's Waterfall methodology, projects typically went through peaks and troughs, with teams expected to go the extra mile at the end of a phase exit to achieve the milestones in order to meet the deadline. Under the Agile approach the pressure on teams is relentless, there is zero downtime. The demand for weekend work and long hours is completely unsustainable and the result is a major increase in staff turnover.

Key takeaways 4.3 – While the flexibility that Agile provides can be a major benefit, it also leads to limitations caused by the lack of a strong and clear vision, and the discipline of fixing timescales and trading scope. This is further accentuated by the level of detail provided up front in the requirements. The adoption of the Behaviour Driven Development (BDD) process is undoubtedly a positive approach to reducing defects and results in finding issues earlier in the development lifecycle, however, this comes at a cost as significant extra resources and engagement are required by the test teams. Finally, the sustainability of continuous delivery on Developers is a major concern, the Pivotal Agile model is attempting to address this by time boxing the Development effort to a strict 9am-5pm.

4.3.2 Lessons Learned

One Size does not fit all

Dell have an established and mature Agile team in their Dell Commerce Services organization, these team are responsible for Dell's online and offline Sales and Order Entry applications. These applications are established platforms which lend themselves to an Agile approach. They can be regularly updated with new functionality without impacting their Business Partners.

With an ERP rollout to a green field Business Unit, in Dell's case there are 80+ Applications which are interdependent on each other. The nature of an ERP Program is that it's an "All or nothing" approach, we cannot deliver the General Ledger module without the sub-ledgers, such as Accounts Receivable (AR) and we cannot deliver AR without the Order Entry module or the related Collections, Payments modules etc. In effect this has forced Dell to take a hybrid Agile approach whereby we are Agile through the Development cycle and revert to Waterfall during the end to end testing and user acceptance testing cycles. This has resulted in a major increase in Change Requests (CR) and Defects. The increase in defects is due to the lack of review in the early design phase of the Program which was previously forced on teams as part of the Waterfall Design Phase Exit review. Under Agile this no longer happens, so the result is that teams only get a full view of the interlocking team's solution during end to end testing, hence the large number of CR's and defects. Agile recommends a weekly Scrum of Scrums meeting to mitigate this situation, the intention is that all interlocking teams come together to discuss dependencies and clarify solutions. Given the nature of Agile's continuously evolving backlogs and Sprint plans, this is not a catch all approach.

Business Alignment and Engagement

One of the main benefits of taking an Agile approach is that the Business Partners will be able to see the functionality earlier, this has been the case at Dell and the feedback from the Business Partners is that they appreciate the opportunity to see demonstrations of functionality every two weeks as part of the Sprint cycle, this allows them to find issues early and is a clear selling point. Where frustration grows is in an ERP environment, none of this functionality can be switched on in a live Production environment until everything is ready to be switched on in one “Big Bang”. In effect this can mean that the functionality reviewed 9 months earlier is now only coming to fruition. For the Business Partners, this is what they experienced under the Waterfall methodology.

Agile requires the Business Partners to be significantly more invested in the Project than they would have been in a Waterfall approach. They are expected to be part of Sprint reviews every two weeks through the entire Program. This is a major overhead as the majority of Business Partners have “day jobs” which will always need to take priority, in the case of an Accountant at month end.

Funding Model

Dell’s annual budget process is based on a Waterfall approach which has clearly defined Projects with agreed scope, schedules and resources. True Agile does not align to this budget approach as delivery is supposed to be continuous and the backlog decremented as time and budget allows, this is not feasible for an ERP Program.

The increased cost of testing and the impact on Development resources requires a cost benefit analysis. A tripling of the cost of testing and significant increase in Development resource turnover would bring into question the true “value” of adopting an Agile approach in an ERP environment.

Release and Environment Management

Dell’s Release management processes are not aligned with Agile delivery, this causes major challenges with regard to code management and the planning of releases windows. This impacts the Development teams who find themselves having to continuously move environments and disable code so as not to impact other Programs co-existing in the same environment. This has an obvious impact on test quality and adds major risk to release management.

In summary, the research questions outlined in this Chapter described and interpreted the findings of the Dell Survey and looked specifically at Dell's experience in a large multiphase ERP Program. It focused on the research objectives of the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program and illustrated that through the lens of the Agile ERP Quadrant.

It provided concrete examples of Dell's experience across these categories and tied back the findings to the company wide findings outlined in Chapter 3. This section of Chapter 4 concluded that the adoption of Agile is not a one size fits all and highlights specific areas associated with ERP Programs that require a more hybrid approach. Section 4.4 will go into detail on Dell's Agile Adoption.

4.4 Dell's Agile Adoption

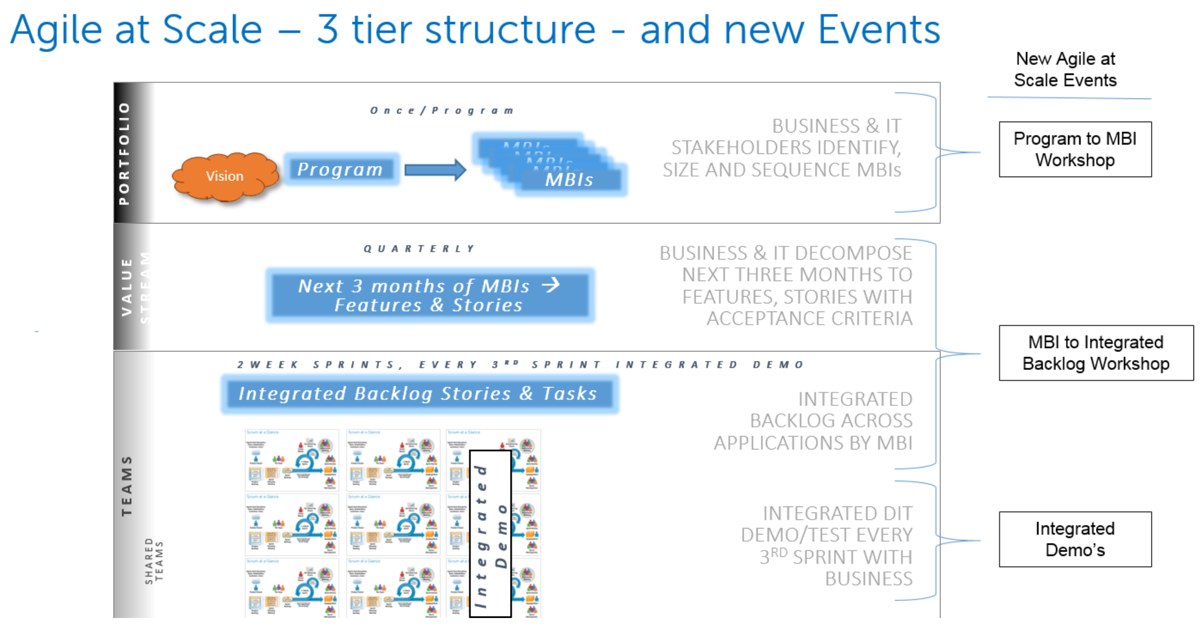
Following on from a successful adoption of Agile in Dell's Commerce Services function, Dell now wants to widen the adoption of Agile and Agile-At-Scale to the wider Dell IT community. Dell's Commerce Services function were early adopters of Agile and had over two years of a head start on other functions. The benefits of monthly code delivery were one of the key drivers in Dell's wider IT community following suit.

The goal is to enable IT to **deliver business value faster** and more efficiently by removing Delays. The approach is to

- Demo and get feedback faster
- Removing waste and friction from our process

  The Plan of approach is to

- Break business programs down into "**Minimum Business Increments**"
- Decomposing MBI's into Features and Stories and **planning ~3 months of work across all feature teams** with deliverables for that MBI
- Getting **visibility** to the status of the work within an MBI
- **Increasing the frequency of demonstration** and integration of completed work across the MBI.



Agile at Scale – 3 tier structure - and new Events
New Agile at Scale Events
PORTFOLIO
Once/Program
Vision
Program
MBIs
BUSINESS & IT STAKEHOLDERS IDENTIFY, SIZE AND SEQUENCE MBIs
Program to MBI Workshop
VALUE STREAM
QUARTERLY
Next 3 months of MBIs → Features & Stories
BUSINESS & IT DECOMPOSE NEXT THREE MONTHS TO FEATURES, STORIES WITH ACCEPTANCE CRITERIA
MBI to Integrated Backlog Workshop
TEAMS
SHARED TEAMS
2 WEEK SPRINTS, EVERY 3RD SPRINT INTEGRATED DEMO
Integrated Backlog Stories & Tasks
Integrated Demo
INTEGRATED BACKLOG ACROSS APPLICATIONS BY MBI
INTEGRATED DIT DEMO/TEST EVERY 3RD SPRINT WITH BUSINESS
Integrated Demo's

Fig 4.7 Agile at Scale – 3 Tier structure and new Events.

DellPrint Transform Program:

DellPrint is an investment by Dell Technologies to replace the old tools and ways of working with a new approach which will enable Dell to lead as an end-to-end technology solutions provider. "DellPrint" refers to the new transformational blueprint which spans 100+ IT Applications, with a goal to create simple, global, and consistent ways of working.

Key issues for Dell:

Disconnected scrum teams across applications – Prior to Dell's DellPrint Agile journey, IT Development Teams were very much siloed, they had a deep knowledge of their specific application but had a limited knowledge of other applications in their function and very little knowledge of applications outside their function.

Lack of clarity of requirements – There was a culture of providing very high level, ambiguous requirements in the planning stage of a project and these were not revisited until the user acceptance testing phase, this resulted in a high number of change requests being uncovered very late in the development cycle.

Late discovery of issues – the waterfall methodology resulted in the majority of code defects being discovered at the latter end of the testing phase, this was often too late in the cycle and resulted in functionality needing to be deferred to a future date to all for resolution.

What are Dell IT doing differently?

- Decomposing DellPrint Programs into smaller increments – an MBI
- Sequencing MBI's, completing development of functionality and demonstrating along the way to get feedback faster
- Integrated Dev backlog across applications covering next 3 months - Decomposing top MBI's to detailed features & stories
- Integrated demonstrations every 6 weeks (every 3rd sprint) to show dev progress, obtain feedback and reduce time & issues in final SIT/UAT
- All Dell IT work in one place in one TFS (Team Foundation Server)

<u>Business Participation Required</u>

- Participation in decomposing each program into meaningful demonstrate-able functionality
- Participation in sequencing the functionality so IT works on the biggest “bang for the buck” first
- Defining the detailed acceptance criteria for features to be developed in the next 3 months
- Participate in demonstrations, provide feedback – ideally the same people that would be in UAT.

4.4.1 Scrum Workflow

This diagram outlines the Scrum and Agile Workflow. In this section I will provide an in-depth analysis of the activities and artefacts associated with this globally recognised Scrum framework as defined by the scrum alliance, the scrum governing body.

Section 4.4.1 outlines the process and procedures contained in the scrum framework.

- **Product Planning**
- **MBI Grooming**
- **Sprint Planning**
- **Product Deployment**

Section 4.4.2 provides the details behind each of the activities and artefacts
Section 4.4.3 outlines Dell's Agile-At-Scale IT Delivery Framework and how Dell adopted the scrum framework as laid out in sections 4.4.1 and 4.4.2.

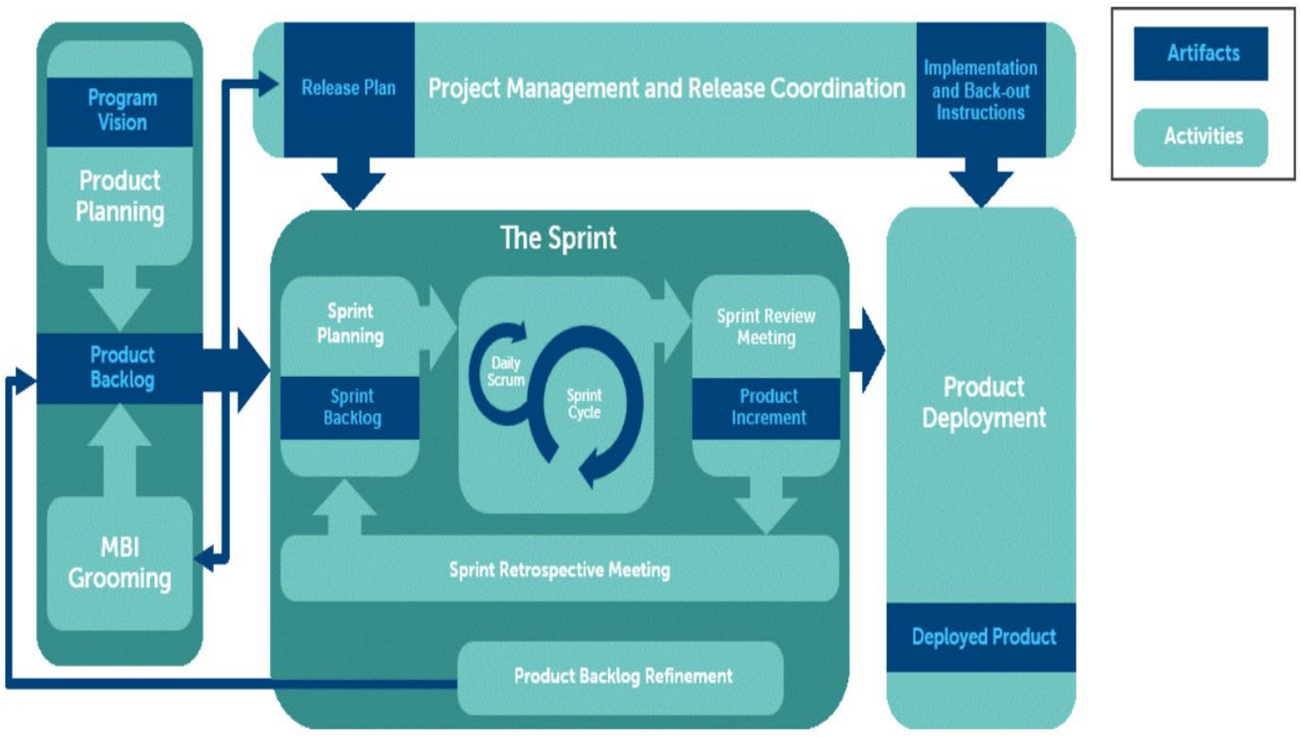


Fig 4.8 Scrum Workflow

4.4.2 Review and Confirm the Program Vision

The Product Owner is responsible for the success of the product and its Return On Investment (ROI), and should therefore make sure the Program Vision Document is understood and sets the direction to guide the whole Scrum Team.

The Program Vision Document (PVD) should be completed as part of the project funding process. In case the project has been initiated without a PVD, the Product Owner should take all the project Initiation documentation (e.g., BRD, MRD, Project Charter) as input to produce a PVD.

As Ken Schwaber [25] puts it: "The minimum plan necessary to start a Scrum project consists of a vision and a Product Backlog. The vision describes why the project is being undertaken and what the desired end state is." Schwaber (2004), p. 68) [25]

Sometimes the Product Owner may not be fully knowledgeable for all the business aspects that pertain to a product. In that case, it is expected that other business stakeholders should provide inputs to support the Product Owner in order to build a consistent Program Vision.

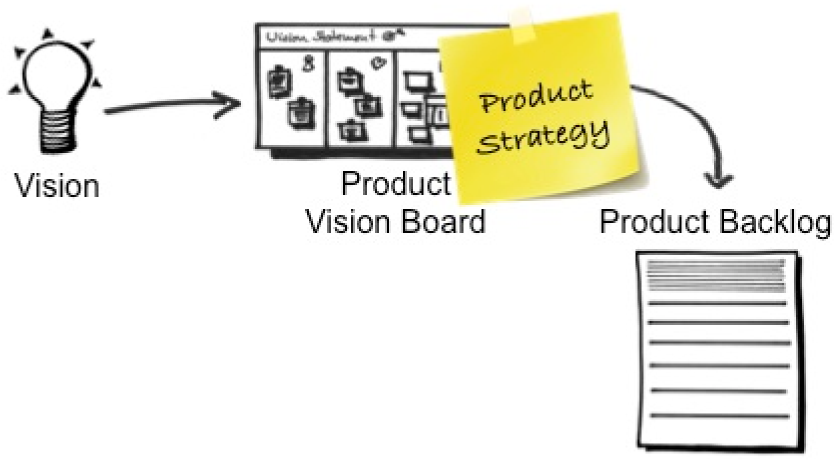


Fig 4.9 Program Vision

4.4.2.1 Refine Business Objectives

The Product Owner is responsible to refine the business objectives stated on the Program Vision Document. Objectives must be transparent to everyone and are typically in the form of Minimum Business Increments (MBIs) in the product backlog. The Scrum Team works together to refine the objectives, but the Product Owner is the focal point for managing objectives.

Objectives have the following characteristics:

- **Objectives need refinement**. Objectives can't be written at one time. Thus, user story writing workshops and product backlog refinement events take place.
- **Objectives need to be broken down**. Epics are split into stories. Stories are split into tasks. The product backlog is split into a series of sprint backlogs. The sprint backlog is the output of sprint planning.
- **Objectives need be verified**. The product increment is the implementation of the sprint goal and the Definition of Done. All of these are verified in the Sprint Review.

Create Product Backlog

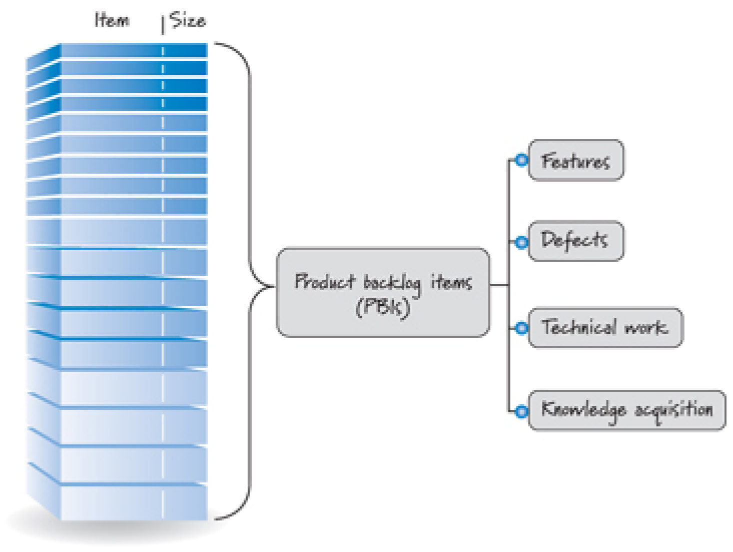


Fig 4.10 Create Product Backlog

Once the Product Owner has collected the business objectives, he or she then compiles these business objectives into a formal structure known as the Product Backlog. The Product Owner will ensure that the Product backlog lists all presently known features, functions, requirements, enhancements, and fixes that will drive the changes to be made to the product.

A good Product Backlog items should have the following attributes:

- **ID** - A unique identifier
- **Description** - A description of the need feature or function needed
- **Category** - A method of grouping similar backlog items (e.g., feature, documentation, knowledge acquisition, user training material, etc.)
- **Priority** - An indication of the importance of the item relative to other items
- **Size Estimate** - described in Story Points to indicate relative sizing.
- **Value** - An indicator of business value.
- **Acceptance Criteria (AC)**: what is necessary for the Product Owner to accept the item as completed. Consider the following example:
- o **User Story example**: As a Frequent Rewards Customer,” I want to get a discount when purchasing a product, so that I can be rewarded for spending a lot of money.”
- o **Acceptance Criteria**: For the above story, the acceptance criteria could be to demo that: I receive a 5% discount if my rewards credit is greater than 50 points over the past year; I do not receive a discount if my rewards credit is less than 50 points over the past year; I receive 10% discount if my rewards points are greater than 100 over the past year.

  Other important elements of the Product Backlog are the Definition of Ready (DoR) and the Definition of Done (DoD):

- The DoR means what information the Development Team needs in order to be able to **start** working on the item. Example: Stories are created in TFS, dependencies on external systems determined, acceptance criteria defined.
- The DoD represents what the Development Team must execute in order to call the item as **completed**. Example: Software is promoted to the SIT environment, automated tests created, SIT test passed.

4.4.2.2 Project Management and Release Coordination

The purpose of the **Project Management and Release Coordination** activity is:

- Identify the release scope and goals
- Plan the Compliance tasks required for the release
- Plan the schedule of the release, and determine the number of Sprints required
- Plan the test strategy for the release
- Ensure communication when there are interdependencies between Scrum Teams
- Plan for Implementation related activities

A release usually consists of functionality being developed over multiple Sprints. Therefore, it is necessary to plan the scope of the release, the number and the duration of Sprints followed by the Product Deployment activity.

4.4.2.3 Define Release Scope and Coordinate Dependencies

Conduct Release Planning Meeting:
Release scoping is an approach used to realize the benefits of incrementally delivering business requirements. The first step is conducting the initial Release Planning Meeting. This meeting is facilitated by Product Owner for the team.

Everyone who is impacted by the release needs to be in the Release Planning Meeting. They are needed to help develop the plan, identify dependencies, and forecast the release. If people are missing, information will be missing.

- Use the Release & Sprint Planning Agendas to assist with Release Planning.
- The Release Plan Template may be used to document all the necessary release information.
- For reference on how to slice the scope in Minimum Business Increments, see MBI section below.
- Notice that the Release Scope is not a static document. It will be alive during the all project lifecycle, and the scope will be updated as result of changes in the Product Backlog.

During this meeting, the Product Owner and Scrum Team:

Presents/Reviews the prioritized user stories to be delivered in the current release and indicates which stories to go into which Sprints based on Business value priority, size of product backlog items, risk and dependencies.

Considers Infrastructure Needs for the Release: For the high-priority user stories that are most likely to be part of the upcoming Release, consider what infrastructure or design considerations will be created or impacted.

The following topics should also be covered in the meeting:

1. The Release scope and time-box
2. The Release Definition of Ready and Definition of Done
3. The Start and End dates for the release
4. The number and duration of Sprints
5. Sprint Definition of Ready and Definition of Done
6. The members of the Scrum Team and the Delivery Resources that may be needed
7. Concern, risks and issues
8. Code Lock Security scans and Penetration testing requests. The metrics that will be collected and tracked for each Sprint and for the release

Coordinate Dependencies:
Impediments affecting one Scrum team might have an impact on other Scrum teams. Coordination between Product Owners and Technical Program Managers is needed to solve such impediments. It may be necessary to adopt changes in scope (or in scope priorities) in order to work around impediments.

In order to coordinate the work (and the removal of cross dependencies) a good practice is to establish a Scrum of Scrums process.

Define Team Norms:

Given the members of the Scrum Team are defined as part of the Release Planning Meeting, it is a good practice to hold a meeting with the specific purpose of establishing the Team Norms. For guidance on how to create an initial set of Team Norms.

Create Release Plan:

Based on the results of the Release Planning Meeting, the initial Release Plan is created and shared with the team.

For projects that utilize highly coupled code structures that prevent end-to-end testing to be completed during the Sprints, a **Hardening Sprint** can be incorporated in the Release Plan. The Hardening Sprint should have the following characteristics:

- It has the same length as any regular Sprint in the project
- The Hardening Sprint Goal should be always to complete the end-to-end testing for the respective release
- The Hardening Sprint Backlog includes only end-to-end testing, and the fix of defects found during this test

Notice that IT teams should avoid the need of a Hardening Sprint by decoupling the source code as much as possible, therefore minimizing the impact of possible interlocks. The Hardening Sprint should be used only as an alternative when the project fails in eliminating those interdependencies.

Note: Performance Testing is also completed as part of Hardening Sprint if not completed in respective Sprints. The Performance testing team and the Scrum team decide if Performance testing is to be conducted during the sprint or during the Hardening Sprint depending on the Business needs.

Create Implementation and Back-out Instructions:

Implementation and Back-out Instructions is a technical document that provides the details of the sequence of events that will need to be executed during the launch. Due to its technical content, it is usually elaborate by the Development Team members.

4.4.2.4 Sprint Planning

The purpose of the Sprint Planning activity is to:

- Determine the Sprint Goal.
- Establish the Sprint Backlog, as a mean to achieve the Sprint Goal
- Forecast the Sprint Backlog

Conduct Sprint Planning Meeting

The Technical Program Manager ensures that every Sprint starts with a Sprint Planning meeting. The objective of this meeting is to determine what can be delivered in the upcoming Sprint and how the work will be achieved. The Sprint Planning Meeting must include the Product Owner and the Development Team.

Propose the Sprint Scope

The Product Owner discusses the objective that the Sprint should achieve based on prioritization, or value to the business. The PO should also consider compliance constraints and non-functional requirements. The Development Team then discusses each of these items to ensure clarity on the items and the interdependencies between them.

For every backlog item being proposed for Sprint Backlog, the Product Owner must also provide a clear statement of the **Acceptance Criteria** for that item. This criterion will be used later, during the Sprint Review for the Product Owner to assess the completion of that backlog item.

The Development Team forecasts the functionality that will be developed during the Sprint.

The Development Team decides how it will build this functionality into a “Done” product Increment during the Sprint.

Establish the Sprint Backlog

Generally, all high-order Product Backlog items would already have an estimated effort, usually measured by points-based results from the Product Planning and Product Backlog Refinement activities.

If any of the Product Backlog items proposed do not have an estimated effort, the Development Team may estimate it at this time.
Below are some other items that must be considered while establishing the Sprint Backlog:

(a) Capacity of the team: When the Development Team forecasts how much work they can complete in the sprint they should consider any known impacts to their capacity over the timebox of the upcoming sprint - such as holidays and vacations, or members joining/leaving the team.

(b) Velocity of the team: Team Velocity concept is based on the assumption that the amount of work a team will do in the coming sprint is roughly equal to what they’ve done in prior sprints. A team's velocity is determined by historical average amount of Story Points that the team is able to complete per Sprint. This implies that, if you are planning a team's first Sprint, you won't have historical data, and will not be able to use Velocity as a factor to calibrate your Sprint Planning.

(c) Compliance with policies and Standards: The Development Team must comply with Dell Policies and Standards as IT solutions are designed, and should consider how these constraints should be reflected in the backlog items proposed for the Sprint:

- Global Information Lifecycle Management Policy
- Global IT Records Keeping Guidance Policy
- Secure Application Development Standard
- Use of Approved Technology Standard

(d) Any Impediments or dependencies on other teams or resources: If there are impediments that will prevent the Development team from completing any of the proposed Product Backlog items, the Development Team will generally not commit to include these in the Sprint, unless there is an expectation that Technical Program Manager has committed to removing the impediments to allow reasonable time to complete them within the Sprint.

Based on the proposed scope, the **Development Team** performs analysis to determine the feasibility of the Increment represented by the proposed Sprint Backlog as well as determine how it will build this functionality into a "Done" product Increment during the Sprint. Work may be of varying size, or estimated effort. However, enough work is planned during Sprint Planning for the **Development Team** members to forecast what they believe they can do in the upcoming Sprint.

Define the Sprint Goal

The Sprint Goal is an objective set for the Sprint that can be met through the implementation of items in the Product Backlog. It provides guidance to the Development Team on why it is building the Increment. The Sprint Goal gives the Development Team some flexibility regarding the functionality implemented within the Sprint. The Sprint Goal should focus the Development Team to work together rather than on separate initiatives.

The Sprint Goal is used to help focus the Scrum Team on the objectives of the sprint, the higher purpose of why the sprint is necessary. If at a later stage, the work turns out to be different than the Development Team expected, they collaborate with the Product Owner to negotiate the scope of Sprint Backlog within the Sprint.

Examples of Sprint Goals

- To deliver a functioning webpage that highlights all of the offerings for a particular line of business.
- To deliver a physician search feature on a webpage
- To update a mobile application to allow users to check account balance

Accept Sprint Goal and Sprint Backlog?

The entire Scrum Team collaborates on understanding the work included in the Sprint Backlog. The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team. Only the Development Team can assess what it can accomplish over the upcoming Sprint.

By the end of the Sprint Planning, the Development Team should commit to the Sprint Backlog and be able to explain to the Product Owner and Technical Program Manager how it intends to work as a self-organizing team to accomplish the Sprint Goal and create the anticipated Increment.

Complete Sprint Planning

Work planned for the first days of the Sprint by the Development Team is decomposed by the end of this meeting, often to units of one day or less. The Development team may break the backlog items into tasks. Task estimates in hours will help balance the work effort among the team, as well as to help gauge the remaining effort during the Sprint Cycle.

Product Deployment

The purpose of Product Deployment Activity is:

- Conduct End-to-End testing (if it could not be completed during the Sprint)
- Deploy the product to the production environment
- Conduct User Training (as necessary)

4.4.2.5 Conduct End-to-End Testing

Once it has been determined that product is ready for deployment, the product increment may be subject of End-to-End testing as part of the product release process.

- Test Cases and Defects should be created and managed in Team Foundation Server (TFS).
- See the "Test Case Work Item" and "Defect Work Item" training material

Deploy Product Increment to Production

During the deployment of the Product Increment to production, it must be ensured that the code is deployed to the appropriate environment and that post-install testing occurs to validate a successful deployment, in accordance with Release Management guidelines.

The following steps are to be executed:

Conduct Go/No Go Meeting (if needed) - A formal Go/No Go meeting can be conducted to ensure IT and business management agree that the implementation can proceed as planned. The Technical Program Manager facilitates the meeting. The Product Owner must attend, and other Scrum Team and Business Stakeholders should attend as the team determines is appropriate.

Perform Launch Sequence of Events - The "Installation Instructions", as documented in the Implementation and Back-out Instructions, are executed by the designated implementers. These steps contained the detailed instructions for deploying the system to production environment. The launch is coordinated by the Technical Program Manager or a designee.

Perform Post-Install Testing - The designated IT team members and business team members perform the documented steps to validate the business operations using the new or enhanced system. These steps are intended to discover any unanticipated side effects caused by deploying the new system components into production. They are documented as the “Deployment Validation” steps in the Implementation and Back-out Instructions document.

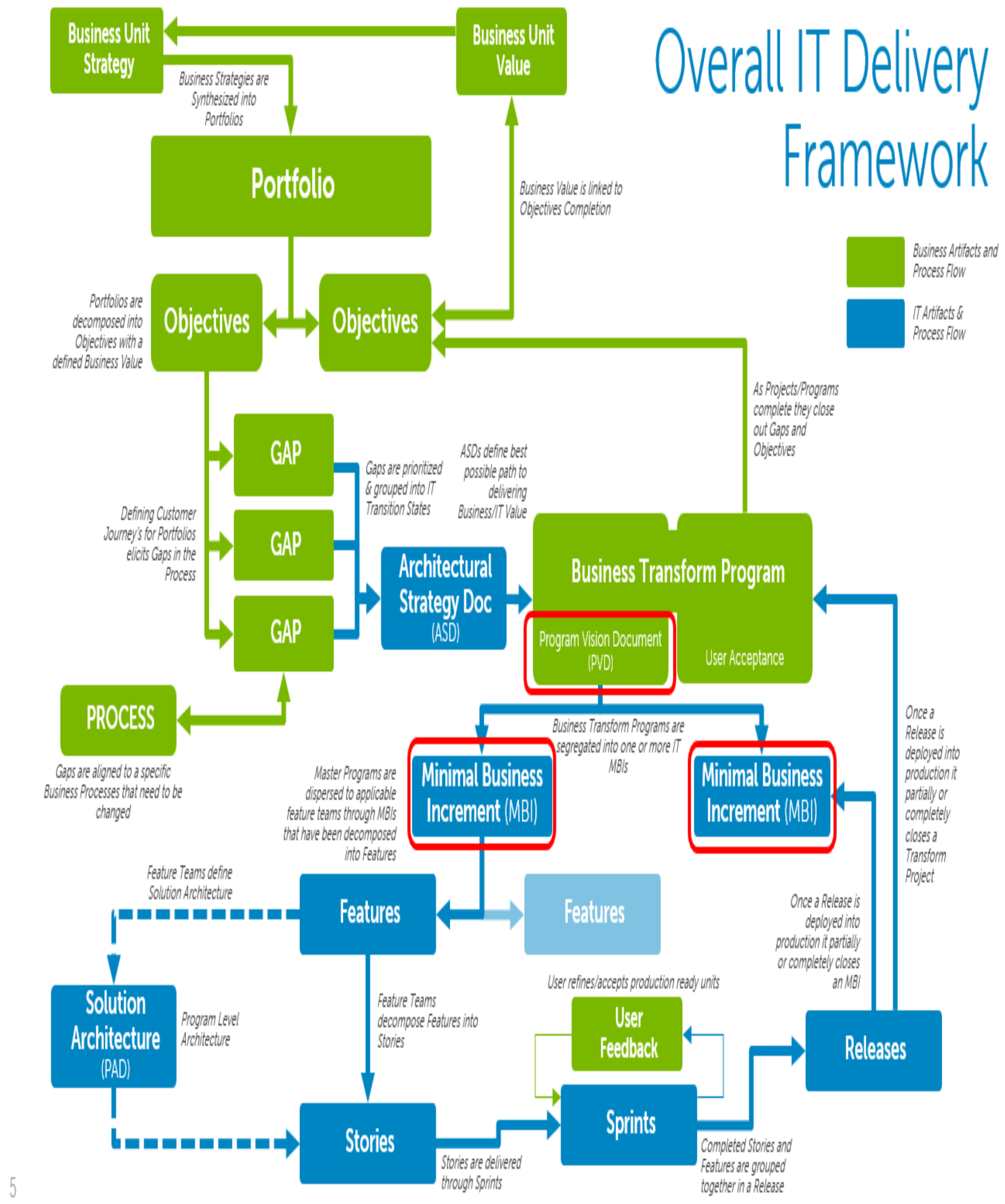
Complete or Back Out Deployment - Based on the results of the deployment, the designated personnel execute either the "Supplemental Tasks" to finish out the deployment or the "Rollback Instructions" to restore the environment to pre-implementation stage. These instructions are documented in the Implementation and Back-out Instructions.

Create and Conduct User Training

The Product Owner is responsible to ensure User Training Material is created and training is conducted, if necessary.

4.4.3 Dell's Agile-At-Scale IT Delivery Framework

In this section, we look at the business artifact, the Program Vision Document (PVD) and the associated IT Minimum Business Increment (MBI)



Overall IT Delivery Framework
Business Unit Strategy
Business Unit Value
Business Strategies are Synthesized into Portfolios
Portfolio
Business Value is linked to Objectives Completion
Business Artifacts and Process Flow
IT Artifacts & Process Flow
Portfolios are decomposed into Objectives with a defined Business Value
Objectives
Objectives
As Projects/Programs complete they close out Gaps and Objectives
GAP
GAP
GAP
Gaps are prioritized & grouped into IT Transition States
ASDs define best possible path to delivering Business/IT Value
Defining Customer Journey's for Portfolios elicits Gaps in the Process
Architectural Strategy Doc (ASD)
Business Transform Program
Program Vision Document (PVD)
User Acceptance
PROCESS
Gaps are aligned to a specific Business Processes that need to be changed
Business Transform Programs are segregated into one or more IT MBIs
Master Programs are dispersed to applicable feature teams through MBIs that have been decomposed into Features
Minimal Business Increment (MBI)
Minimal Business Increment (MBI)
Once a Release is deployed into production it partially or completely closes a Transform Project
Feature Teams define Solution Architecture
Features
Features
Once a Release is deployed into production it partially or completely closes an MBI
User refines/accepts production ready units
Solution Architecture (PAD)
Program Level Architecture
Feature Teams decompose Features into Stories
User Feedback
Releases
Sprints
Stories
Stories are delivered through Sprints
Completed Stories and Features are grouped together in a Release
5

Fig 4.11 Dell's Agile-At-Scale IT Delivery Framework

Defining an MBI

- In the context of a large funded program, it is the minimum amount of incremental business value that can be built, deployed, and consumed and that makes sense from a business perspective. The business and IT together are responsible for defining MBIs.
- A Product Manager in IT 'owns' the definition and delivery of the MBI. Ideally, there will also be a single point of accountability in the business.
- A funded program should be broken up into several MBIs. An MBI should be as small as possible (minimum) and it should be less than 3 months of work.

Sequencing the MBI's

- The MBIs within a program will then be sequenced using both the MBI relative value and relative size to deliver the 'biggest-bang-for-the-buck' business value first.
- MBI's will be finished and delivered to the business along the timeline for the completion of the entire program. Completed MBI's will be the milestones of projects, not waterfall milestones.
- All feature teams, regardless of application, segment, organization, share the same sequence/priority. The backlog for the most important MBI is more important than the next MBI.

Linking Backlog and Deployments

- All Features required to deliver an MBI will be linked to that MBI in TFS (Team Foundation Server) via parent/child links.
- Even if the business decides not to deploy an individual MBI into production, IT will still take an MBI all the way through the Dev and Test process and get the functionality ready to deploy.

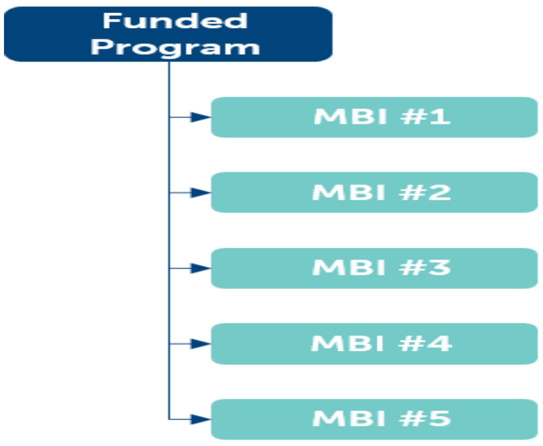


Fig 4.12 Funded Program

Benefits of using MBIs

- Better alignment with the business: By breaking up a program into MBI’s with the business present and creating an integrated backlog with the business present, we have more opportunity for the development teams and engineers in IT to interact with the business and clarify requirements.
- Smaller batch sizes enabling Faster time to Value: Break large programs into smaller chunks, sequence them, and deliver them incrementally.

– By delivering projects incrementally, we can deliver the first minimum business increment sooner and the business can start to use that to deliver value to the company.

– Interlock problems solved (or minimized): TPMs can now track the work of interlocks altogether with your program scope.

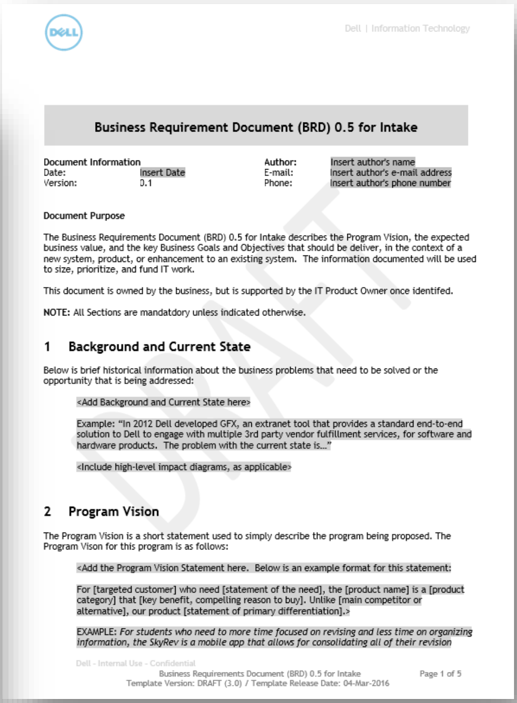
– Increased visibility: Full visibility in TFS2, from MBIs burn up charts down to Feature team & sprint level views, all in TFS2

- Eliminate waste by finding defects faster: earlier end to end integration allows for defects to be discovered faster. The shorter the time between creating and discovering a defect, the less ‘defect interest’ has to be paid.
- No more manual reporting: An IT Analytics tool provides standardized MBI reports when your backlog is properly configured.

The PVD

The Program Vision Document describes the Program Vision, the expected business value, and the key Business Goals and Objectives that should be deliver. The information documented will be used to size, prioritize, and fund IT work. The document contains 5 major sections:

- The Background and Current State, brief historical information about the business problems that need to be solved or the opportunity that is being addressed.
- The Program Vision, a short statement used to simply describe the product being developed.
- The Program Scope described in terms of Business Goals and Gaps (aka MBIs).
- Additional Requirements to describe data, volume and usage needs.
- Program Management Details, describing known risks, issues, assumptions, constraints, etc.



Dell | Information Technology

Business Requirement Document (BRD) 0.5 for Intake

Document Information		**Author:**	Insert author's name
Date:	Insert Date	E-mail:	Insert author's e-mail address
Version:	0.1	Phone:	Insert author's phone number

Document Purpose

The Business Requirements Document (BRD) 0.5 for Intake describes the Program Vision, the expected business value, and the key Business Goals and Objectives that should be deliver, in the context of a new system, product, or enhancement to an existing system. The information documented will be used to size, prioritize, and fund IT work.

This document is owned by the business, but is supported by the IT Product Owner once identifed.

NOTE: All Sections are mandatdory unless indicated otherwise.

1 Background and Current State

Below is brief historical information about the business problems that need to be solved or the opportunity that is being addressed:

<Add Background and Current State here>

Example: "In 2012 Dell developed GFX, an extranet tool that provides a standard end-to-end solution to Dell to engage with multiple 3rd party vendor fulfillment services, for software and hardware products. The problem with the current state is…"

<Include high-level impact diagrams, as applicable>

2 Program Vision

The Program Vision is a short statement used to simply describe the program being proposed. The Program Vison for this program is as follows:

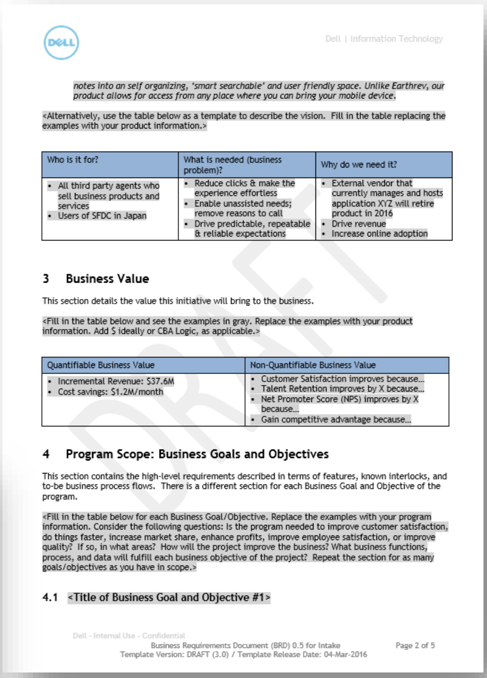
<Add the Program Vision Statement here. Below is an example format for this statement:

For [targeted customer] who need [statement of the need], the [product name] is a [product category] that [key benefit, compelling reason to buy]. Unlike [main competitor or alternative], our product [statement of primary differentiation].>

EXAMPLE: *For students who need to more time focused on revising and less time on organizing information, the SkyRev is a mobile app that allows for consolidating all of their revision*

Dell - Internal Use - Confidential

Business Requirements Document (BRD) 0.5 for Intake Page 1 of 5
Template Version: DRAFT (3.0) / Template Release Date: 04-Mar-2016



Dell | Information Technology

notes into an self organizing, 'smart searchable' and user friendly space. Unlike Earthrev, our product allows for access from any place where you can bring your mobile device.

<Alternatively, use the table below as a template to describe the vision. Fill in the table replacing the examples with your product information.>

Who is it for?	What is needed (business problem)?	Why do we need it?
• All third party agents who sell business products and services • Users of SFDC in Japan	• Reduce clicks & make the experience effortless • Enable unassisted needs; remove reasons to call • Drive predictable, repeatable & reliable expectations	• External vendor that currently manages and hosts application XYZ will retire product in 2016 • Drive revenue • Increase online adoption

3 Business Value

This section details the value this initiative will bring to the business.

<Fill in the table below and see the examples in gray. Replace the examples with your product information. Add $ ideally or CBA Logic, as applicable.>

Quantifiable Business Value	Non-Quantifiable Business Value
• Incremental Revenue: $37.6M • Cost savings: $1.2M/month	• Customer Satisfaction improves because… • Talent Retention improves by X because… • Net Promoter Score (NPS) improves by X because… • Gain competitive advantage because…

4 Program Scope: Business Goals and Objectives

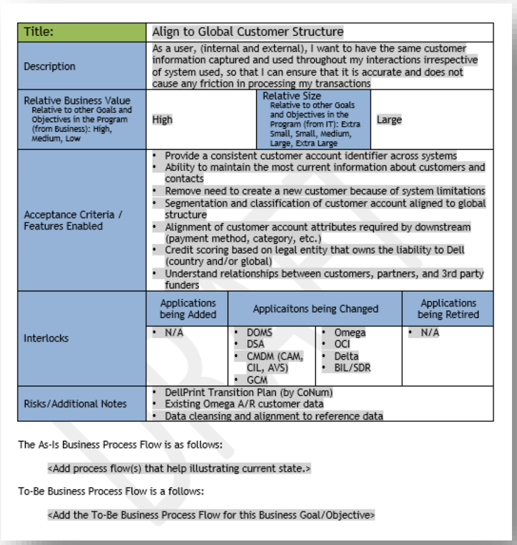
This section contains the high-level requirements described in terms of features, known interlocks, and to-be business process flows. There is a different section for each Business Goal and Objective of the program.

<Fill in the table below for each Business Goal/Objective. Replace the examples with your program information. Consider the following questions: Is the program needed to improve customer satisfaction, do things faster, increase market share, enhance profits, improve employee satisfaction, or improve quality? If so, in what areas? How will the project improve the business? What business functions, process, and data will fulfill each business objective of the project? Repeat the section for as many goals/objectives as you have in scope.>

4.1 <Title of Business Goal and Objective #1>

Dell - Internal Use - Confidential

Business Requirements Document (BRD) 0.5 for Intake Page 2 of 5
Template Version: DRAFT (3.0) / Template Release Date: 04-Mar-2016



Title:	Align to Global Customer Structure			
Description	As a user, (internal and external), I want to have the same customer information captured and used throughout my interactions irrespective of system used, so that I can ensure that it is accurate and does not cause any friction in processing my transactions			
Relative Business Value Relative to other Goals and Objectives in the Program (from Business): High, Medium, Low	High		Relative Size Relative to other Goals and Objectives in the Program (from IT): Extra Small, Small, Medium, Large, Extra Large	Large
Acceptance Criteria / Features Enabled	• Provide a consistent customer account identifier across systems • Ability to maintain the most current information about customers and contacts • Remove need to create a new customer because of system limitations • Segmentation and classification of customer account aligned to global structure • Alignment of customer account attributes required by downstream (payment method, category, etc.) • Credit scoring based on legal entity that owns the liability to Dell (country and/or global) • Understand relationships between customers, partners, and 3rd party funders			
Interlocks	Applications being Added	Applicaitons being Changed		Applications being Retired
	• N/A	• DOMS • DSA • CMDM (CAM, CIL, AVS) • GCM	• Omega • OCI • Delta • BIL/SDR	• N/A
Risks/Additional Notes	• DellPrint Transition Plan (by CoNum) • Existing Omega A/R customer data • Data cleansing and alignment to reference data			

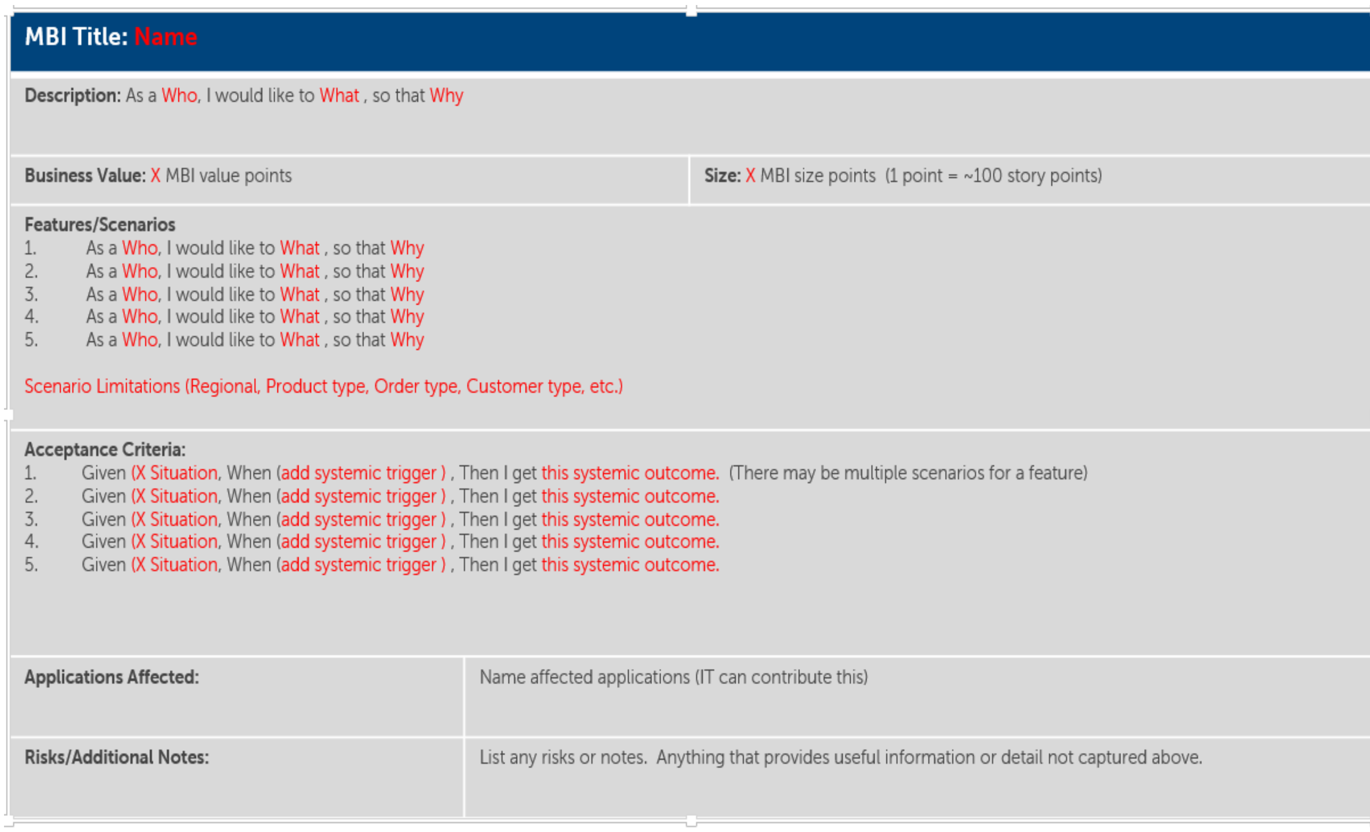
The As-Is Business Process Flow is as follows:

<Add process flow(s) that help illustrating current state.>

To-Be Business Process Flow is a follows:

<Add the To-Be Business Process Flow for this Business Goal/Objective>

Fig 4.13 MBI



MBI Title: Name

Description: As a Who, I would like to What , so that Why

Business Value: X MBI value points	**Size:** X MBI size points (1 point = ~100 story points)

Features/Scenarios
1. As a Who, I would like to What , so that Why
2. As a Who, I would like to What , so that Why
3. As a Who, I would like to What , so that Why
4. As a Who, I would like to What , so that Why
5. As a Who, I would like to What , so that Why

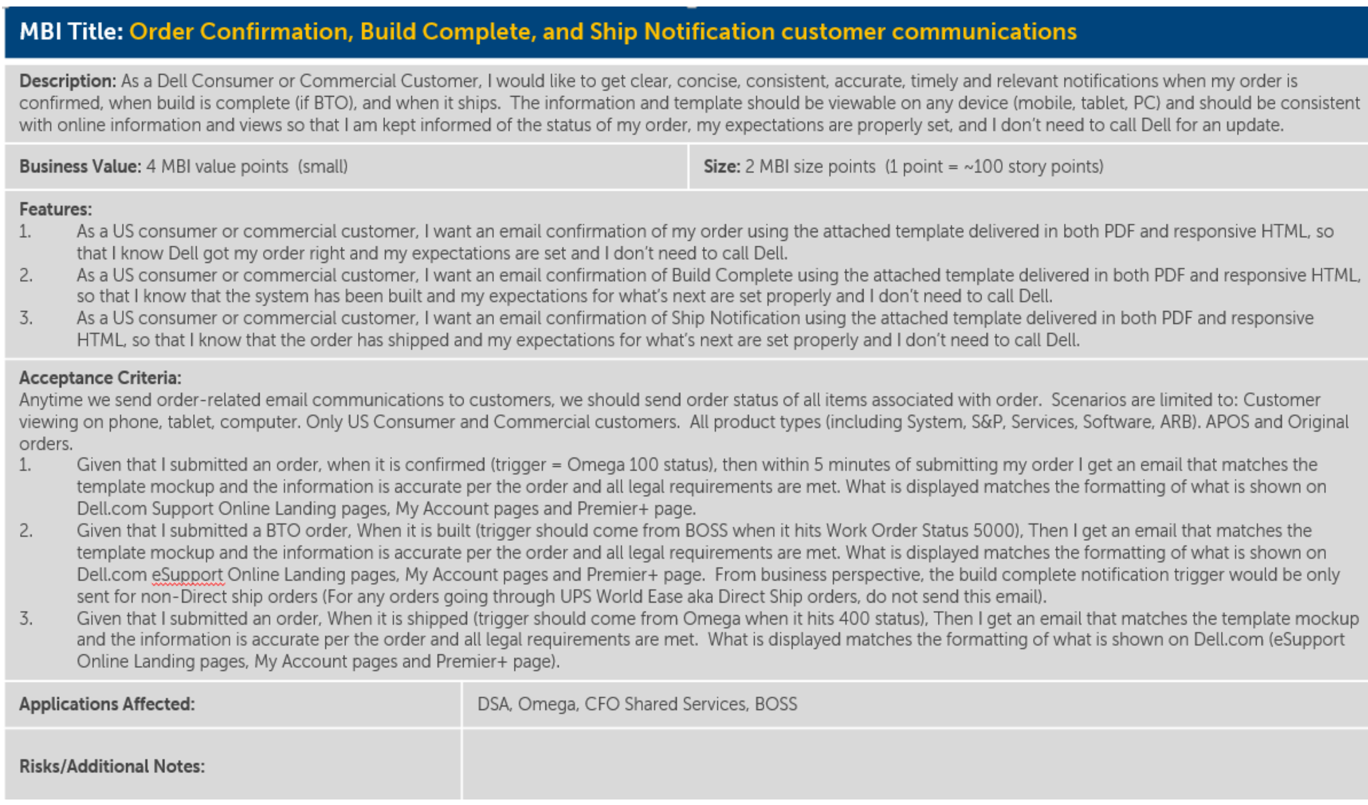
Scenario Limitations (Regional, Product type, Order type, Customer type, etc.)

Acceptance Criteria:
1. Given (X Situation, When (add systemic trigger) , Then I get this systemic outcome. (There may be multiple scenarios for a feature)
2. Given (X Situation, When (add systemic trigger) , Then I get this systemic outcome.
3. Given (X Situation, When (add systemic trigger) , Then I get this systemic outcome.
4. Given (X Situation, When (add systemic trigger) , Then I get this systemic outcome.
5. Given (X Situation, When (add systemic trigger) , Then I get this systemic outcome.

Applications Affected:	Name affected applications (IT can contribute this)
Risks/Additional Notes:	List any risks or notes. Anything that provides useful information or detail not captured above.

Fig 4.14 MBI Description

Example of an MBI



MBI Title: Order Confirmation, Build Complete, and Ship Notification customer communications

Description: As a Dell Consumer or Commercial Customer, I would like to get clear, concise, consistent, accurate, timely and relevant notifications when my order is confirmed, when build is complete (if BTO), and when it ships. The information and template should be viewable on any device (mobile, tablet, PC) and should be consistent with online information and views so that I am kept informed of the status of my order, my expectations are properly set, and I don't need to call Dell for an update.

Business Value: 4 MBI value points (small)	**Size:** 2 MBI size points (1 point = ~100 story points)

Features:

1. As a US consumer or commercial customer, I want an email confirmation of my order using the attached template delivered in both PDF and responsive HTML, so that I know Dell got my order right and my expectations are set and I don't need to call Dell.
2. As a US consumer or commercial customer, I want an email confirmation of Build Complete using the attached template delivered in both PDF and responsive HTML, so that I know that the system has been built and my expectations for what's next are set properly and I don't need to call Dell.
3. As a US consumer or commercial customer, I want an email confirmation of Ship Notification using the attached template delivered in both PDF and responsive HTML, so that I know that the order has shipped and my expectations for what's next are set properly and I don't need to call Dell.

Acceptance Criteria:
Anytime we send order-related email communications to customers, we should send order status of all items associated with order. Scenarios are limited to: Customer viewing on phone, tablet, computer. Only US Consumer and Commercial customers. All product types (including System, S&P, Services, Software, ARB). APOS and Original orders.

1. Given that I submitted an order, when it is confirmed (trigger = Omega 100 status), then within 5 minutes of submitting my order I get an email that matches the template mockup and the information is accurate per the order and all legal requirements are met. What is displayed matches the formatting of what is shown on Dell.com Support Online Landing pages, My Account pages and Premier+ page.
2. Given that I submitted a BTO order, When it is built (trigger should come from BOSS when it hits Work Order Status 5000), Then I get an email that matches the template mockup and the information is accurate per the order and all legal requirements are met. What is displayed matches the formatting of what is shown on Dell.com eSupport Online Landing pages, My Account pages and Premier+ page. From business perspective, the build complete notification trigger would be only sent for non-Direct ship orders (For any orders going through UPS World Ease aka Direct Ship orders, do not send this email).
3. Given that I submitted an order, When it is shipped (trigger should come from Omega when it hits 400 status), Then I get an email that matches the template mockup and the information is accurate per the order and all legal requirements are met. What is displayed matches the formatting of what is shown on Dell.com (eSupport Online Landing pages, My Account pages and Premier+ page).

Applications Affected:	DSA, Omega, CFO Shared Services, BOSS
Risks/Additional Notes:	

Fig 4.15 Example of an MBI

Sequencing MBIs – WSJF approach

Development teams can be most effective when focused on a very small set of items:

– Cycle time is better
– Multitasking is decreased
– Dependencies are fewer.

Simply prioritizing work as High, Medium, and Low would not accomplish this goal. The result of sequencing MBIs should be an ordering such that it's always clear which MBI is to be developed next.

Sequencing of MBIs is performed within Programs. Since Programs themselves are prioritized, the ordering of work across the organization should always be clear.

The Weighted Shortest Job First (WSJF) approach is recommended for sequencing MBIs. Here is the formula for calculating WSJF:

Using the Fibonacci sequence of values (1, 2, 3, 5, 8, 13, and 21), you should rate each parameter for each MBI against the other MBIs:

– For Business Value, consider:
– Business Impact: What is the revenue impact? Is there a potential penalty or negative impact if it is delayed? Do users prefer this MBI over others?
–

$$\textbf{WSJF} = \frac{\textbf{Business Value}}{\textbf{Job Size}}$$

– Time Criticality: How would the Business Value decay over time? Is there a fixed deadline? Are there dependencies from downstream programs?
– Risk Reduction-Opportunity Enablement Value (RR-OE): Will the MBI open up new business opportunities? What else does this MBI do for the business?
– For Job Size, consider: How big is the MBI? How long will it take to develop the MBI?

The MBI with the highest WSJF should be worked on first**.**

Sequencing MBIs – WSJF approach: An Example

Consider putting the information in a spreadsheet.

See this Example:

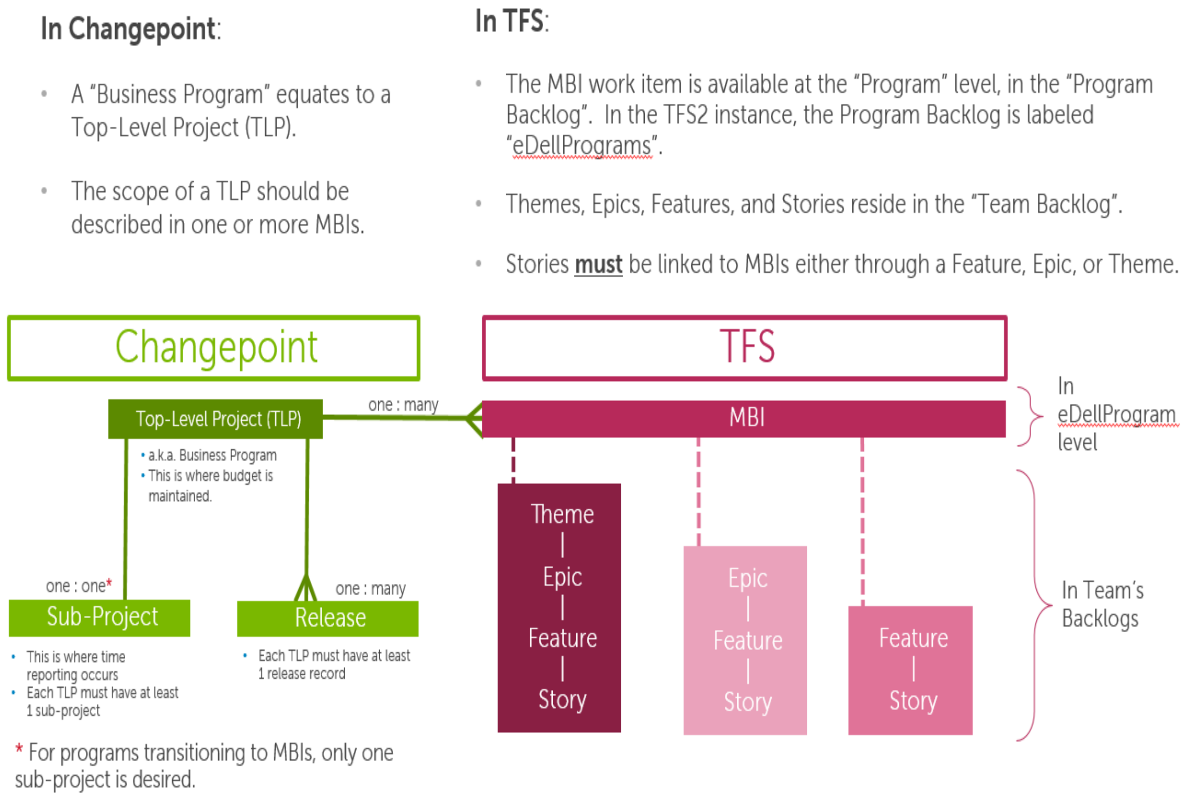
The MBI with the highest WSJF is the highest priority and should be worked on first.

In the example, MBI D would have the worked on first.

MBI	**Business Value**	**Job Size**	**WSJF**
MBI A	**5**	**3**	**1.7**
MBI B	**1**	**3**	**0.3**
MBI C	**13**	**8**	**1.6**
MBI D	**21**	**5**	**4.2**

Table 4.16

Changepoint (Dell's Project Portfolio Management Tool) and TFS (Requirement's Management Tool)



In Changepoint:

- A "Business Program" equates to a Top-Level Project (TLP).
- The scope of a TLP should be described in one or more MBIs.

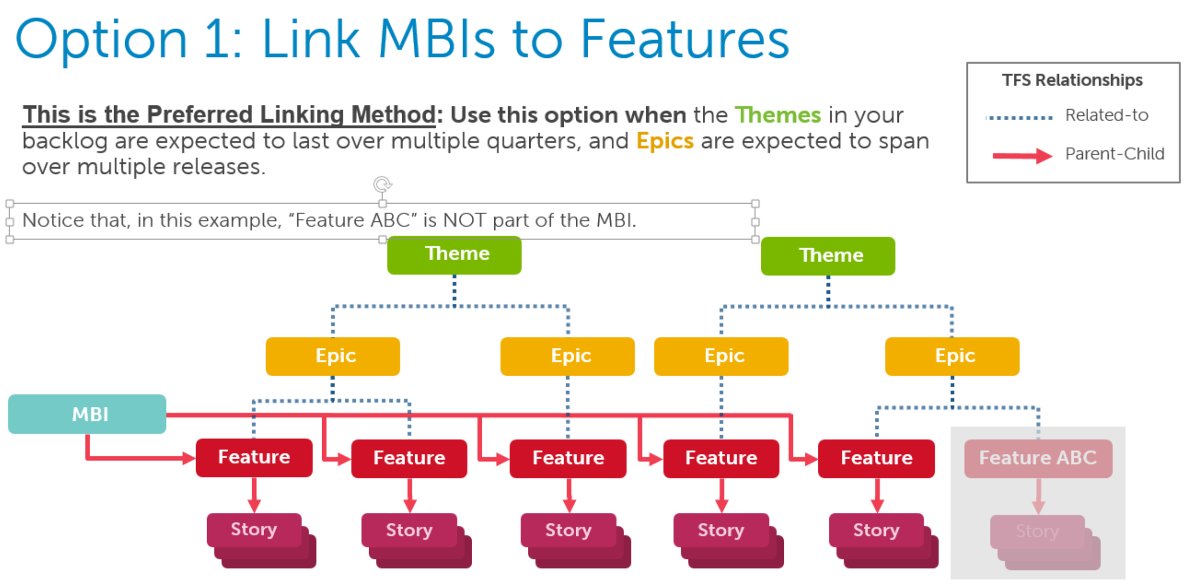
In TFS:

- The MBI work item is available at the "Program" level, in the "Program Backlog". In the TFS2 instance, the Program Backlog is labeled "eDellPrograms".
- Themes, Epics, Features, and Stories reside in the "Team Backlog".
- Stories **must** be linked to MBIs either through a Feature, Epic, or Theme.

Changepoint
TFS
Top-Level Project (TLP)
one : many
MBI
In eDellProgram level
• a.k.a. Business Program
• This is where budget is maintained.
Theme | Epic | Feature | Story
Epic | Feature | Story
Feature | Story
In Team's Backlogs
one : one*
one : many
Sub-Project
Release
• This is where time reporting occurs
• Each TLP must have at least 1 sub-project
• Each TLP must have at least 1 release record
* For programs transitioning to MBIs, only one sub-project is desired.

Fig 4.17 Dell's Toolset

Relating MBIs to other Work Items in TFS



Option 1: Link MBIs to Features

This is the Preferred Linking Method: Use this option when the Themes in your backlog are expected to last over multiple quarters, and Epics are expected to span over multiple releases.

Notice that, in this example, "Feature ABC" is NOT part of the MBI.

Fig 4.18 Scope Hierarchy

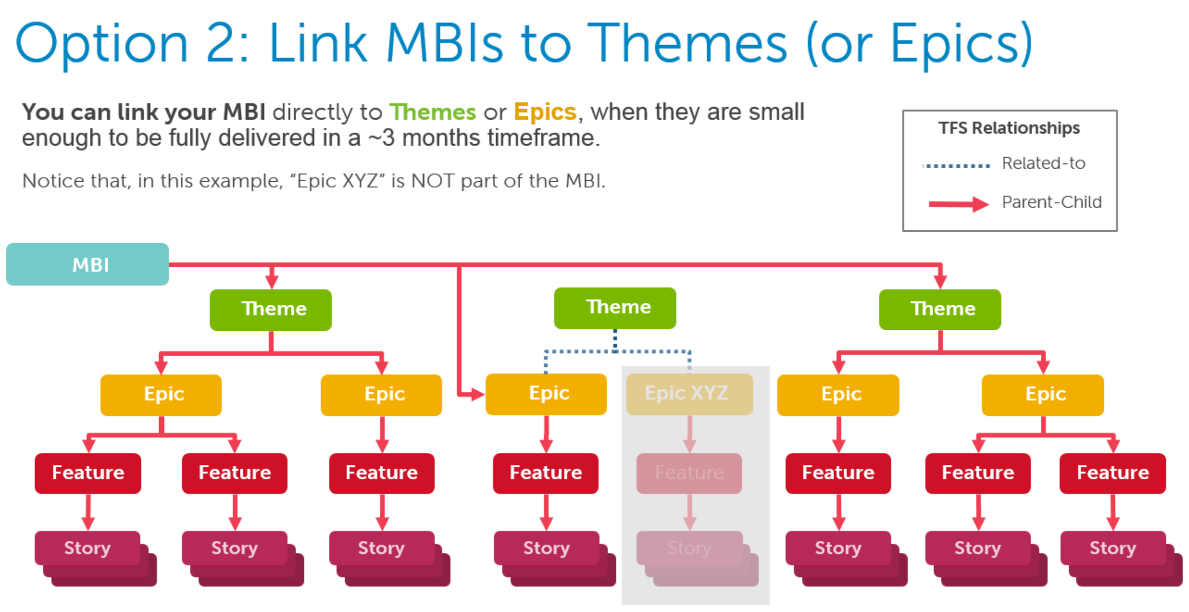

Option 2: Link MBIs to Themes (or Epics)
You can link your MBI directly to Themes or Epics, when they are small enough to be fully delivered in a ~3 months timeframe.
Notice that, in this example, "Epic XYZ" is NOT part of the MBI.
TFS Relationships
Related-to
Parent-Child
MBI
Theme
Theme
Theme
Epic
Epic
Epic
Epic XYZ
Epic
Epic
Feature
Story

Fig 4.19 Linking MBI's to Themes/EPICS

In summary Chapter Four, section 4.4 Dell's Agile Adoption presents an in-depth analysis of Dell Technologies Agile Adoption and Dell's Agile-At-Scale IT Delivery Framework. The research illustrates the details behind the Scrum Workflow and describes the Project Teams associated roles and responsibilities, the artefacts and the Scrum/Sprint Planning procedures. It reveals the concept of the MBI (Minimum Business Incremental Business Value) and how this is used to form and sequence the Product backlog.

Chapter 5 Conclusions

5.0 Introduction

The purpose of this chapter is to outline the conclusions relating to the research objective of this thesis, to present the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile Methodologies for an ERP Program.

This thesis's contribution to research and practice uniquely illustrates, through the lens of the Agile ERP Quadrant view (Fig 2.18), a new understanding of the challenges, benefits, risks, limitations and lessons learned of Adopting Agile for an ERP Program. (Fig 4.1) builds on the Agile ERP Quadrant and denotes the positive, negative and neutral impacts by magnitude.

This research is limited by the lack of detailed case studies outlining the adoption of Agile for a large scale, multiphase ERP Program.

5.1 Conclusions

The Literature Review provides a definition and investigation of ERP Programs. It illustrates the challenges of implementing an ERP Program and the reasons for ERP failures as well as the best practices to avoid such failures. It looks specifically at the Project Management of ERP Programs and looks at the associated Project Management knowledge areas. It reveals a detailed analysis of the Software Development Methodologies and provides a comparison across Waterfall and the many evolving Agile Methodologies, including the Scaled Agile Framework (SAFe).
The Literature Review provides an introduction to the concept of ERP systems, the distinction between ERP systems and other IT systems is very relevant to this thesis as the interdependencies across ERP modules and interlocking systems plays a major part in the understanding of the benefits, risks, challenges and limitations of adopting Agile-At-Scale Methodologies for an ERP Program.
The Literature Review illustrates the challenges and risks associated with ERP Implementations and reveals that the high degree of complexity and change in ERP Programs requires an effective Project Management Methodology. It describes the role of Project Management in ERP implementations and emphasizes its impact across the Project Management knowledge areas. It concludes by revealing how it greatly improves the odds of an ERP implementations success. It goes into depth on the Software Development Methodologies, it compares and contrasts Waterfall and the various Agile Methodologies and asks which is best suited to an ERP Program. The Literary Review concludes with a detailed breakdown of the benefits, risks, challenges and limitations or adopting Agile-At-Scale for an ERP Program.

Chapter Three presents the Participant Observer Research in the form of a comprehensive survey across Dell’s IT function. The survey reveals Dell’s Agile experiences, the areas where Agile is working well and the areas that need improvement. It presents the detailed findings of an extensive Dell IT survey, specifically on Dell’s Agile adoption experiences. It describes the key highlights, opportunities and takeaways.

Chapter Three concludes that while adopting Agile allows for better traceability and the ability to deliver key requirement to the business partners, it requires significant investment in training and business alignment and introduces challenges across Release Management, Environment Management and Roadmap planning.

Chapter Four describes and interprets the findings of the Dell Survey and looks specifically at Dell's experience in a large multiphase ERP Program. It focuses on the research objectives of the Project Management Benefits, Risks, Challenges and Limitations of adopting Agile-At-Scale Methodologies for an ERP Program and illustrates that through the lens of the Agile ERP Quadrant.

Chapter Four breaks down the challenges, benefits, risks, limitations and lessons learned of Adopting Agile for an ERP Program. It provides concrete examples of Dell's experience across these categories and ties back the findings to the company wide findings outlined in Chapter Three. Chapter Four concludes that the adoption of Agile is not a one size fits all and highlights specific areas associated with ERP Programs that require a more hybrid approach.

Chapter Four presents an in-depth analysis of Dell Technologies Agile Adoption and Dell's Agile-At-Scale IT Delivery Framework. The research illustrates the details behind the Scrum Workflow and describes the Project Teams associated roles and responsibilities, the artefacts and the Scrum/Sprint Planning procedures. It reveals the concept of the MBI (Minimum Business Incremental Business Value) and how this is used to form and sequence the Product backlog.

In summary, this research presents the Project Management challenges and causes of failure for ERP Programs. It provides an in-depth analysis of the Software Development Methodologies and provides a detailed study of Dell Technologies Agile-At-Scale journey and outlines Dell's Agile Framework which can be adopted for a large-scale Global ERP Implementation

This thesis concludes that Agile-At-Scale methodologies for an ERP Program has many benefits, particularly around business partner alignment and engagement. It also provides for much needed flexibility to change course as needed and to ensure delivery of a quality product.

Agile adoption also has significant challenges and risks. Moving from a Waterfall Methodology to being Agile and maturing to Agile-At-Scale requires a very significant investment and alignment across release, environment management as well as funding models.

What is clear from the research is that adopting Agile-At-Scale for an ERP Program is a journey. As the Dell case study demonstrates, evolving from a Waterfall methodology for ERP to a full scale Agile-At-Scale approach needs to be tackled firstly in a hybrid Agile approach. Whereby, like Dell, the organization are Agile through the Development cycle and revert to Waterfall during the end to end testing and user acceptance testing cycles. As the organization evolves the next step is to become more Agile in the testing cycles. However, the key challenge remains, given the nature of ERP Programs, it's an "all or nothing" deployment approach, so in the case of a green field ERP deployment, the Agile concept of continuous delivery to a Production (live) environment cannot be realized.

Based on my research findings, I would recommend Agile techniques for custom development projects with fixed scope and fluid cost or schedule, but not in its truest form for a standard ERP implementation because ERP projects generally require a distinct and collective development and deployment of product features that need to be delivered simultaneously to achieve the business objectives. Agile may be useful for delivering incremental custom code but usually doesn't allow for the necessary planning, baselining, and managing of scope, cost, and schedule that is required of a typical ERP project. A hybrid Agile approach through the development and testing cycles clearly has many benefits.

Looking to the future of Project Management beyond Waterfall and Agile, Dell Technologies acquired Pivotal in recent years. The Pivotal Cloud Foundry (PCF) is fast becoming the proven solution for companies seeking software-led, digital transformation. Pivotal Labs provide software development consultancy and it is revolutionizing IT Development and by association IT Project Management. Pivotal want software engineering to become the value-engine of the organization. Pivotal's aim is for the end-to-end product development process to be a core competency of the organization, driven by empowered, autonomous and self-organizing agile teams and guided by principles of user-centric design, lean start-up methodologies, and lean engineering practices. Their mission is to "transform the way the world builds software". Dell is currently at the early stage of adopting this approach in what is referred to as the "Path to Pivotal"

On a personal level, the evolution from Waterfall to Agile and now to Pivotal highlights the changing roles within IT Project Management and further muddies the waters between IT Product Managers and IT Project Managers. The traditional IT Project Manager's role was focusing primarily on the execution side. They take the product vision from the Product Manager, develop a project timeline around it, and plan the work for the development team to hit important goals and deadlines. Or, to put it simply, their responsibility is to successfully bring a project to completion within the agreed budget, time, and quality.
A Product Manager's role is strategic, much like a CEO but for the product. They're the ones who set and own the overall product direction, staying with it until the product is removed from the market. It is their responsibility to understand the user needs, translate them into a design or MVP (Minimum Viable Product), and lead a development team to build the product and meet those needs. Product Managers deal with the What? And Why? Project Managers to How? and When?
This changing roles within IT Project Management and cross over with the role of the IT Product Manager is a clear opportunity for further research.

From my experience, Agile is not only the present but the future of IT Project Management and while it is may not be fully conducive to an ERP environment, ERP Systems are quickly evolving to becoming more cloud based which will allow for the Agile model to be a better fit.

References

1. Introduction

[1] Liaquat, H., Jon, D. P., & Rashid, A. M. (2002). Enterprise Resource Planning: Global Opportunities & Challenges. ISBN: 193070836x Idea Group Publishing.

[2] Panorama Consulting Solutions (2016*). Panorama's 2016 ERP report | panorama consulting solutions.* Available at: http://panorama-consulting.com/resource-center/2016-erp-report/ (Accessed: 14 September 2016).

[3] Nordin, N., & Adegoke, O. (2015). Learning from ERP implementation: A case study of issues and challenges in technology management. Jurnal Teknologi, 74(1).

[4] Neal, H. (2010). *ERP implementation strategies – A guide to ERP implementation methodology.* Available at: http://blog.softwareadvice.com/articles/manufacturing/erp-implementation-strategies-1031101/ (Accessed: 14 September 2016).

[5] Fetouh, A. A., el Abbassy, A., & Moawad, R. (2011). Applying Agile Approach in ERP Implementation. IJCSNS, 11(8), 173.

[6] Earl, Michael, Jeffery Sampler, and James Short, (1995) "Strategies for Reengineering: Different
ways of Initiating and Implementing Business Process Change," Centre for Research in Information
Management, London Business School,

2.1 ERP

[1] Swartz, D., & Orgill, K. (2001). Higher education ERP: Lessons learned. *Educause Quarterly, 24*(2), 20-27.

[2] Nah, F. F., Lau, J. L., & Kuang, J. (2001). Critical factors for successful implementation of enterprise systems. *Business Process Management Journal, 7*(3), 285-296.
[ ] Beheshti, H. M. (2006). What managers should know about ERP/ERP II. *Management Research News, 29*(4), 184-193

[3] Harrison, J. L. (2004). *Motivations for enterprise resource planning (ERP) system implementation in public versus private sector organizations.* (Ed.D., University of Central Florida). *ProQuest Dissertations and Theses,* . (305080817).

[4] Lieber, R. B. (1995). Here comes SAP. *Fortune, no.October, 2*, 122-124.

[5] Siau, K. (2004). Enterprise resource planning (ERP) implementation methodologies. *Journal of Database Management, 15*(1), i-vi.

[6] Parr, A., & Shanks, G. (2000). A taxonomy of ERP implementation approaches. *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on,* 10 pp. vol. 1.
Nash, K. S. (2000). Companies don't learn from previous IT snafus. *Computerworld, 16*(21), 32-33.

[7] Dover, C. (2012). *Worldwide enterprise resource management applications 2012–2016 forecast and 2011 vendor shares* . (MARKET ANALYSIS No. 238476, Volume: 1).IDC.

[15] https://www.alliedmarketresearch.com/ERP-market

[16] Carton F, Adam F and Sammon D. (2007) "Project management: a case study of a successful ERP implementation"
Business Information Systems, University College Cork, Cork, Ireland

[17] "Stakeholder Power Analysis" – IIED – International Institute for Environment and Development

2.2 Challenges of ERP implementations

[8] Dillard, J. F., & Yuthas, K. (2006). Enterprise resource planning systems and communicative action. *Critical Perspectives on Accounting, 17*(2), 202-223.

[9] Ehie, I. C., & Madsen, M. (2005). Identifying critical issues in enterprise resource planning (ERP) implementation. *Computers in Industry, 56*(6), 545-557. ;
Helo, P., Anussornnitisarn, P., & Phusavat, K. (2008). Expectation and reality in ERP implementation: Consultant and solution provider perspective. *Industrial Management & Data Systems, 108*(8), 1045-1059.

[10] Scott, J. E., & Vessey, I. (2000). Implementing enterprise resource planning systems: The role of learning from failure. *Information Systems Frontiers, 2*(2), 213-232.:
Helo, P., Anussornnitisarn, P., & Phusavat, K. (2008). Expectation and reality in ERP implementation: Consultant and solution provider perspective. *Industrial Management & Data Systems, 108*(8), 1045-1059.
Maditinos, D., Chatzoudes, D., & Tsairidis, C. (2012). Factors affecting ERP system implementation effectiveness. *Journal of Enterprise Information Management, 25*(1), 60-78.

[11] Zornada, L., & Velkavrh, T. B. (2005). Implementing ERP systems in higher education institutions. *Information Technology Interfaces, 2005. 27th International Conference on,* 307-313.

[12] Huang, S., Chang, I., Li, S., & Lin, M. (2004). Assessing risk in ERP projects: Identify and prioritize the factors. *Industrial Management & Data Systems, 104*(8), 681-688.

[13] Lamers, M. (2002). Do you manage a project, or what? A reply to "Do you manage work, deliverables or resources",< i> international journal of project Management</i>, april 2000. *International Journal of Project Management, 20*(4), 325-329.

2.3 Project Management of ERP Implementations

[14] Charlie C. Chen, Chuck C. H. Law, and Samuel C. Yang,*(* FEBRUARY 2009)
Managing ERP Implementation Failure: A Project Management Perspective
IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, VOL. 56, NO. 1,

2.4 ERP Failures

[1]. http://www.softwareadvice.com/erp/software-pricing/

[2]. http://blogs.wsj.com/cio/2013/12/11/avons-failed-sap-implementation-reflects-rise-of-usability/

[3].http://www.pcworld.com/article/253507/epicor_customer_lawsuit_may_show_danger_of_ going_solo_on_erp_software_project.html

[4.]https://www.reddit.com/r/talesfromtechsupport/comments/30mmog/new_erp_system_fast_cheap_good_pick_none_of_three/

[5] Goeun S. (2013) "Challenges in Implementing Enterprise Resource Planning (ERP) System in Large Organizations:
Similarities and Differences Between Corporate and University Environment" –

[6] Barker T and Frolick M. N.(2003). ERP IMPLEMENTATION FAILURE: A CASE STUDY" Information Systems Management

[7] Somers T M and Nelson K (2001) "The Impact of Critical Success Factors across the Stages of ERP Implementations"

2.5 Software Development Methodologies

[1] Ramel D, “Survey: Agile Not Catching Waterfall Just Yet.” [Online]. Available:http://adtmag.com/articles/2010/11/08/agile-not-overtakingwaterfall.

[2] Jacobson, I “Software Development a la Mode.” [Online]. Available: http://blog.ivarjacobson.com/software -development -a-la-mode/.

[3] Highsmith J and Cockburn A, (2000.).*Agile Software Development*. The Agile Software Development Series,

[4] Rally Software, (2009).*The Agile Impact Report: Proven Performance Metrics from the Agile Enterprise*.

[5] “Agile Manifesto.” [Online]. Available: http://agilemanifesto.org/.

[6] Dybå T and Dingsøyr T, (2008) “Empirical Studies of Agile Software Development: A Systematic Review,” *Information and Software Technology*, vol. 50, no. 9-10, pp. 833-859,.

[7] Beck K, (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, MA,

[8] Schwaber K. and Beedle M, (2002) *Agile Software Development with Scrum*. Prentice-Hall, Upper Saddle River, NJ,.

[9] Merriam-Webster, “Merriam-Webster Online.” [Online]. Available: http://www.merriam-webster.com/dictionary/.

[10] Schatzki T.R. (2001)., *The Practice Turn in Contemporary Theory*. Routledge Taylor and Francis Group,

[11] Petersen K (2010)., “Implementing Lean and Agile Software Development in Industry,” PhD Dissertation, Blekinge Institute of Technology,

[12] Dieste O, Lopez M, and Ramos F, (2008) “Formalizing a Systematic Review Updating Process,” in *Software Engineering Research, Management and Applications*, , pp. 143-150.

[13] Business Dictionary, “Business Dictionary.” [Online]. Available: http://www.businessdictionary.com/defin (ition/process.html

[14] Dominguez, J 2009 “The CHAOS Report (2009) on IT Project Failure.” Available: http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure.

[15] Aranda,J “Standish, the CHAOS report, and science.” [Online]. Available: http://catenary.wordpress.com/2008/09/24/standish-the-chaos-report-andscience/

[16] Standish Group, "The Standish Group Report CHAOS." [Online]. Available: http://www.projectsmart.co.uk/docs/chaos-report.pdf.
[17] Royce W, (1970) "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings IEEE WESCOM*,.

[18] Rational Software Corporation (1998), "Rational Unified Process: Best Practices for Software Development Teams.".

[19] IABG, "Das V-Modell." [Online]. Available: http://v-modell.iabg.de/.

[20] Boggs R. A , (2004). "The SDLC and Six Sigma: An Essay on Which is Which and Why," *Issues in Information Systems*, vol. 5, no. 1,

[21] Boehm B.W., (1988) "A Spiral Model of Software Development and Enhancement," *Computer*, vol. 21, no. 5, pp. 61-72,.

[22] Cohn M, (201) .*Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley,

[23] Abrahamsson P Salo O, Ronkainen J, and Warsta J (2002)., *Agile Software Development Method: Review and Analysis*. VTT Electronics,

[24] Sutherland J and Schwaber K, (2007) "The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process." PatientKeeper, Inc,.

[25] .(2004) *Agile Project Management with Scrum*. Microsoft Press,.

[26] Stapleton J, (1997) .*DSDM Dynamic Systems Development Method: The Method in Practice*. Addison-Wesley,

[27] DSDM Consortium, "DSDM Atern; the Agile Project Delivery Framework." [Online]. Available: http://www.dsdm.org.

[28] Palmer S R and Felsing M , (2002) *A Practical Guide to Feature-Driven Development*. Prentice Hall,.

[29] Cockburn A, (2001) .*Crystal Clear: A Human-Powered Software Development Methodology for Small Teams*. Addison-Wesley,

[30] Poppendieck M ,and Poppendieck T (2003)., *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional,

[31] Poppendieck M, (2007). "Lean Software Development," in *ICSE COMPANION '07: Companion to the Proceedings of the 29th International Conference on Software Engineering*,

[32] Petersen K,(2010) "Is Lean Agile and Agile Lean? A Comparison Between Two Software Development Paradigms," *:Modern Software*

Engineering Concepts and Practices: Advanced Approaches; Ali Dogru and Veli Bicer (Eds.), IGI Global,.

[33] Hibbs C, Jewett S, and Sullivan M (2009)., *Art of Lean Software Development.* O Reilly Media, Inc.,

[34] Jalali S and Wohlin C. (2010) "Agile Practices in Global Software Engineering - A Systematic Map," in *5th IEEE International Conference on Global Software Engineering*, , pp. 45-54.

[35] Little T, Greene F, Phillips T Pilger R, and Poldervaart R (2004), "Adaptive Agility," in *Agile Development Conference (ADC '04)*, , pp. 63-70.

[36] Esfahani H and Yu, E (2010) "A Repository of Agile Method Fragments," in *New Modeling Concepts for Today's Software Processes*, vol. 6195, J. Münch, Y. Yang, and W. Schäfer, Eds. Springer Berlin / Heidelberg, , pp. 163-174.

[37] Qumer A and Henderson-Sellers B, (2008). "A Framework to Support the Evaluation, Adoption and Improvement of Agile Methods in Practice," *Journal of Systems and Software*, vol. 81, no. 11, pp. 1899-1919,

[38] Shashank S.P.and Darse D.H.P, (2011). "Finding Common Denominators for Agile Practices: A Systematic Literature Review," Master Thesis, Blekinge Institute of Technology,

[39] Couper M.P., (2000) N "Web Surveys: A Review of Issues and Approaches," *The Public Opinion Quarterly*, vol. 64, no. 4, pp. 464-494,.

[40] Lotz M July 5, (2013) http://www.seguetech.com/waterfall-vs-agile-methodology/

"Waterfall vs. Agile: Which is the Right Development Methodology for an ERP Project?"

[41] Leffingwell D, SAFe Creator, and CEO, Scaled Agile Inc. http://www.scaledagileframework.com/

[42] Waters K.(17 March 2007) | *http://www.allaboutagile.com10 Key Principles of Agile Development*

3.1 Research Methodology

[1] Kaplan and Maxwell (1994) "Combining quantitative and qualitative methods in information systems research", MIS Quarterly, 12(4), pp. 571-586

[2] Yin, 1994), "Case Study Research, Design and Methods", Sage Publications, Beverly Hills, California.

[3] Clark, 1972,

[4] Elden and Chisholm, 1993

[5] Carr and Kemmis, 1986.

[6] Orlikowski and Baroudi (1991, p.5) "Studying information technology in organizations: research approaches and assumptions", Information Systems Research, 2(1), pp. 1-28

[7] Boland, 1985

[8] Walsham 1993, p. 4-5

[9] Blum, 1952 "Getting individuals to give information to the outsider", Journal of Social Issues, 8, pp. 35-42

[10] Quinn Patton Michael (1986)

[11] Bronisław Malinowski

[12] Miles & Huberman, 1984

[13] Darke et al., 1998

[14] Isenhardt, 1989; "Building theories from case study research", Academy of Management Review, 14, pp.532-550

[15] Maimbo & Pervan, 2005

[16] Bodgan and Biklen, 1982

[17] Marshall and Rosman 1999

[18] Faulkner 1982

[19] Becker and geer 1982

[20] Benbasat et al 1987