# CONTROLLING SCHEDULE DURATION DURING SOFTWARE PROJECT EXECUTION

ZANA AHMEDSHAREEF

A thesis submitted in partial fulfilment of the requirements of the University of Brighton for the degree of Doctor of Philosophy

July 2015

# Abstract

This thesis describes a method of identifying the influences on schedule delays in projects that develop large software systems. Controlling schedule duration is a fundamental aspect of managing projects because of the financial losses associated with late projects. While challenges with controlling software projects have been investigated, there still seemed to be more to be learned about the interplay of a range of factors during project execution and that affect project duration when developing and integrating software systems within enterprise architecture environment.

This research investigated the activities involved in controlling schedule duration in projects at a global software company that developed large software systems for its client, using globally distributed teams located in the United Kingdom and India. The projects used a version of the Iterative and Incremental Development method within a Component-based model. The empirical data used in the research came from past project performance reports, comprising both text and numeric data, created by and for project participants, purely for internal use.

The research adopted a mixed method approach to enquiry, employing a method which had two distinct but integrated research phases. A quantitative analysis of numeric data collected from three projects identified the points where schedule delays occurred in the constituent phases of the projects. These phases which most contributed to project delay were then subject to greater in-depth examination. Qualitative analysis of textual data collected from the six project phases contributing most to project delay identified relevant categories of project phenomena using Grounded theory techniques. These were used to develop explanatory models inspired by Actor-network theory of the interaction among project actors during project execution.

The key research findings included firstly uncovering of a greater degree of interaction and interdependency among the project actors, which illuminated more complexity besetting the development process, than expected. Secondly, the actual project phases and activities were less self-contained and more susceptible to interactions with their

surrounding context. The project was influenced by emergent constraints that were not explicit in the project plans; such as the need for project activities to wait for services provided by other project actors (often, external parties to the project), the existence of resource clashes and conflicts in priorities, and competing service requests from a range of different projects. Thirdly, there was an assumption that the project control system had a project manager with direct control over the project resources/actors whose contribution was essential to the successful completion of the project. In this environment, day-to-day control was in fact devolved to phase managers who depended on actors outside their control such as those involved in other phases of the project, or parties outside the project with concerns relating to the project, such as those concerned with the overall architectural integrity of the broader programme of work within which these projects were located.

# Contents

# List of tables

# List of figures

# List of equations

# List of abbreviations

The following abbreviations were used repeatedly in this thesis. Other abbreviations used rarely are not listed here; they exist in the text and are explained where they are first introduced.

| | |
|---|---|
| ABC | The company that provided research data |
| AD | Actual duration |
| AN | Activity network |
| ANT | Actor-network theory |
| APM | Association for Project Management |
| AT | Assembly Test |
| BAC | Budget at completion |
| BM | Build manager |
| CASE | Computer aided software engineering |
| CBD | Component-based development |
| CCB | Change control board |
| CCM | Critical chain method |
| CLT | Client |
| CONS | Consumer |
| CPDM | Cross project deliver managers |
| CPM | Critical path method |
| CR | Change request |
| DBT | Design, Build, and Test |
| DM | Design manager |
| DSDM | Dynamic system development model |
| DV | Duration variance |
| ES | Earned schedule |
| ESB | Enterprise service bus |
| ESBM | Enterprise service bus manager |
| ESER | Empirical software engineering research |
| EV | Earned value |

| | |
|---|---|
| EVA | Earned value analysis |
| FD | Functional design |
| FD/TD Transition | Functional design to Technical design transition |
| FF | Finish-to-finish |
| FS | Finish-to-start |
| FV | Finish variance |
| GSD | Global software development |
| GT | Grounded theory |
| GTM | Grounded theory method |
| HLD | High level design |
| IBM | International business machine company |
| IID | Iterative and incremental development |
| IT | Integration Test |
| LSD | Lean software development |
| LSS | Large software systems |
| MMR | Mixed method research |
| MRE | Magnitude of relative error |
| OPP | Obligatory passage point |
| PC | Percentage complete |
| PD | Product development |
| PhD | Doctor of philosophy |
| PLD | Product line development |
| PM | Project manager |
| PMI | Project management institute |
| PS | Peer supplier |
| PSA | Phase schedule accuracy |
| PSC | Phase schedule change |
| PSD | Phase schedule delay |
| PV | Planned value |
| QUAL | Qualitative |
| QUAN | Quantitative |
| RAG | Red-amber-green |
| RQ | Research question |

| | |
|---|---|
| SAM | Solution architecture manager |
| SD | Scheduled duration |
| SF | Start-to-finish |
| SOA | Service oriented architecture |
| SoS | System of systems |
| SPI | Schedule performance index |
| SS | Start-to-start |
| STV | Start variance |
| SV | Schedule variance |
| TAM | Technical architecture manager |
| TD | Technical design |
| TDM | Test data manager |
| TEM | Technical environment manager |
| TM | Test manager |
| TS | Tester |
| UK | United Kingdom |
| UT | Unit Test |

# Acknowledgements

I am indebted to my supervisors Doctor Bob Hughes - for his sustained effort, valuable recommendations, and intellectual challenge; and Professor Miltos Petridis - for his continuous support and invaluable guidance throughout the duration of my research.

I would like to express my sincere thanks to two of the ABC Directors (who for reasons of confidentiality must unfortunately remain anonymous) for making the project data available to the research. Their eagerness to improve the work practices in ABC has contributed to enriching this research.

I am very grateful to Doctor Roslyn Cameron for her numerous and invaluable guidance on mixed methods research and methods of project management research.

I also would like to thank the Doctoral College staff at the University of Brighton for their continuous support throughout my research.

I would like to express my immense gratitude to my dear wife Gona who always gave me time and support despite the heavy work burdens and family demands she was under herself throughout the duration of my research.

Thanks to my son Darya for his understanding of the pressure I was under and giving me the space and time to carry on with the works of this thesis; to my daughter Dlara, thanks for being my companion during the final two years of my research; and to the newly arrived, my daughter Lanya, for bringing us joy at the final months of completing this dissertation.

I am indebted to my highly regarded father (Ahmed Sharif) and dear mother (Suhaila Abdulkarim) for being the source of inspiration, encouragement, and making life easier for me.

# Declaration

*I declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the original work of the author. The thesis has not been previously submitted to this or any other university for a degree, and does not incorporate any material already submitted for a degree.*

Zana Ahmedshareef

July 2015

*Dedicated to…*

*my father Ahmad Sharif & mother Suhaila Abdulkarim,*

*my beloved wife Gona,*

*and my dear children Darya, Dlara, and Lanya*

# 1 Introduction

## 1.1 Statement of the problem

In an increasingly competitive business environment, controlling schedule duration in software projects is crucial in ‘time and materials’ contracts, since schedule delay by the producer increases the cost incurred by the acquirer. The difficulty of such control (Ebert, 2007; Deephouse et al. 1996) increases when developing and integrating large software systems (Patanakul, 2014) within an enterprise architecture environment (Petersen et al. 2014), and through globally distributed teams (Moløkken-Østvold & Jørgensen, 2005; Damian & Lanubile, 2004; Herbsleb & Moitra, 2001). Such projects have an increased risk of failure (Verner et al. 2014; Verner et al. 2012a, 2012b).

The challenges of managing software projects have been investigated by many. It will be seen in this thesis that existing studies can be differentiated by the degree to which they (i) focus on separate areas (topics) of project management in isolation rather than holistically (ii) address the external rather than internal factors influencing project success (iii) are empirically based (iv) base their findings on quantitative rather than qualitative analysis. For example, recent work investigating causes of software project failure (Lehtinen et al. 2014) identifies external factors but do not look at the internal factors within the projects that show how problems evolve from one week to the next during the project. A literature review by McLeod & MacDonell (2011) of research over a 10 year period (1996-2006) indicates focus of empirical studies on individual, rather than interrelationship and interaction among, factors influencing project outcome. Previous work which did focus on examining project schedule behaviour (Rainer, 1999) produced very useful insights, but concluded that it was unable to determine definite causes that explain schedule delay (page: 150). Therefore, there seems still more to be learned about the internal factors that influence schedule duration leading to schedule delay.

A focus on controlling schedule duration during project execution should recognise that the problems are more complex than can be explained by studying schedule behaviour alone, as project managers are confronted with interdependent problems and dynamic situations that interact with each other (Ackoff 1979, page: 99), and projects in practice can operate in a complex, uncertain and unstable environment (Schön, 1983, page: 18). The practice of software project management is no different (Kitchenham, 1987). In such environments, delivering software projects where meeting deadlines is critical meet with particular difficulties. Hence, the research and practice bodies of knowledge need to match that complexity if they are to provide practical solutions to the challenges facing such projects. One way to overcome the perceived ‘split between industry practice and academic research’ (Jacobson et al. 2012), is to develop knowledge of use to practice (Basili et al. 1986; Sjøberg et al. 2007; Basili et al. 2006).

Boehm (1981 page: 607) said long ago that ‘Often, software projects have had effective macro-level planning and control capabilities, but have come to grief because of a lack of visibility into project dynamics at the micro level’. Boehm also noted that ‘Software project status assessment and control would be a fairly straightforward process if everything on the project progressed according to plan. However, particularly on large projects, a great many deviations from the original plan are all happening at the same time’ (*ibid*, page: 612). Furthermore, others commented ‘A major defect in much of the research to date has been its inability to integrate our knowledge of the micro components such as project management, programming, and testing for driving implications about the behavior of the organization in which the micro components are embedded’ (Abdel-Hamid & Madnick 1991, page: 7).

Therefore, the problem can be summarised as:

*The need to have better understanding of the project behaviours that influence software project progress; in particular, the interactions that emerge among project actors during project execution and the way they influence controlling schedule duration and leading to schedule delay; when developing and integrating large software systems within enterprise architecture environment through globally distributed teams. For this understanding to be useful, however, it needs to draw from empirical data utilising both the quantitative and qualitative aspects of the domain whilst investigating the interdependent areas of the project.*

It was within this context that the research about to be described was undertaken.

## 1.2 Scope of the investigation

Controlling software project schedule does not happen in isolation, but within a complex environment where many elements interact and influence one another while the project progresses through its lifecycle. Figure 1.1, based on materials from APM (2006); Cadle & Yeates (2008); PMI (2008); Chrissis et al. (2011), shows the project control element within the overall project undertaking.

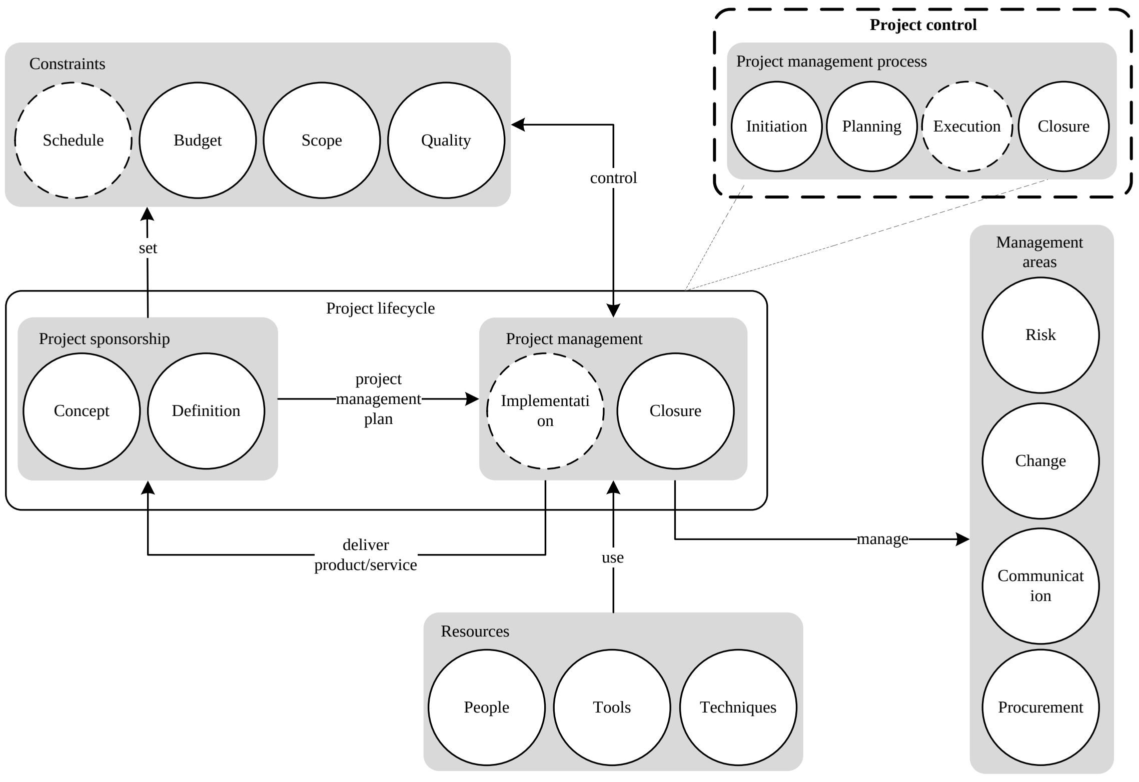


**Figure 1.1 Project elements**

During the project lifecycle, the project sponsors set constraints (schedule, budget, scope, quality) for the project to operate within; the project management attempts to control these constraints while using project resources (people, tools, and techniques) to deliver project deliverables. In so doing, the project management follows a process (initiation, planning, execution, closure) employing project control mechanisms, while simultaneously managing other areas of the project (risk, change, communication, and procurement).

This investigation focuses on controlling schedule during project execution which takes place at the implementation stage of the project lifecycle (see the elements marked with dotted line in Figure 1.1). Thus, while carrying out the activities (execution) implementing the project management plan, the project manager attempts to control project. The project schedule encompasses estimate of the project duration and the times when activities and events are planned to occur, based on the logical dependencies among the activities and estimated duration of each activity.

Given this context, the questions to which answers were sought included:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

In this research, the term ‘mechanisms’ include the creation of progress reports and holding progress meetings; and the term ‘project actors’ refers equally to the human and nonhuman elements (constituent parts) of the project.

To answer the research questions, the literature related to controlling software projects was examined. However, existing studies did not seem to provide a complete explanation of the empirical data from a technology consulting firm, named ABC for anonymity, which competes on the global software development services industry. This justified a closer investigation of ABC projects. The research analysed the schedule performance reports - which were created by and for project participants - of the Design, Build, and Test stages of three completed projects. This revealed that the Test phase of these projects was particularly problematic. The study then analysed in more depth six Test phases across the three projects to identify the influencing factors on schedule delay across the six Test phases.

## 1.3 Structure of the thesis

The reminder of the thesis is organised as follows.

Chapter 2 surveys the relevant literature that relates to the research questions. It examines schedule control mechanisms (RQ1), factors influencing project duration (RQ2), and interaction within the development environment (RQ3). The chapter then locates the position of the investigation described in this dissertation within existing work.

Chapter 3 provides the context of the case company (ABC) who provided the empirical data; describing their software delivery model and approach to managing and controlling project execution. The chapter describes the contents of the project performance reports, which were used as source data of this research, and positions the ABC's work processed in the context of current accepted good practice.

Chapter 4 describes the research methodology of this programme of study and presents the research design which consists of four phases, the details of which are presented in the four subsequent chapters in more depth (Quantitative approach, Case selection, Qualitative approach, and Explanation development). Chapter 4 then locates the methodological position of this research within existing work.

Chapter 5 defines the research methods adopted for the quantitative approach, and analyses the numeric data collected from the project performance reports of three projects using schedule-related metrics in order to develop answers to RQ1. One project is used to illustrate the quantitative approach, followed by an analysis of two further projects.

Chapter 6 examines the extent to which the quantitative analysis was able to develop answers to RQ1, concluding that the numeric data of the project performance reports did not support identifying the causes of schedule delay. Therefore, a case study research was designed for in-depth examination of the textual data for selected six cases in order to develop answers to RQ2 and RQ3 in the subsequent two chapters.

Chapter 7 defines the research methods adopted for the qualitative approach, and analyses the textual data collected from the project performance reports of six cases (Test phases) using Grounded theory techniques to develop categories of project phenomena present during project execution, in order to answer RQ2. The analytical approach used in one case is explained in detail to illustrate the qualitative approach, followed by the findings of a further five cases.

Chapter 8 describes the consolidations of explanatory models of schedule delay based on analyses of the interaction among project actors using Actor-network theory concepts in order to develop answers to RQ3. The thesis then reflects on the research journey as a whole.

Chapter 9 summarises the general conclusions of this thesis.

Finally, Appendix A provides the empirical data investigated in this research, where identifiable information has been anonymised for confidentiality, as well as results of the remaining analyses that were not presented in Chapters 7.

## 1.4 Previously published work

During the course of this research, a number of ideas were developed into academic outputs - papers and presentations, which were validated by the academic peer review process and accepted for publication, material from which has been incorporated in this thesis:

Ahmedshareef, Z. (2013a). *An Empirical Examination of the Internal Dynamics of*

*Software Project Management: a Mixed Methods Approach.* Paper presented at the UK Academy for Information Systems, 8th Annual PhD Consortium, University of Oxford, UK.

Ahmedshareef, Z. (2013b). An Empirical Examination of the Internal Dynamics of Software Project Management: a Mixed Methods Approach. *Research Poster Competition 2013*. Brighton, UK: University of Brighton.

Ahmedshareef, Z. (2013c). *The Dynamics of Software Project Management.* Paper presented at the Faculty of Science and Engineering Doctoral College Research Student Conference, Brighton, UK.

Ahmedshareef, Z. (2014). *The Dynamics of Controlling Schedule Duration in Software Projects.* Paper presented at the Faculty of Science and Engineering Doctoral College Research Student Conference, Brighton, UK.

Ahmedshareef, Z., Hughes, R., & Petridis, M. (2014a). *Applying Actor-Network Theory to Software Project Management Research.* Paper presented at the 13th European Conference on Research Methodology for Business and Management Studies, London.

Ahmedshareef, Z., Hughes, R., & Petridis, M. (2014b). Exposing the Influencing Factors on Software Project Delay with Actor-Network Theory. *The Electronic Journal of Business Research Methods, 12*(2), 132-146.

Ahmedshareef, Z., Petridis, M., & Hughes, R. (2013). *Empirical Examination of Internal Dynamics of Software Project Management: Mixed Methods Approach.* Paper presented at the 12th European Conference on Research Methodology for Business and Management Studies, Portugal.

Ahmedshareef, Z., Petridis, M., & Hughes, R. T. (2014). The Affordances of Mixed Method in Software Project Management Research. *International Journal of Multiple Research Approaches, 8*(2), 198–217.

# 2 Controlling software projects

In general, controlling software projects can be seen as the ability to minimise surprises, minimise deviations, and early signalling of these deviations during project execution, which require (i) managing the project so that performance stays at or above some reasonable and accepted standard (ii) making sure that original expectations are not allowed to exceed what's possible for a project performing at that standard (DeMarco 1982, page: 5). Therefore, attempting to maintain a project's progress on schedule, during execution, can be seen as a way of controlling its duration.

## 2.1 Introduction

This chapter examines relevant literature that relates to the research questions (RQs), and organises what was found, under three headings corresponding to the RQs:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

Section 2.2 '*Schedule control mechanisms*' examines relevant literature on the Critical path method, Gantt charts, Earned value analysis, and Progress reports in relation to RQ1.

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

Section 2.3 '*Factors influencing project duration*' examines relevant literature on project failure, schedule delay, large software systems, global software development, and project risk management in relation to RQ2.

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

Section 2.4 ‘*Interaction within the development environment*’ examines relevant literature on complexity of interaction in software projects, and interactions within the primary development models: waterfall, iterative and incremental development, component-based model, lean software development and critical chain method in relation to RQ3.

The chapter concludes with locating the position of this investigation within existing work.

## 2.2 Schedule control mechanisms

This section summarises the literature reviewed related to RQ1:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

A distinction is sometimes drawn between project monitoring and project control with project monitoring referring to collecting project data and measuring progress and project control referring to selecting and taking corrective action/s to bring the project back on schedule - see Figure 2.1, which is based on material from Cadle & Yeates (2008), Hughes & Cotterell (2009), and PMI (2008).

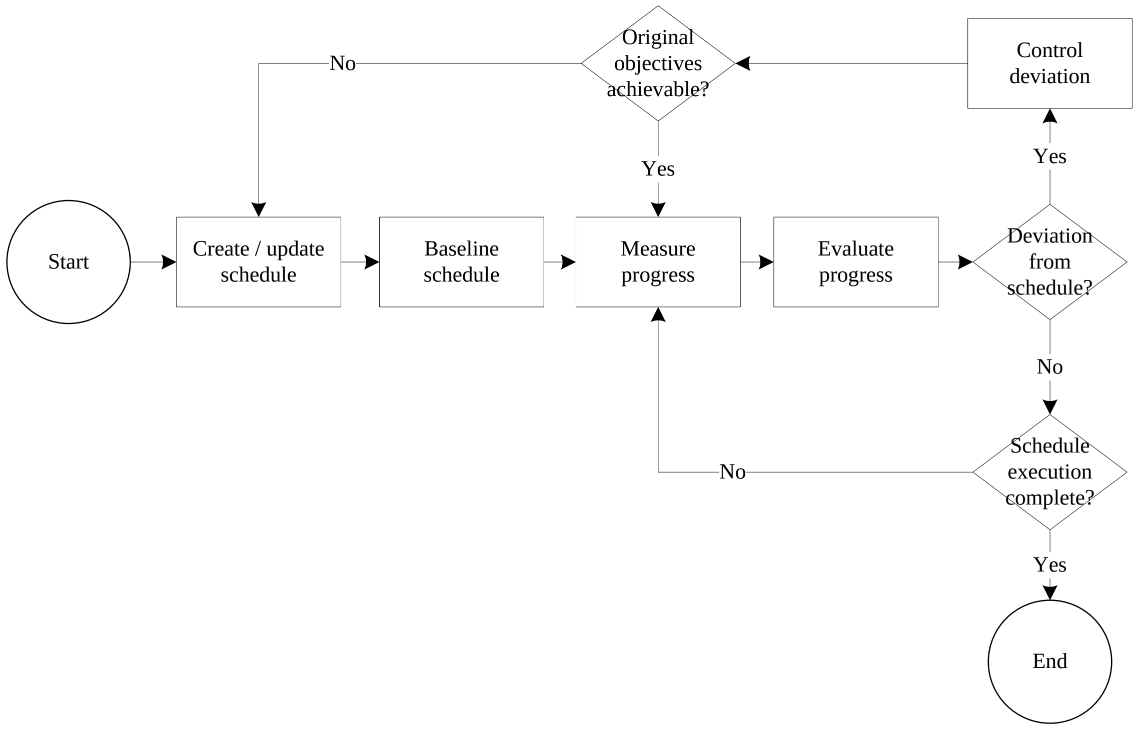


**Figure 2.1 Project monitoring and control process**

A project control cycle needs the creation of a project schedule (Figure 2.1). A project planner would use something like the waterfall model of development phases to help them identify the tasks in a software engineering project. The activity network (AN) could then be created (see section 2.2.1), which could be produced by typing in activity details, estimated durations, and activity dependencies into something like MS Project. A basic Gantt chart (see section 2.2.2) can then be generated and this would be used to allocate resources to activities. Some start dates may have to be delayed because of resource clashes, but this is recorded on the Gantt chart, not the AN. The schedule can then be 'baselined' to allow comparison of the actual progress against the targets on the baseline schedule. A baseline schedule is a version of the project schedule, accepted and approved by the project management team (PMI 2008, page: 159), which is used as a reference point from which performance measurements can be made (Kerzner, 20013, page: 567).

Then, the project manager (PM) may use Earned value analysis (see section 2.2.3) to measure progress (Figure 2.1) where the PM has to collect data from team leaders and convert it into form that higher management can understand. Appropriate measurement of progress can be an indicator of the degree of control on project schedule (DeMarco 1982, page: 3).

Project progress can then be evaluated (Figure 2.1) at progress review meetings where project progress is reported on (see section 2.2.4). Where deviations are identified, corrective actions are considered and the most appropriate actions are implemented to bring progress back on schedule (i.e. controlling schedule duration). These could include: reducing the duration of activities on the critical path, relaxing the precedence rules (Hughes & Cotterell, 2009), overlapping activities (Lockyer & Gordon, 2005), reducing scope or quality, bringing more experienced staff onto critical activities (Lockyer & Gordon, 2005), and using overtime or increasing staff (Cadle & Yeates, 2008). Monitoring progress may continue until project completion.

Exerting control over one aspect of project constraint may affect others. The project triple constraints (iron triangle) of time, cost, scope/quality are well known in the traditional project management literature, where, for example, reducing schedule might increase the cost in employing additional staff (Cadle & Yeates 2008, page: 209). Moreover, controlling project execution may become more difficult for management when the overlapping of activities increases, because the number and direction of interactions among these activities become more complex (Lockyer & Gordon, 2005).

### 2.2.1 Critical path method

Activity networks (AN) enable the identification of activity dependencies by depicting activities as nodes and the dependencies among them as links showing the precedence of some activities over the others (Lockyer & Gordon, 2005; Burke, 2013). The Critical path method (CPM) enables the identification of the activities of an AN that affect the project completion date; this then, allows the project manager to focus on these activities during project execution to ensure their progress is maintained on schedule and to have contingency plans in place should they slip (Kelley & Walker, 1959; Hughes & Cotterell, 2009; Cadle & Yeates, 2008; Lockyer & Gordon, 2005).

The CPM has been criticised for requiring ongoing re-calculation of the project duration due to the changing of the path during project execution, as a result of change in the duration of the non-critical activities, particularly with complex projects that exhibit multiple interactions and overlapping activities (Schonberger, 1981) - though computer software can do this if used. In addition, as the CPM does not take into account resource availability to carry out these activates, the critical path can potentially change during project execution as resource availability fluctuates (Hughes & Cotterell, 2009; Kerzner, 2013).

Whilst it can be argued that the CPM can be used as a schedule control mechanism to bring the deviated critical activities back on schedule, it is more of a planning tool. CPM can show where one activity delay can create delays in other, dependent, activities but was not designed primarily to identify the causes of schedule delay.

### 2.2.2 Gantt chart

The Gantt chart, developed by Henry Gantt in the 1910s, organises project activities on a calendar showing when each activity starts and finishes. It can model the logical dependencies among the activities (like an AN), but also documents management decisions about when staff and other resources will be committed to the project (Lockyer & Gordon, 2005; Cadle & Yeates, 2008; Burke, 2013).

The Gantt chart is seen as a simple method of activity scheduling, easy to develop and update for small or medium size projects, and can show progress status (Hughes & Cotterell, 2009; Boehm, 1981; Burke, 2013; Lockyer & Gordon, 2005). However, the Gantt chart does not readily identify which of the activities might be critical to meeting project end date (though a Gantt chat can automatically be converted to a CPM through tools such as MS project).

Similar to CPM, the Gantt chart is more of a planning tool and was not designed primarily to identify the causes of schedule delay. However, it can be used as a control mechanism as it shows where an activity took longer than planned or its start was delayed (or both), and where there have been resource clashes.

### 2.2.3 Earned value analysis

Earned value analysis (EVA) is a management approach used to monitor project execution. It integrates measures of the scope, schedule and cost for a particular unit of work as a basis for comparing actual progress and cost performance against a baseline plan (Fleming & Koppelman, 2010; Budd & Budd, 2010).

Started as cost/schedule control system criteria (C/SCSC) by the U.S. Department of Defence in the 1960s (Kwak & Anbari, 2011), the EVA measures can be used as early warning signals to assess the performance of the project's schedule and cost against the planned targets during project execution. This information may then be used by management to devise corrective action/s to bring the performance back on track (Vandevoorde & Vanhoucke, 2006; Fleming & Koppelman, 1998). The EVA measurement was created to track cost performance, and although it offers some schedule performance metrics, they calculate schedule performance in cost units (described below).

The EVA has a set of base measures for cost which include: Planned Value (PV), Earned Value (EV), and Budget at Completion (BAC). PV is analogous to an agreed price for a unit of work to be completed, and when the work has actually been completed, the amount of PV is said to have been earned and becomes EV (Fleming & Koppelman, 2010; PMI, 2008; Burke, 2013). Among the different conventions used in assigning EV to completed work is 0/100 where EV is assigned value of 0% until the task is 100% complete (Anbari, 2003; Fleming & Koppelman, 2010) - other variations can be found in Boehm (1981, page: 613) and Kerzner (2013). BAC is the overall estimated project cost, that is, the total PV for the project (PMI, 2008; Fleming & Koppelman, 2010; Budd & Budd, 2010).

Schedule performance indicators, derived from the base measures above, include Schedule variance (SV) and Schedule performance index (SPI) that are relevant to this research. The SV is used to measure the *amount* of work actually done (EV) against the amount that was planned to be done (PV) at a particular point in time (in units of cost), SV = EV – PV. Thus, a positive SV value indicates more work was done compared to the amount planned; a negative SV indicates less work was done compared to the amount planned; and an SV value of 0 means the amount of work done was the same as the amount that was planned to be done.

The SPI is used to assess the state of the project schedule to establish the actual *rate* of progress against the planned rate of progress (in units of cost); i.e. what had actually been achieved against what was planned to have been achieved, by calculating the ratio of EV to PV (SPI = EV divided by PV) for a particular point in time. An SPI value greater than 1 means that the rate of progress is ahead than the rate that was planned; an SPI less than 1 indicates that the rate of progress is behind the rate that was planned; hence an SPI of 1 indicates that the project is progressing at the rate that was planned (PMI, 2008; Chrissis et al. 2011).

For example, if it was planned to do £3,000 worth of work (PV) in the first month of the project, and at the end of the month the actual work done (EV) was worth £1,500; then the SV = 1,500 – 3,000 = – 1,500 which means that the project did not earn as much as it planned to earn; it was behind its first month's target by £1,500 worth of work. The SPI = (1,500/3,000) = 0.5; meaning that the project did not earn at the rate it planned by the end of the moth; the rate of doing actual work was half than the rate that was planned; hence it is progressing slower than was planned.

Other performance indicators include the percentage complete (PC) measure, expressed as a percentage value, is an estimate of the amount of work that has been completed relative to the overall estimated project cost (PMI, 2008), that is EV divided by BAC (Budget at Completion). Using the same example above, if the overall estimated cost of the project was £10,000 then PC = (1,500/10,000) is 15% by end of the first month.

Using the SV and SPI indicators described above to measure schedule performance has been criticised from various respects:

- At the end of the project/phase the SV always becomes 0 and the SPI always becomes 1 (Lipke, 2003; Henderson, 2007): this has two implication (i) it indicates that the amount of work done was according to plan and the rate of progress was according to plan respectively even if these figures were behind plan prior to the project completion (ii) it indicates that the project was completed according to plan even if the project end date was extended; that is, the values of SV and SPI will remain 0 and 1 respectively, as at the time of when the project ended, throughout the extension period of the schedule (Vandevoorde & Vanhoucke, 2006).

- In the case where a project is underperforming during its execution, the SV and SPI measures might accurately reflect the situation during the early and middle stages of its execution, however, they both show performance improvement/recovery towards the end of the project (Lipke, 2003) contrary to what might actually be happening, hence providing an unreliable measure of project performance at the latest stages of the project, which may be the most crucial to meet project end date (Vandevoorde & Vanhoucke, 2006).

- The measures do not distinguish between the impact of critical and non-critical activities on schedule delay; allocating significance to all activities equally (Leach, 1999). For example, it is possible for the EVA measures to be favourable because a lot of non-critical tasks are ahead of schedule while some critical ones are behind and will impact schedule duration. This masks the significance/weight of the critical activities that are deviated from the non-critical ones, and thus the project manager would not know which activity they ought to focus on first to correct deviation from the schedule.

- The measurements do not recognise project phases or key milestones as significant events, as they are based on project level data (Bower, 2007). Bower (2007) proposed Phase earned value analysis (PEVA) instead of EVA. The key difference between EVA and PEVA is that one measures cost at project level and the other duration at phase level.

- Less importantly, the dual meaning of the measures and naming confusion could be sources for misperception. An SV = 0 or SPI = 1 measure taken during the progress of work activities mean that progress is according to plan; however, taken at the end of the phase/project means that the activity is complete (Vandevoorde &

Vanhoucke, 2006). The ‘schedule’ word in the SV and SPI names although imply time measurements; they actually measure cost of the actuals against the targets planned at particular points in time of the project schedule (Vandevoorde & Vanhoucke, 2006; Henderson, 2007).

The above limitations of the SV and SPI indicators illustrate that they are unreliable for identifying the *amount* of schedule delay. An extension metric to EVA has been proposed called ‘Earned schedule’ (Lipke, 2003), which defines the SV and SPI in units of *time* rather than units of *cost* (as with the EVA measures).

The Earned schedule (ES) is the time/duration at which the amount of EV accumulated should have been earned (Henderson, 2006); it attempts to fix the EVA issues by calculating SV and SPI differently, and calls them SV(t) and SPI(t) respectively to differentiate them from the original measures obtained from EVA - (t) standing for time. Thus, SV(t) = ES – AT and SPI(t) = ES / AT; where ES is calculated by determining the time increment of PV in which the EV should have occurred; this can be achieved by tracing the EV curve as a horizontal line at a certain point in time, i.e. at the AT (actual time - the actual time duration from the beginning of the project to the time of assessing progress, for example 7 weeks), backward or forward to the PV curve, and then from this point a vertical line is drawn downwards onto the x-axis (the time) to ascertain the earned portion of the schedule (Lipke, 2003), the ES then is the duration from the beginning of the project until this point in time - see Figure 2.2 (adapted from Vandevoorde & Vanhoucke, 2006).

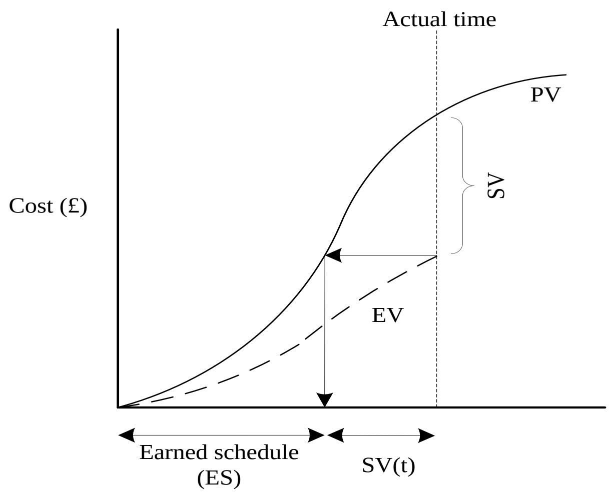


**Figure 2.2 Earned schedule**

A positive SV(t) value or an SPI(t) greater than 1 indicate being ahead of schedule (in time units), whilst a negative SV(t) value or an SPI (t) less than 1 indicate being behind schedule, and an SV(t) value of 0 or an SPI(t) value of 1 indicate being on schedule. The above indicates that the ES approach may have overcome the shortcomings of EVA through measuring the SV and SPI in units of time rather than cost (Vandevoorde & Vanhoucke, 2006; Lipke at al. 2009; Henderson 2003, 2006; Kwak & Anbari, 2011). However, ES have criticised for being 'unnecessarily complex' (Bower & Finegan 2009, page: 441). Bower (2007), in his PhD thesis, criticises the ES for exhibiting similar shortcomings to the EVA measure, for example in the ES's lack of consideration of critical path activities, and unreliability in calculating the ES for the month partially completed when the EV and PV are measured monthly (pages: 64-68). Therefore, it would appear that further empirical research to test the strength and practical implications of the ES approach is needed.

Another metric used to measure time deviation is Finish variance (FV), which is the amount of time between the baseline (i.e. the original planned) finish date of an activity and its current finish date (Corporation, 2012). (FV = Finish - Baseline Finish). A negative FV indicates that the task finished earlier than planned; a positive FV indicates a late task (but not that more effort was spent, as the activity may have started late); an FV of zero means that the task finished exactly when planned. Other metrics can be found in Anbari (2003) and Hughes & Cotterell (2009).

### 2.2.4 Progress report

Project performance can be evaluated at progress review meetings using the information derived in the previous section. Progress reporting is one of the mechanisms used to control schedule duration, where the position of the project can be quantified (e.g. through EVA) and textual explanations may be provided (e.g. through RAG status) to support the numeric measures (Burke, 2013).

The Red/Amber/Green (RAG) status report, sometimes called traffic-light, reporting is used to ascertain the likelihood of meeting the planned target dates. A Green flag indicates that work progress 'is according to plan', an Amber flag indicates that work progress is ‘not according to plan, but recoverable’, and Red flag indicates that work progress is ‘not according to plan, but recoverable only with difficulty’ (Hughes & Cotterell, 2009; Kerzner, 2013). The RAG report though may indicate obstacles to progress, it can be seen as a subjective judgment of the current status of progress, and so might hide critical issues if the wrong flag/light is waved by the reporter at a particular performance meeting.

## 2.3 Factors influencing project duration

This section summarises the literature reviewed related to RQ2:

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

The reviewed literature on factors influencing software project outcome include relevant studies that address generic factors influencing project failure (see section 2.3.1) as well as factors specifically influencing schedule delay (section 2.3.2). RQ2 follows on from RQ1, and with its opening ‘in such an environment’ it refers to projects developing large software systems through globally distributed teams (see RQ1); therefore, relevant studies that examine developing large software systems (section 2.3.3) or in a globally distributed environment (section 2.3.4) were also reviewed for factors influencing schedule duration. Finally, relevant studies investigating project risk management (section 2.3.5) were examined for their potential in identifying the factors influencing project outcome. It may be noted that, the professional bodies of knowledge such as PMI (2008), APM (2006), CMMI (Chrissis et al. 2011), or PRINCE2 (OGC, 2009) were not examined in detail here because they offer generic frameworks that cannot be used effectively to look for specifics factors influencing schedule duration.

### 2.3.1 Project failure

Considering the outcome of a software project as success or failure appears to be relative to the perspective of the project participant and their role in the project (McLeod & MacDonell, 2011). Example empirical studies suggest that developers (Procaccino et al. 2006) and project managers (Procaccino & Verner, 2006) see project success from personal and project related perspectives. At the personal level, project success was seen in delivering quality product, having a sense of achievement, and having been provided with enough independence. At the project level, success was seen in the final system to be working as intended, having met customer requirements, and the project was delivered when needed by the customer (not necessarily on-schedule). Thus, the conventional criteria of meeting project schedule, cost, scope/quality may reflect only the organisational or project performance perspective - see for example (Lehtinen et al. 2014).

Therefore, rather than an agreed upon concept, project outcome may be better viewed against the complex interaction of the project participants' perspectives (McLeod & MacDonell, 2011). A successful project in the eyes of the software provider may be seen as failure by the acquirer. As this research investigated particular problems of schedule delay, it focussed on project failure as not meeting schedule targets. Table 2.1 lists some (for illustration rather than comprehensiveness) of the common factors reported in the reviewed literature as causing failure in software projects. The factors put forward by the studies reviewed in this section present very usefully the different factors influencing project failure, and can be useful indicators generally (Verner et al. 2008), but then may be less specific to what influences schedule duration.

| # | Factor | Description | Studies |
|---|---|---|---|
| 1 | Project goal | Project objectives are undefined, unclear, unmeasured, or unrealistic | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Tarawneh et al. (2008); Ebert (2007); Charette (2005) |
| 2 | Estimates | Estimates are inaccurate, unreliable, or inflated | Cerpa & Verner (2009); Lehtinen et al. (2014); Verner et al. (2008); Tarawneh et al. (2008); Nelson (2007); Charette (2005); Verner et al. (1999); Brooks (1995) |
| 3 | Requirements | Requirements are undefined or unclear; or the final system fails to meet customer/user requirements | Lehtinen et al. (2014); McLeod & MacDonell (2011); Pertersen et al. (2014) Nasir & Sahibuddin (2011); Verner et al. (2008); Ebert (2007); Procaccino & Verner (2006); Procaccino et al. (2006); Charette (2005); Boehm & Ross (1989) |
| 4 | Schedule | Project schedule or delivery date is unrealistic, milestones are unmeasurable, or aggressive schedule with overlapping phases | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Cerpa & Verner (2009); Lehtinen et al. (2014); Verner et al. (1999); Tarawneh et al. (2008); Verner et al. (2008); Nelson (2007); Sauer et al. (2007); Brooks (1995); Abdel-Hamid & Madnick (1991); Boehm & Ross (1989) |
| 5 | Reporting project status | Progress reports are inaccurate, out-of-date, or overoptimistic | Lehtinen et al. (2014); Nasir & Sahibuddin (2011); Charette (2005); Brooks (1995) |
| 6 | Project risk | Project risks are unmanaged, uncontrolled, or unassessed | Cerpa & Verner (2009); Nasir & Sahibuddin (2011); Nelson (2007); Verner et al. (2008); Ebert (2007); Charette (2005); Verner et al. (1999) |
| 7 | Communication | Lack of, ineffective, or excess | Lehtinen et al. (2014); McLeod & MacDonell (2011); Nasir & Sahibuddin |

| # | Factor | Description | Studies |
|---|---|---|---|
| | | communication | (2011); Charette (2005); Brooks (1995) |
| 8 | Technology | Missing or insufficient tools during project execution, introducing new technology in the middle of the project, or project team unfamiliar with the technology | Lehtinen et al. (2014); Lu et al. (2010); McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Charette (2005); |
| 9 | Change control | Ineffective management and monitoring of change, or no change control system in place | Nasir & Sahibuddin (2011); Cerpa & Verner (2009); McLeod & MacDonell (2011); Verner et al. (1999) |
| 10 | Scope | Inappropriate, unachievable, or large scope; or scope changes during project execution | McLeod & MacDonell (2011); Cerpa & Verner (2009); Tarawneh et al. (2008) |
| 11 | Staffing | Inadequate or unmotivated project staff, staff added late to the project, or large number of staff (big team) | Lehtinen et al. (2014); Cerpa & Verner (2009); McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Procaccino et al. (2006); Verner et al. (1999); Abdel-Hamid & Madnick (1991); Boehm & Ross (1989); Brooks (1995) |
| 12 | Customer | Lack of involvement from customer; the customer has unrealistic expectations, is unavailable, or is unsatisfied | Lu et al. (2011); McLeod & MacDonell (2011); Lehtinen et al. (2014); Cerpa & Verner (2009); Nelson (2007); Procaccino & Verner (2006); Procaccino et al. (2006); Verner et al. (1999); DeMarco (1982) |
| 13 | Project characteristics | The project is large, complex, or has a long duration | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Charette (2005); Brooks (1995) |

| # | Factor | Description | Studies |
|---|---|---|---|
| 14 | Development process | The development method is inadequate, ineffective, or non-standard | Tarawneh et al. (2008); McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Lehtinen et al. (2014); Lu et al. (2010); Charette (2005); Cerpa & Verner (2009); Verner et al. (1999); Boehm & Ross (1989); Brooks (1995) |
| 15 | Project management | Inadequate or insufficient planning, inadequate project monitoring and control, irregular project meetings, informal project management methods/techniques/quality controls, or inexperienced project manager | Lehtinen et al. (2014); Nasir & Sahibuddin (2011); Ebert (2007); Lu et al. (2010); Charette (2005); McLeod & MacDonell (2011); Verner et al. (1999) |
| 16 | Product | The final system does not work as intended | Procaccino & Verner (2006); Procaccino et al. (2006) |

**Table 2.1 Factors causing software project failure**

### 2.3.2 Schedule delay

Schedule delay refers to completing a project or project phase later than was originally planned. The reviewed literature on project schedule delay suggests that various factors can influence schedule delay (see Table 2.2). These studies tend to attempt an understanding of the external factors influencing schedule duration, usually relying on the project participants' recall of these factors (Deephouse et al. 1996). Studies that attempt such understanding through an in-depth analysis of, for example written records (Deephouse et al. 1996), might provide a different/internal view of how progress or schedule duration is controlled. For example, Phan et al. (1995) found that the reasons for schedule delay include factors such as: customer and management changes, technical complexities, unrealistic project plans, staffing problems, inability to detect problems early, insufficient project planning, underestimated project scope, and insufficient contingency planning. While this is illuminating, it does not seem to offer insight into how the potential influence of some of these factors can combine with others to cause schedule delay.

Detailed studies into project schedule behaviour include Rainer's PhD thesis (Rainer, 1999), where an in-depth case study was conducted into two projects at IBM. The research tested previous hypotheses on 'waiting' in software project schedules, investigated the change in actual time usage in both project and phase levels, and investigated the relationship between these two levels and their relationship to schedule behaviour. Rainer compared the workload, capability, planned schedule, and events of two projects in order to understand the relationship among these factors and changes in the schedule. The study suggests that these factors influence on one another and the project schedule, but concluded that the research was unable to determine definite causes that explained schedule delay (page: 150). The projects studied were collocated and developed small to medium size software systems. Future studies that investigate the execution of projects that develop large software systems in globally distributed environments might yield a different perspective on schedule delay.

Earlier work by Goldratt (1997) on Critical chain - see also section 2.4.6, which transferred ideas from organisational production lines where the different stages of manufacture depend on the components developed by earlier ones, sheds light on several factors that may be impacting slippage; for example (i) multitasking through increasing lead time (page, 126) (ii) task interdependencies through individual task delays accumulating to eventually cause project delay (page, 128) (iii) progress measurements' deficiencies in differentiating the critical from non-critical tasks thereby masking the critical path from the project manager's attention (page, 73).

Organisations can adopt various approaches to understand the causes of project slippage. Some of these approaches can be employed during a project so that remedial action can be taken, for example; as part of software process improvement (Allison, 2005; 2010) or adapting a projects' practices to meet industry standards such the one illustrated by Bass et al. (2013)'s study where an organisation tailored their Agile practices to meet CMMI level 5 requirements; another example is ABC projects, the company provided the research data (presented in Chapter 3) applying CMMI levels 3 and 5 process areas (at different sites) to their practices. Other approaches take place after a project has been completed for organisational learning, such as; project post implementation reviews used to record organisational/project specific knowledge after project completion in order to be developed internally for later projects (Doll et al. 2003); or root cause analysis technique/causal analysis and resolution process area of CMMI (Chrissis et al. 2011) used to detect and analyse the causes of problems (Lehtinen et al. 2014).

Table 2.2 lists some of the factors associated with schedule delay (rather than project failure in general) in software projects reported in the reviewed literature (for illustration rather than comprehensiveness). These include both those based on surveys and interviews; those conducted by large companies in different countries/cultures and those of small or medium companies in one country; both building custom and packaged software; and a range of project durations. Once again, these studies appear to list individual/isolated factors rather than the interaction among them and the way this may influence schedule duration.

| # | Factors | Studies |
|---|---|---|
| 1 | Changing requirements, or less time spent upfront to understand requirements | Patanakul (2014); McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Moløkken-Østvold & Jørgensen (2005); Blackburn et al. (1996) |
| 2 | Interdependency between software components | McLeod & MacDonell (2011); Blackburn et al. (1996) |
| 3 | Long repeated rework cycles | Rainer (1999); Blackburn et al. (1996) |
| 4 | Sequential (not overlapping) development stages | Rainer (1999); Blackburn et al. (1996) |
| 5 | Integration of software components | Lehtinen et al. (2014); McLeod & MacDonell (2011); Blackburn et al. (1996) |
| 6 | Large team size | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Sauer et al. (2007); Moløkken-Østvold & Jørgensen (2005); Brooks (1995); Blackburn et al. (1996) |
| 7 | Reduced information flow among development teams | McLeod & MacDonell (2011); Blackburn et al. (1996) |
| 8 | Minimum reuse of existing software components | Blackburn et al. (1996) |
| 9 | Less experienced workforce | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Carmel (1995) |
| 10 | Lack of concurrent activity and concurrent information flow | Moløkken-Østvold & Jørgensen (2005); Blackburn et al. (1996) |
| 11 | Reduced monitoring and control: e.g. slippage one day at a time, masking true progress status (the 90% syndrome), or fuzzy milestones. | McLeod & MacDonell (2011); Lehtinen et al. (2014); Deephouse et al. (1996); Brooks (1995); Boehm (1981) |
| 12 | Unrealistic schedule | Lehtinen et al. (2014); Nasir & Sahibuddin (2011); Deephouse et al. (1996) |
| 13 | External factors (such as starting late or lack of funding) | DeMarco (2011); McLeod & MacDonell; (2011) Patanakul (2014); Rainer (1999); |

| # | Factors | Studies |
|---|---|---|
| | | Jenkins et al. (1984) |
| 14 | Increased volume of coordination and communication | Patanakul (2014); McLeod & MacDonell (2011); Brooks (1995); Nasir & Sahibuddin (2011); Deephouse et al. (1996) |
| 15 | Improper planning | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Nelson (2007) |
| 16 | Inappropriate/ineffective methods and techniques | McLeod & MacDonell (2011); Lehtinen et al. (2014); Nasir & Sahibuddin (2011) |

**Table 2.2 Factors influencing schedule delay in software projects**

### 2.3.3 Large software systems

RQ2 sought to identify the influencing factors on schedule duration, including, in projects developing large software systems (LSS). The reviewed literature on projects developing LSS identifies specific factors (see Table 2.3) being exacerbated in LSS projects due to size. These studies used approaches such as surveys, case studies, and interviews on collocated projects; studies into developing LSS in a distributed project environment would complement their findings.

Projects developing LSS have been characterised as being uncertain, complex, involving large number of partners, and typically commissioned by governments and delivered by private firms (Patanakul, 2014). Large software systems can be distinguished by needing more than one management level to coordinate the development effort, multiple teams developing parts of the software, and taking more than six months of development period or 500,000 Lines of Code (Zmud, 1980; Sommerville, 2011). An LSS can be developed within an enterprise architecture environment (see section 2.4.4) where multiple subsystems are developed, each can operate independently, but are all subordinated to the central/enterprise system (Petersen et al. 2014).

Relevant past research include Petersen et al. (2014)’s case study into obstacles to progress in LSS projects and its effect on delay at Ericsson AB. The study examined the development of a system of systems (SoS) comprising 9 complex systems, that uses Iterative and Incremental Development (IID) with Agile practices. SoS is a set of independent systems incorporated into a larger system to deliver specific capabilities. Petersen et al. (2014)’s study examined the SoS view (i.e. integration of the systems) and the system view (i.e. individual system development and delivery) looking for common causes that obstruct the flow of the development process. The study identified factors obstructing progress relating to the following areas: architecture and technical dependencies, defects in project artefacts, requirements definition and management, overhead in updating and aligning release plans, and generic factors applicable to the overall development process such as being overloaded with work.

| # | Factor | Description | Source |
|---|---|---|---|
| 1 | Complexity of interdependency and interaction | The interdependency and interaction required among: the large number of people, the stages of the development process, the integrated development environment (e.g. CASE tools - computer aided software engineering), or the many parts of different systems that form the LSS (architecture dependencies) may increase uncertainty of delivering LSS projects on schedule. As will be seen in the course of this thesis (Chapter 7) that problems with interdependency and interaction among project elements can influence schedule duration | Patanakul (2014); Petersen et al. (2014); Nelson (2007); McLeod & MacDonell (2011); Hustad & Lange (2014); Zmud (1980); Jones (1995); Curtis et al. (1988); Belady & Lehman (1976); Kitchenham (1987); Malcolm (1990); McDermid (1990); Yeo (2002); Brooks (1995); Shermer et al. (1981); Rook (1986) |
| 2 | Requirements | The complexity of developing LSS may make it challenging to define clear, concise, and complete requirements upfront, which results in unanticipated changes during the development process | Patanakul (2014); Petersen et al. (2014); Nelson (2007); Zmud (1980); Yeo (2002); Jones (1995); Curtis et al. (1988); Rook (1986) |
| 3 | Change | Change in technology can influence design and architecture, change in environment can influence requirements, and change in the development team can influence development schedule | Petersen et al. (2014); Patanakul (2014); McLeod & MacDonell (2011); Zmud (1980); Royce, (1970); Kitchenham (1987); Bauer & Brichall (1978); Rook (1986) |
| 4 | Development model | Using only one approach, e.g. the IID model, to develop LSS may create development challenges since different teams develop different parts of the system causing instability of the system architecture. A combination | Sommerville (2011); Nelson (2007); Yeo (2002); McLeod & MacDonell (2011); Boehm (1988); Zmud (1980) |

| # | Factor | Description | Source |
|---|---|---|---|
| | | of different models may be used to develop parts of the system depending on the degree of uncertainty of the requirements | |

**Table 2.3 Factors exacerbated in LSS projects**

### 2.3.4 Global software development

RQ2 sought to identify the influencing factors on schedule duration, including, in projects developing software in globally distributed environments. The reviewed literature (see Table 2.4) on Global software development (GSD) suggests that, certain factors can be intensified in GSD projects due to complexity in coordination and control (Conchúir et al. 2009), and increased risk (Verner et al. 2014). Although Hossain et al. (2009) argues that using some of the Agile practices may reduce coordination difficulties exhibited on GSD projects; devising practical solutions to GSD problems is still evolving (Šmite et al., 2010; Damian & Morita, 2006); and more empirical studies are needed (Verner et al. 2012b ; Silva et al. 2010) to determine the factors influencing project duration.

GSD refers to developing software through teams that are located in multiple sites, with multicultural backgrounds, and distributed globally (Herbsleb & Moitra, 2001; Verner et al. 2014). In this model, distributed teams collaborate to develop parts of the software, coordinate the development activities (i.e. manage dependencies among project tasks), and communicate extensively to progress daily project work (Herbsleb, 2007; Malone & Crowston, 1994).

It has been argued that GSD can enable organisations to: use specialised workforce (Damian & Lanubile, 2004; Verner et al. 2014), reduce development cost (Damian & Lanubile, 2004; Carmel & Agarwal, 2001; Nidiffer & Dolan, 2005; Verner et al. 2014), increase speed of delivery through time zone difference in 'round-the-clock' development (Herbsleb & Moitra, 2001; Verner et al. 2014), and gain proximity to the market (Herbsleb & Moitra, 2001; Damian & Lanubile, 2004). However, Conchúir et al. (2009) have found that these benefits may not be fully realised. Verner et al. (2014) rightly noted that 'Organizations frequently consider offshore systems development in the belief that projects can be completed at lower cost. While prices quoted by offshore vendors may be very appealing additional risks must be considered when considering offshore systems development. These risks have associated costs and typically result in additional indirect costs which add to the total payment required for the delivered system. However, such costs are seldom considered by companies at the outset of a project, yet may become painfully apparent once the project is under way' (page: 55).

Relevant past research include Herbsleb et al. (2001)'s modelling of the extent of schedule delay in multisite development (UK, Germany, India) of software at Lucent Technologies. The researchers used data from the project's change management system (used to register request for new, and change existing, functionality; and fixing defects) to measure actual duration of project tasks, and used survey data (including communication patterns and coordination activities); to compare same-site work to cross-site work. The researchers found that work distributed across sites took longer to complete compared to same-site work; and that the factors influencing the delay can be indirect, such as: the size of the change required, the number of software components affected by the change, and the number of people involved in carrying out the change. They also found that, the latter to be the differentiating factor because of the challenges in coordination and communication across sites.

| # | Factor | Description | Source |
|---|---|---|---|
| 1 | Coordination | Activities distributed among people who are located at different sites take longer to perform and complete due to difficulty in managing uncertainties caused by interdependent tasks | Herbsleb & Moitra (2001); Mockus & Herbsleb (2001); Herbsleb (2007); Kommeren & Parviainen (2007); Nidiffer & Dolan (2005); Damian & Lanubile (2004); Ebert & Neve (2001); Carmel & Agarwal (2001); Verner et al. (2014); Herbsleb & Mockus (2003); Herbsleb et al. (2001); Ramasubbu & Balan (2007); Conchúir et al. (2009); Piri & Niinimäki (2011) |
| 2 | Communication | Lack of effective communication among project teams at distance may increase duration of performing project activities due to restricted information flow across sites, indirect interaction, and lack of contextual information | Herbsleb & Moitra (2001); Herbsleb & Mockus (2003); Herbsleb et al. (2005); Kommeren & Parviainen (2007); Nidiffer & Dolan (2005); Mockus & Herbsleb (2001); Ebert & Neve (2001); Herbsleb & Grinter (1999); Damian & Lanubile (2004); Carmel & Agarwal (2001); Verner et al. (2014); Herbsleb et al. (2001) |
| 3 | System integration | Unavailability of the software parts built by various teams, breakdown of the technical environment facilitating the integration, or integration activities taking place at various sites | Herbsleb & Grinter (1999); Battin et al. (2001); Kommeren & Parviainen, 2007); Ebert & Neve (2001); Nguyen et al. (2008) |
| 4 | Project monitoring and | Management and tracking of project activities at distance might become difficult, especially when tasks are split between sites and | Herbsleb et al. (2005); Kommeren & Parviainen (2007); Damian & Lanubile (2004); Verner et al. |

| # | Factor | Description | Source |
|---|---|---|---|
| | control | dependency and coordination are not accompanied by sufficient communication, or different project plans are created at different sites | (2014); Carmel & Agarwal (2001); Holmstrom et al. (2006) |
| 5 | Cost increase | Project progress appears to be less efficient; in a study of Phillips' 10 year experience with a dozen of GSD projects (Kommeren & Parviainen, 2007) it was found that, up to 50% of the development effort was spent on overhead (such as extra project management and team coordination) and communication. Thus, the GSD for Phillips was in practice two to three times more costly compared to collocated development. This finding supports Verner et al. (2014)'s insight, quoted above, that GSD projects may turn out to be more costly than originally thought | Kommeren & Parviainen (2007); Conchúir et al. (2009) |
| 6 | Time-zone differences | Time-zone differences across the distributed sites could create delays in response time, and difficulties arise in transitioning phase products across sites effectively. As will be seen in the course of this thesis (Chapter 3) that reduced overlapping time across the sites delays completing project activities | Conchúir et al. (2009) |

**Table 2.4 Factors exacerbated in GSD projects**

### 2.3.5 Project risk management

The reviewed literature on project risk management (see Table 2.5) suggests that project risk can affect project outcome (Tate & Verner, 1990). Project risk might be defined as any event that constrains achieving project objectives, and can be managed through a process of: devising an approach to manage risks, identifying risks, assessing the risks' likelihood of occurrence and impact, devising a mitigation strategy, and controlling the risk throughout the development lifecycle (Burke 2013, page: 328; Cadle & Yeates 2008, page: 260). The reviewed studies into project risk management (Table 2.5) can be classified according to the purpose of the study.

| # | **Purpose** | **Studies** |
|---|---|---|
| 1 | Identify sources of uncertainty or risk factors on projects | Shahzad & Al-Mudimigh (2010); Shahzad et al. (2009); Zhou et. al (2008); Boehm & Ross (1989) |
| 2 | Measuring and controlling project risk and risk management strategies | Bannerman (2008); Taylor (2006); Jiang et al. (2002) |
| 3 | Evaluation of risk identification methods | Bakker et. al (2010); Feng et. al (2009); Williams et. al (2004a); Williams et. al (2004b) |
| 4 | Develop risk management frameworks, tools, or techniques | Alberts & Dorofee (2010); Dhlamini et al. (2009); Fan & Yu (2004); Rabbi & Mannan (2008); Carr et al. (1993) |
| 5 | Prioritisation of project risks | Sun (2009) |

**Table 2.5 Studies into project risk management**

Whilst these studies put forth useful ideas about project uncertainties, they seem less helpful in providing practical solutions to the challenges facing day to day project management. For example, Bakker et al. (2010) peer reviewed risk management literature from 1997 - 2009 and concluded that risk management is not being conducted in practice as it has been proposed in research, so ‘less is known about what actually happens inside the risk management process’ (page: 500). Taylor (2006) also argues that there is a mismatch between the prescriptions of managing risk in the literature - those of the Project Management Institute (PMI) - as opposed to the actual practice of experienced project managers. Furthermore, recent systematic literature reviews on risk mitigation in global software development (Verner et al. 2014) found that empirical support for the risks identified in the reviewed studies was moderate to low, and that risk mitigation advice was also limited.

Finally, while Carmel (1995) suggested that risk management (as opposed to risk) has no effect on schedule duration, Tate & Verner (1990) demonstrated that the ongoing management of risk and embedding risk mitigation strategies within the development process (e.g. the selection of a development model, approach to data structure, development tools, or mix of staffing) can reduce the risks of not meeting schedule, requirements, staff, and user-acceptance targets.

## 2.4 Interaction within the development environment

This section summarises the literature reviewed related to RQ3:

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

Software project execution is concerned with controlling a schedule of the technical tasks associated with developing the software; that is, the software development process (Rook, 1986; Kitchenham, 1987). Interaction among the process elements (constituent parts), however, can affect project outcome (Yamamoto et al. 2009; Nandhakumar, 1996). These elements could be human and nonhuman (Clegg et al. 1997), mutually influencing one another (Akkermans & Helden, 2002), in a complex network of interrelationships (Butler & Fitzgerald, 1999), within an enterprise development environment (Petersen et al. 2014); where factors internal and external to the project are interdependent (Scott & Vessey, 2002).

Therefore, examining the interaction among the process elements of software engineering models may indicate how they influence schedule delay (Kim & Pan, 2006; Butler & Fitzgerald, 1999). The complexity of such interactions is examined first (section 2.4.1), followed by examination of how the interactions may operate within the primary development models: waterfall (section 2.4.2), iterative and incremental development (section 2.4.3), component-based model (section 2.4.4), lean software development (section 2.4.5), and critical chain method (section 2.4.6).

### 2.4.1 Complexity of interaction in software projects

Software complexity has been defined as the interrelationship between the components that make up software in terms of: the number and variety of components, the number and variety of interactions, the number and variety of interdependencies, and the rate of change of the system (Schneberger & McLean, 2003).

The preceding definition can be extended to the managing of the software project as a whole to include human and nonhuman project elements (constituent parts) and any other element participate in delivering the project. Hence, complexity of software project management can refer to ‘the interrelationship between the project elements in terms of: the number and variety of project elements (human and nonhuman), the number and variety of interactions among project elements (that exert constraining or empowering influences), the number and variety of interdependencies (that exert direct or indirect influences), and the rate of change of the project situation/context’.

The reviewed empirical studies suggests that complexity can affect schedule duration (Carmel, 1995), or project failure (Chua & Verner, 2005), and that not only causal correlation between the factors, but also interrelationship between them can affect project outcome, and that different factors can vary in their strength of influence at different times (Nandhakumar 1996, page: 70).

### 2.4.2 Waterfall model

RQ3 sought examining the interactions that develop during project execution to understand the influencing factors on schedule delay. The Waterfall model (Royce, 1970), one of the primary development models, is known for its development phases following one another with the earlier phases feeding/flowing into the later ones, like a waterfall. This approach facilitates interaction among project phases/activities through (i) stage-gating, where project products are approved before moving to the subsequent phase (ii) complete definition of requirements before design commences (Pressman, 2005).

Thus, the interaction among the elements of the process can be seen as sequential, which make the model appropriate for projects that develop large software systems where the user requirements are well defined, understood, and less prone to change during the development process (Pressman, 2005). However, this sequential interaction may introduce delays due to waiting on obtaining the necessary approvals (stage-gating) before work can start in subsequent phases.

### 2.4.3 Iterative and Incremental development

Another primary development model is the iterative and incremental development (IID) emerged in the 1950s (Larman & Basili, 2003) promising to allow changes to the system design during the development process. Incremental development involves going through all the Design, Build, and Test stages of developing a subsystem or set of components, obtaining user feedback, and then going through the next increment and incorporating the feedback (Sommerville, 2011; Gilb, 1985). Iteration usually means creating different versions of the same software. Incremental in itself should not involve changes to existing functionality: in practice it does as later increments often require changes be made to earlier ones so that they inter-operate correctly; for example, the development team may change requirements and design at any time, leading to a mismatch between what is being worked on in each increment (Cockburn, 2008). This indicates increasing interaction among the development phases and various increments.

Since its inception, various implementations of the IID model have emerged such as (i) the Evolutionary delivery (Gilb, 1988; Gilb, 1985), where the product is partitioned into small delivery chunks encompassing the whole lifecycle stages (ii) the Phased development, that partitions the development process into larger chunks (Graham, 1992), more like small Waterfalls (iii) the Incremental Build and Test, where the software product is sliced at the build and test stages only to code and test the slices incrementally, whilst retaining the full product in the requirements and design stages (Graham, 1992). Thus, interaction among the process elements of an IID approach can be seen as sequential and/or parallel (Moløkken-Østvold & Jørgensen, 2005), this indicates increasing complexity in such projects as they require more management control and monitoring (Redmill, 1992; Graham, 1992).

This increased interaction, though considered appropriate for large scale software projects, was criticised for the time and effort taken for medium and small size projects (Sommerville, 2011; DeMarco & Boehm, 2002). Thus, a variation of the IID model, Agile, emerged in the 1990s favouring self-containing work activities, which attempts to limit the interaction of the development team with the outside world; for example through (i) embedding the client representative into the development team (ii) lessening the constraints prescribed by the standard process models (iii) focusing on the current design without accommodating future architectural changes. A variant of Agile, DSDM Atern (Dynamic System Development Method) claims to exercise control of project progress through ‘time-boxing’ i.e. focus on delivering on time by reducing scope if necessary (Stapleton, 1997). However, limiting the interaction of the internal project elements with the external surroundings cannot guarantee: the full-time availability of the client within the development team (Sommerville 2011, page: 60; Boehm 2002, page: 66), the effective representation of different views of all stakeholders (Sommerville 2011, page: 60; Boehm 2002, page: 66; Cadle & Yeates 2008, page: 80), or to overcome interaction challenges when developing software in a globally distributed environment (Sommerville 2011, page: 60; Cadle & Yeates 2008, page: 80; Boehm 2002, page: 67).

The numerous interactions among the project elements in an IID model increases the complexity of the system being developed, which may introduce delay in the project progress not immediately apparent, because the delay can be caused indirectly and by multiple factors:

- The execution of more than one increment in parallel was not a feature of the original IID and thus is not essential, but it does occur, and can be treated as a variant of IID (Moløkken-Østvold & Jørgensen, 2005). In this case, user feedback from one increment may not be incorporated in the subsequent increments.

- The developing software in the IID model can change frequently, so updating project deliverables takes additional time leading to project delays (Sommerville, 2011).

- Coordinating project activities, among teams working in different stages, becomes more difficult (Graham, 1992: Redmill, 1992), particularly when developing large software systems through globally distributed teams (Sommerville, 2011), due to communication problems where there are many interdependent tasks and changes in the software (Herbsleb, 2007).

### 2.4.4 Component-based model

The interaction among the project actors (the quest of RQ3) in the CBD model is different compared to the preceding two models. The component-based development (CBD) reuses existing software components rather than developing the whole software from scratch. The process model involves defining, developing, and integrating loosely coupled components to build the software systems. The CBD can be characterised as typically utilising independent components with specified interfaces, standards that facilitate the integration of components, middleware that handles component communication, and a development process which supports this (Sommerville, 2011; Cadle & Yeates, 2008; Pressman, 2005). These software components may be located on different computer machines and communicate through specified protocols (Distributed components). Some of these components (called Services) may operate independently of the platform they were developed for: thus a Service oriented system is a type of CBD (Summerville 2011, page: 455). In a Service oriented architecture (SOA), software components are stand-alone services, can execute on geographically distributed computers, and provide information, computer resources, and/or data access services to other components on request (Summerville 2011, page: 509).

A SOA system (Hustad & Lange, 2014) can be developed though a software product line approach; which is a set of applications with common application architecture and shared components, with each application specialised to reflect different requirements, and can be reused to develop new applications. The new application may require building new components, adapting some of the existing components, and/or configuring existing components to meet the new requirements (Sommerville, 2011; Greenfield & Short, 2003; Ebert & Neve, 2001). The new (SOA) application can be developed by the *product development* team, whilst the common application architecture and shared components are developed and/or modified by the *product line development* team. It is possible for the common application architecture and shared components to be developed using an enterprise architecture framework, which is a generic structure (i.e. set of descriptions, methods, common vocabulary, and reference models) that embed application domain knowledge (at organisation level) that can be extended to create more specific subsystems (Winter & Fischer, 2006; Urbaczewski & Mrdalj, 2006; Sommerville, 2011).

Thus, interaction among the process elements, in the CBD model, can be seen as a request/response form of interaction, with the service provider accommodating requests from multiple service requesters; and where a product line development approach is used and which is based on an enterprise architecture environment, more and complex interactions are expected (Hustad & Lange, 2014) between the various teams and various systems and subsystems. These may introduce delay in the project progress, due to:

- Increased complexity in coordination, communication, and control due to multiple systems being developed and integrated, particularly in a globally distributed environment (Petersen et al. 2014; Hustad & Lange, 2014).

- The need to develop adapters (code that facilitate integration between new and existing components/subsystems) which may arise as additional work during the Build stage (Sommerville, 2011).

- Multiple suppliers developing parts of the software (individual subsystems), where they may want to modify the shared components simultaneously when integrating their software within the enterprise system (Sommerville, 2011).

- Potential challenges with integrating the components that are sourced from outside organisations, certification, or requirements trade-off (Sommerville, 2011).

- Management control may be reduced due to some of the subsystems being developed by teams outside the immediate control of the project manager of the product development team.

Recent relevant research include Hustad & Lange (2014)'s case study into a SOA development within enterprise architecture environment in five projects at a large Norwegian government organisation, in which four of the projects ran in parallel and the fifth oversaw the integration and communication among the four projects. This was a large and complex project involving multiple stakeholders across different locations within Norway and aiming to create a common integration platform for the large number of existing legacy systems and offer shared services to its stakeholder organisations, using Agile practices. The researchers used interview for data collection and allowed emergence of themes to make sense of what was happening on the projects including development of visual models. The researchers found several factors that increased the complexity of the projects studies, such as: coordination and communication challenges with running the projects in parallel, difficulty in project management and development method, and challenges with internal and external competencies.

### 2.4.5 Lean software development

A more recent development approach, Lean software development (LSD), indicates yet another type of interaction among the project actors compared to the conventional models described thus far. The LSD is the application of Toyota’s product development practices to software development process, which focuses on optimising the flow of work items during project execution, increasing efficiency, and reducing waste (Poppendieck, 2007). The LSD combines specific techniques, such as reducing waste (e.g. unclear requirements or defects) with the IID model (Ikonen et al. 2010). The LSD is similar to the production line approach to development (section 2.4.4) in focusing on continuous flow of work.

The LSD uses a visual board (or Kanban) to track progress of developing software components through the project phases; i.e. as the component is designed, then moved to build, then to test, and one can check that a component is not sitting somewhere and not picked up by the next phase (Middleton & Joyce, 2012; Ikonen et al. 2011). The interaction between the phases is managed through the Queue and WIP mechanisms, which indicate the amount of work waiting to be processed by the phase, and the amount of work the phase is processing at the time, respectively. Thus, the interaction among project elements in a LSD model can be seen as a regulating/dynamic one; where the downstream phase only takes up work from the queue if it can process them simultaneously to the ones being processed. This approach appears to bring to fore factors obstructing progress, which may help in decreasing delay in the project progress as it occurs.

Reviewed empirical studies applying the Lean Kanban approach in a software engineering context indicate improvement in the delivery time (Staats et al. 2011), due to the process being under the development team’s control rather than being influenced by upstream or downstream phases or third parties (Middleton & Joyce 2012, page: 25). As Kanban was not designed with software engineering projects in mind (Janes & Succi 2009), further academic research is needed to ascertain the practicality of the approach in the field; in particular, its suitability to projects: developing LSS (Pernstål et al. 2013), executed in a GSD environment, or in a supplier-acquirer relationship since Lean does not seem to work well with targets, milestones, and Gantt charts (Middleton & Joyce, 2012).

### 2.4.6 Critical chain method

Another development approach is the Critical chain method (CCM), developed by Goldratt (1997), which applies certain techniques to a development model, such as: providing realistic estimate of task duration, focusing of the resource on one task at a time instead of multitasking, and inserting time buffers at specified points in the schedule to absorb slippage (Kerzner, 2013) - the latter two features appear to be similar to the Lean Kanban approach described earlier. The CCM is also similar to the production line approach to development (section 2.4.4) in its emphasis on continuous flow of development work.

The CCM uses one estimate of duration and one estimate of contingency/additional time, which can be a percentage of the overall critical chain tasks, to allocate buffers prior to critical activities and just before the end of the project end date, which account for the uncertainties associated with duration estimates (Kerzner, 2013). Thus, the interactions among process elements, in CCM, can be seen as an adjusting one; since the usage of the buffer means schedule overrun, which may indicate problems obstructing progress, and help in decreasing delay in the project progress.

In the reviewed literature, the CCM has been praised for improving control on schedule duration through embedding resource dependencies, unchanging critical path, and embedding uncertainties of activity duration into schedule buffers (Leach, 1999) compared to the time dependency view of critical path method (Herroelen et al. 2002). Whilst, others have criticised the hype surrounding the CCM for it being merely an integration of existing project management principles rather than a revolutionary approach (Raz et al. 2003; Trietsch, 2005); other researchers argued that fundamental differences exist between the CCM and critical path method (Lechler at al. 2005). However, interim findings from an empirical study (Stratton, 2009) concluded that the CCM's weaknesses that were identified by the previous critics (above) did not appear as significant in the case study results.

## 2.5 Summary

Organisations can take various approaches to understand the causes of project slippage (section 2.3.2). Academics too have undertaken many studies to investigate factors affecting project outcome (as summarised in Tables 2.1, 2.2, 2.3, 2.4, and the factors listed in section 2.4.4). Key factors suggested by these studies relate to: project goal, estimates, requirements, schedule, reporting project status, project risk, communication, technology, change control, scope, staffing, customer, project characteristics, development process, project management, product, interdependency, interaction, coordination, component/system integration, time-zone difference, team size, management control, and multiple suppliers. Despite these insights, software projects are still delivered late and short of their objectives (McLeod & MacDonell, 2011; Winter et al. 2006; Nasir & Sahibuddin, 2011) and researches such as Rainer (1999) and Phan et al. (1995) concluded that it was difficult to confirm individual (Rainer 1999, page: 150) or combined (Phan et al. 1995, page: 279) causes that explained schedule delay.

Broadly, two observations can be drawn: first on the findings of the studies reviewed in this chapter; second on the methods used to reach the findings.

Firstly, the studies reviewed in this chapter suggest that projects are complex social entities with lots of different interacting components which could go wrong, so identifying a single common cause of failure from the literature is difficult. Furthermore, projects themselves are becoming more complicated as bigger and more technologies/teams are integrated; for example, Hoegl & Weinkauf (2005) observed inherent complexity in managing task interdependencies in multi-team projects that may affect project slippage, through shifting of the responsibility from the teams themselves during the concept phase of the project, to project/programme management during project execution. Although Hoegl & Weinkauf (2005)'s research was conducted in the automotive industry and mainly using questionnaires; their findings should be closely attended to by the software engineering industry for its valuable insights. Perhaps, considering the management of projects developing complex system of systems, such as the one studied by Petersen et al. (2014) (section 2.3.3) within the project environment studied by Hoegl & Weinkauf (2005) would illuminate the picture. Such situation would increase the complexity and interaction that need extra coordination among the project elements, and identifying one factor as causing schedule delay would be misleading for it simplifying (abstracting) what might actually be happening on the ground.

Although the example studies above, including Petersen et al. (2014), do not identify the precise mechanisms by which coordination is carried out (or not), and does not specifically look into the complexity inherent in managing projects, the research presented in this dissertation does look at these. It will be seen in chapters 7 and 8 of this dissertation that task interdependencies and weak coordination actually contribute to schedule delay, and that the use of Actor-network theory (Chapter 8) to identify project actors, the relationships among them, and the interactions influencing schedule delay is a way of getting to grips with the complexities of project execution. This is consistent with and supported by the studies alluded to in the preceding paragraphs.

Secondly, the research methods used by the studies reviewed in this chapter (sections 2.3 and 2.4) employed conventional approaches to project management research - surveys, interviews or case studies (see Table 4.1 in Chapter 4). Although these approaches have contributed widely to our understanding of both the internal factors in managing projects, such as planning or control and the external factors, such as human or organisational aspects; some researchers (e.g. Winter et al. 2006; Sankaran et al.

2013; Clegg, 2013; Söderlund, 2004; Cicmil et al. 2006; Morris, 2002) have argued that the project management literature (including the authoritative PMI Body of Knowledge) appear to have been unable to address the complexity of what actually happens in practice.

The complexity of project management field was emphasised by Winter et al. (2006, page: 641) ‘By far the clearest pattern to emerge from all the practitioner inputs to the Network is the sheer complexity of projects and programmes across all sectors and at all levels, encompassing all manner of aspects including the multiplicity of stakeholders, and the different agenda, theories, practices and discourses operating at different levels within different interested groups, in the ever-changing flux of events’. In addition, Clegg (2013) noted ‘Project managers may not realize it but the most important aspects of what they manage are meanings, interpretations, and politic of projects and not merely the technical aspects’ (page: 18). Therefore, academic research should focus more on project complexity in order to obtain a better understanding of the project and project management field, which suggests that, different research approaches, beyond the conventional ones, may be needed to enhance our understanding and address the complexity inherent in project management research; Muller et al. (2013) argued that ‘if we always do what we always did then we should not be surprised that we always find what we always found’ (page: 24).

The preceding argument suggests that there is no one simple reason for schedule delay; researchers need to attend to the complexity inherent in managing projects in order to better understand the influencing factors on schedule delay; ‘the interactions and interdependencies that tend to characterize our management systems will similarly characterize the problems that beset such systems’ and that ‘no one thing seems to cause difficulty, but the accumulation of simultaneous and interacting factors’ (Abdel-Hamid & Madnick 1991, page: 8). Therefore, we need to:

1. Study empirical data of real projects - this is because schedule delay can be caused by activities and events beyond the immediate control of the project, and that these

events may only emerge as a result of the interactions, internally as well as with the external environment to the project, at play during project execution that cannot be easily foreseen beforehand - the subject of Chapter 3 (A background to ABC).

2. Use different research approaches to the conventional ones used in project management research (Winter et al. 2006), one which integrates various perspectives of the project and examines controlling schedule duration within the context of the interactions that emerge among project actors during actual project execution; an approach that mixes different methods to bring out the multiple facets of project execution - the subject of Chapter 4 (An appropriate research strategy).

# 3 A background to ABC

## 3.1 Introduction

Chapter 2 expressed the need to draw from empirical data in order to have better understanding of controlling schedule duration during software project execution and the way it influences schedule delay.

This chapter describes the context of the projects at ABC - a fictional name for a real organisation - that provided the data for investigating the causes of schedule delay in these projects. This will help to position the ABC organisation’s work processes within the ranges of development practices recognised by both knowledgeable practitioners and researchers, and that were the main subject of Chapter 2 on the existing literature.

When conducting empirical research in software engineering, scholars have called for the provision of sufficient contextual information to ensure that the particularities of the studied area are taken into consideration by other researchers and practitioners (Kitchenham et al. 2002); and have emphasised that empirical studies in software engineering should support ‘the advancement of the software engineering field through an iterative learning process’ (Basili et al. 1986, page: 733). This chapter, therefore, provides a detailed description of ABC’s work processes to equip the reader with the necessary context needed to put the subsequent analyses in perspective. The chapter describes the ABC’s software delivery model, how project execution was managed, and reporting of project performance.

## 3.2 Software delivery model

ABC is a multinational management consulting, technology services, and outsourcing company. It is one of the world's largest consulting firms and is a Fortune Global 500 company. The company has more than 300,000 employees, serving clients in more than 200 cities in more than 50 countries. ABC’s client base include more than three-quarters of the Fortune Global 500. The company manages software development projects through globally distributed teams, largely to lower development cost and win contracts, and both develops custom solutions and configures package applications for its clients. The projects investigated in this research developed and integrated large software systems within an enterprise architecture environment on a ‘time and materials’ contract to a public organisation in the United Kingdom (UK) using development teams in the UK and India. The UK part of ABC is accredited the CMMI for Development (Capability Maturity Model Integration - see Chrissis et al. 2011) level 3, whilst the India part achieved level 5.

### 3.2.1 Enterprise development architecture

The architecture of the development environment is now described to illustrate how the various parts of the enterprise system fit together - see Figure 3.1. As will be noted, development of enterprise systems brings in technologies and constraints that ‘traditional’ development does not have to deal with, and that issues arising from such projects that influence schedule delay would add to our academic knowledge.

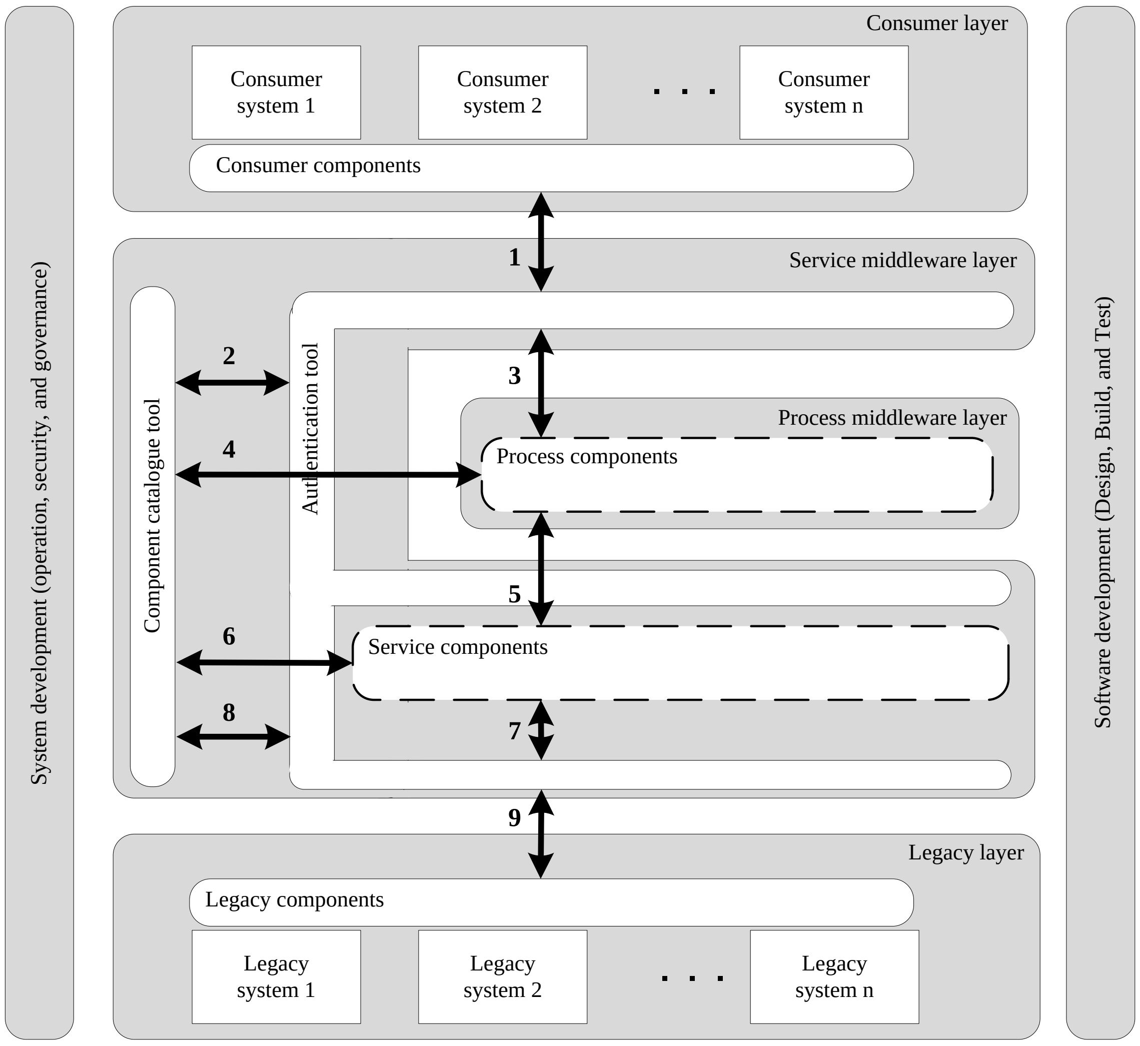


**Figure 3.1 Logical architecture of the development environment**

The projects studied in this research developed the Service and Process software components indicated in Figure 3.1. This is very much like the Service oriented architecture model described earlier, which sits on the existing enterprise architecture environment of the Client, exhibiting increased complexity of interaction among the software components/layers - see section 2.4.4 (Chapter 2).

See Table 3.1 for description of the constituents of the development environment, and Table 3.2 for description of the steps of message routing through the enterprise system architecture.

| **System component** | **Description** |
|---|---|
| Consumer layer | Consuming applications and systems that initiate calls to the rest of the system architecture |
| Process middleware layer | Performs multiple business operations within a single business process function; enabling orchestration of Service components within the Service middleware layer. For example, it can provide business rules or reuse of the process across Consumers |
| Service middleware layer | Mediates the invocation of legacy components within the legacy layer, and process components within the process middleware layer. The legacy components generally store and access data in a way that is different from the presentation requirements of the modern consumer components; and thus, the service components provide the formatting of data structures that match the consumer requests with legacy responses through (i) validation to ensure that the inbound and outbound messages conform to an agreed format (ii) transformation to change the format and structure of the passing messages to accommodate the presentation format of the Legacy and Consumer systems (iii) routing to specify the correct path of the message to pass through from the consumer to the legacy system/s |
| Legacy layer | Legacy systems are typically master databases that provide data access to service components through a set of legacy components |
| Authentication tool | Enforces security policy on request messages coming from the consumer to the service and process components. It also provides validation of message for integrity. This is an off-the-shelf application provided by an external vendor.<br>It is configured to operate within the development environment and the service and process components need to build interfaces to be able to interact with the authentication tool. Although this tool was not a project deliverable, it was newly introduced as part of the programme which the ABC projects needed to interact with, and which had to be configured by the Product Line Development team (see section 3.3). Thus, the Authentication tool was managed by the Technical |

| **System component** | **Description** |
|---|---|
| | environment function (see section 3.3.9) |
| Component catalogue tool | At development time, this is used for registering service and process components and configuring meta data. At runtime it is also used as repository for dynamic lookup of component endpoints managed by the repository. This is an off-the-shelf product provided by an external vendor, it is configured to operate within the development environment and the service and process components need to build interfaces to be able to interact with the Component catalogue tool. Although this tool was not a project deliverable, it was newly introduced as part of the programme which the ABC projects needed to interact with, and which had to be configured by the Product Line Development team (see section 3.3). Thus, the Component catalogue tool was managed by the Technical environment (see section 3.3.9) |
| Consumer components | Software components hosted on the consumer layer. The consumer components are developed by projects managed by ABC, but which are not in scope for analysis in this research |
| Service components | Software components hosted on the service middleware layer. These components are developed by ABC project teams |
| Process components | Software components hosted on the process middleware layer. These components are developed by ABC project teams that are different from the ABC teams that develop the service components |
| Legacy components | Software components hosted on the Legacy layer. The Legacy components are developed by the peer supplier, a company equivalent to ABC that develop parts of the overall software for the client and whose components must work with the components developed by ABC for the client in order for the enterprise system to function correctly |
| System development (Operation, security, governance) | Comprising the operations, security, and governance aspects of developing the overall system. Operations is concerned with how to operate the overall architecture such as log monitoring and error handling; Security is concerned with setting and enforcing security policy; and governance is concerned with governing the architecture layers and the components |

| System component | Description |
|---|---|
| | within them, such as managing versioning and lifecycle of the components |
| Software development (Design, Build, Test) | Comprised the Design, Build, and Test stages for each of the architecture layers, which adopted a product line model. In this approach the consuming projects define their data access requirements to the product line, which comprised (i) Development architecture governance (Technical architecture, Solution architecture, and Technical environment) to decide whether to reuse or manufacture new component (ii) Legacy product line to build/upgrade Legacy components (iii) Service and process product lines to build/upgrade the middleware components (iv) the components then are handed over to the consuming project |
| Double headed arrows (Enterprise service bus) | Represent the Enterprise service bus transferring messages between various systems. Enterprise service bus is a software medium (middleware) used to transport messages and data among various software applications that need to send/receive information from other software applications within the Client's enterprise wide system; and which is managed by ABC |

**Table 3.1 Constituents of the development environment**

| Step | Description |
|---|---|
| 1 | In order for the consumer component, hosted on the consumer layer, to invoke the process component, hosted on the process middleware layer, it first connects with the authentication tool which enforces security policy for the process component |
| 2 | The authentication tool invokes the component catalogue tool to retrieve end point details (i.e. the address) for the process component hosted on the process middleware layer |
| 3 | The authentication tool then invokes the corresponding process component hosted on the process middleware layer |

| Step | Description |
|---|---|
| 4 | The process component invokes the component catalogue tool to retrieve end point details for the service component hosted on the service middleware layer |
| 5 | The process component invokes the service component hosted on the service middleware layer |
| 6 | The service component invokes the component catalogue tool to retrieve end point details for the corresponding authentication tool instance responsible for enforcing security policy for the legacy component hosted on the legacy layer |
| 7 | The service component invokes the Authentication tool instance responsible for enforcing security policy for the legacy component hosted on the legacy layer |
| 8 | The authentication tool invokes the component catalogue tool to retrieve the end point details for the legacy component hosted on the legacy layer |
| 9 | The authentication tool invokes the legacy component hosted on the legacy layer |

**Table 3.2 System architecture message routing**

### 3.2.2 Global software development at ABC

This section describes the global software delivery model adopted by ABC's Product Development team (described in section 3.3), and the interactions taking place during project execution to exchange various project products. The product development (PD) team were located at the UK client site, UK ABC development site, and India ABC development site - see Figure 3.2. Earlier, section 2.3.4 (Chapter 2) outlined the key challenges in global software development, and ABC's situation is no different.

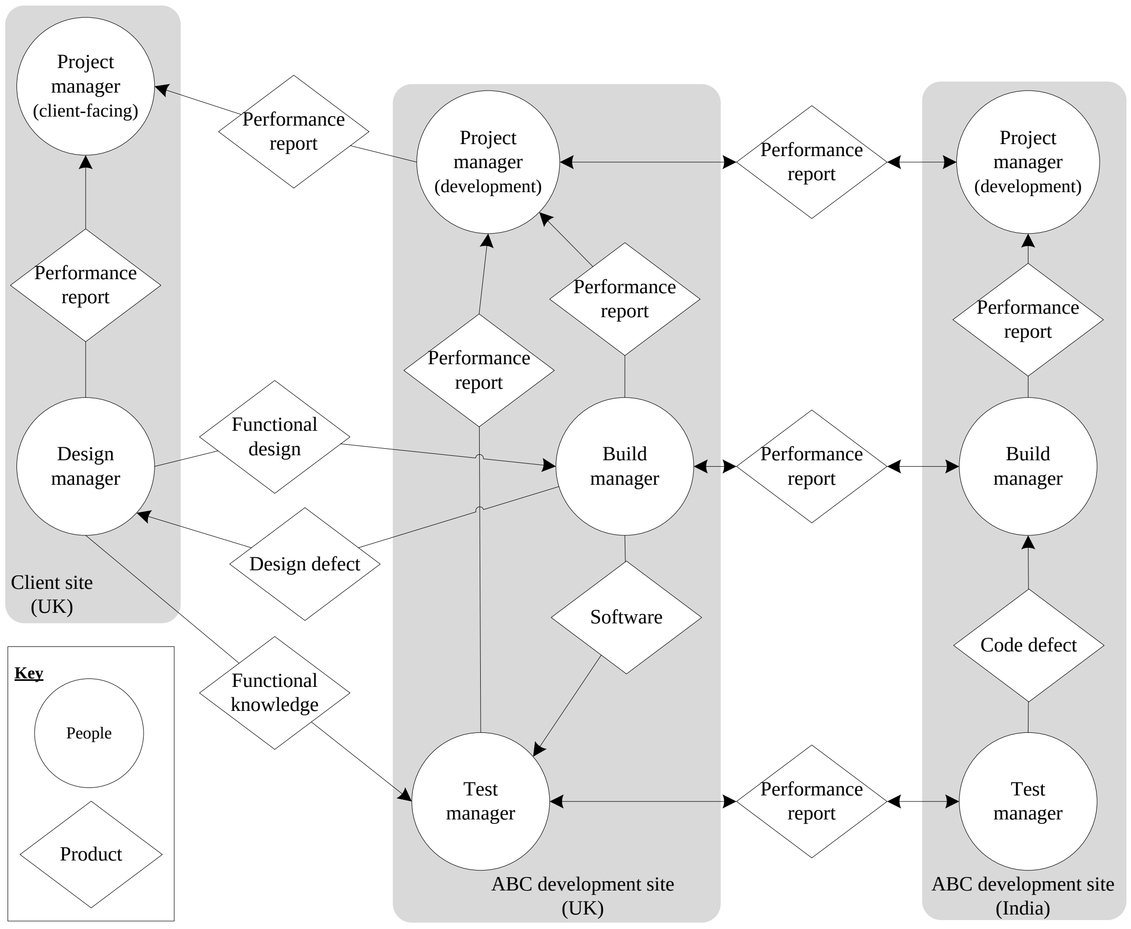


**Figure 3.2 ABC global software development**

Coordination of work activities took place within a location and across locations. Within a location, the Design, Build, and Test (DBT) managers managed dependency among project activities with their corresponding teams and with the Project manager at that location. Across locations, the Build and Test managers of a location coordinated work dependencies with their corresponding peer in the other location, and with the Design manager when needed. Recall from chapter 2 (section 2.3.4) that coordination at distance become challenging in such environment, particularly when overlapping work hours among the sites is limited. The time difference between UK and India was five and half hours (9:00 in the UK is 14:30 in India), which made the overlapping work hours limited to only three and half hours - see shaded area in Figure 3.3.

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| India | 09:00 | 10:00 | 11:00 | 12:00 | 13:00 | 14:00 | 14:30 | 15:00 | 15:30 | 16:00 | 16:30 | 17:00 | 17:30 | 18:00 | | | | | | |
| UK | | | | | | | 09:00 | 09:30 | 10:00 | 10:30 | 11:00 | 11:30 | 12:00 | 12:30 | 13:00 | 14:00 | 15:00 | 16:00 | 17:00 | 18:00 |

**Figure 3.3 UK/India overlapping work hours**

Communication among the team members located at the same site was mainly carried out face-to-face. Communication between the teams located in the UK was carried out using email, instant messaging, and teleconferencing. Communication among the teams across countries was carried out using email, instant messaging, teleconferencing, and video conferencing. There was some travelling between the countries for limited periods of time to develop rapport among the teams. However, section 2.3.4 (Chapter 2) noted communication challenges in globally distributed teams; and this is no different in ABC.

Control of project progress was carried out through weekly tracking of project activities among the Project managers and the DBT managers across locations, as is detailed in section 3.4 in this chapter, using teleconferencing. Controlling the consistency of project artefacts across ABC sites were maintained through using standard configuration management tools such as (IBM’s Clear Case), while controlling dependency and the alignment of project artefacts across projects and suppliers were maintained through regular sharing of these artefacts across the projects (detailed in section 5.2.3 - Chapter 5).

It may be noted in Figure 3.2 that, each of the DBT teams operate as one team but distributed across different locations; for example, the functional design transitioned from the Design manager to the UK Build manager in Figure 3.2 was also transitioned to the India Build manager simultaneously over video conferencing. Similarly, the software transitioned from the UK Build manager to the UK Test manager also involves the India Build and Test managers - the diagram shows only one transaction to maintain simplicity. Table 3.3 provides definition of the people within the PD team that was shown in Figure 3.2.

| **People** | **Description** | **Location** |
|---|---|---|
| Project manager (client-facing) | Manages the overall project (works for ABC) and oversees the client relationship. Captures high level system requirements and manages the Design manager | Client site (UK) |
| Project manager (development) | Manage the overall software development process, overseeing the UK and India Build and Test managers to develop software. Both the UK and India development Project managers work for ABC and report performance of work progress to the client-facing Project manager. In practice, the UK Project manager develops the project plan and is responsible for the day to day project execution and managing the Build and Test managers (in both UK and India). In contrast, the India Project manager provides input to and agrees/disagrees with the project plan, and only intervenes in project execution when needed; for example, when the India Build and Test managers need management support at non-working UK time (e.g. the middle of the UK night) | ABC development site (UK)<br><br>ABC development site (India) |
| Design manager | Manages the Functional Design phase; producing detailed Functional designs, through their team, to be transitioned to the Build manager for coding. The Design manager also provides Functional knowledge of the designs to the Test manager. The Design manager reports performance of work progress to the client-facing Project manager | Client site (UK) |
| Build managers | Manage coding of the Functional designs into software components, through their team, to be delivered to the Test manager for testing. The Build manager also registers Design defects for the Design manager to resolve. Whilst the UK Build manager is largely focused on the management aspects of delivery such as tracking work progress, performance reports, and communication with Client, their India counterpart manages development work on the ground with the Build team. The Build managers, report performance | ABC development site (UK)<br><br>ABC development site (India) |

| People | Description | Location |
|---|---|---|
| | of work progress to the development Project manager | |
| Test managers | Manages the Test phase and is responsible for producing Test products, and testing the code through their team. The Test manager also registers Code defects to be resolved by the Build manager. Whilst the UK Test manager is largely focused on the management aspects of delivery such as tracking work progress, performance reports, and communication with Client, their India counterpart manages Test activities on the ground with the Test team. The Test managers report performance of work progress to the development Project manager. | ABC development site (UK)<br><br>ABC development site (India) |

**Table 3.3 Definition of people within PD**

### 3.2.3 Iterative and incremental development at ABC

ABC employed a customised version of the Iterative and Incremental Development method (IID), where a project consisted of multiple increments - an increment represents one development cycle comprising the Design, Build, and Test (DBT) stages, which deliver a portion of the software functionality - see Figure 3.4.

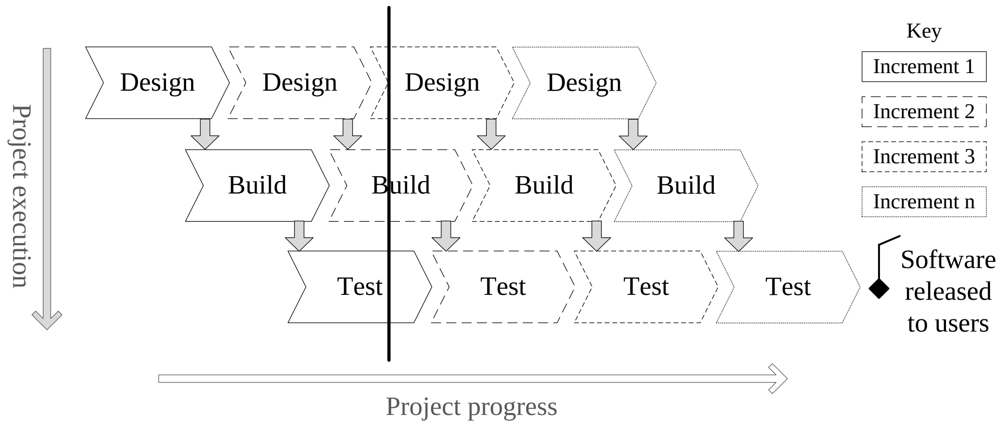
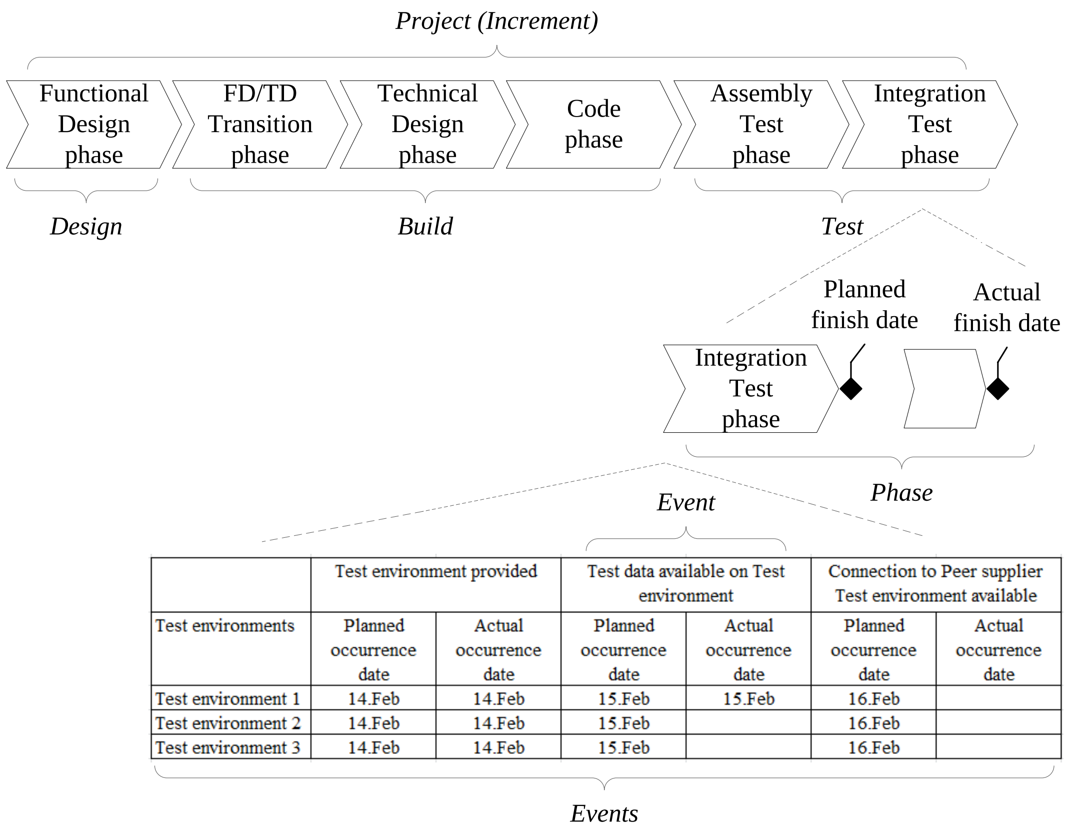


**Figure 3.4 ABC's version of Iterative and Incremental Development**

The vertical arrow 'Project execution' shows the increment cycle, where workflow moves across phases; from design to build to test. The horizontal arrow 'Project progress' shows workflow moving to the next increment of functionality within a particular phase; for example, following completion of design work in increment 1; the team started to design increment 2 of the functionality and so on. The same principle applied to the build and test phases.

A problem controlling this version of IID (i.e. semi-parallel execution of increments) is that at any one time, the functional project teams will be working on several increments of the same project, which may influence the commitment each phase may be expected to have. For example, the Test stage in increment 1 may require fixes of the code developed by the Build stage in Increment 1, however, at the time of executing the Test stage in Increment 1 (see the solid vertical line cutting through the phases), the Build resources are working on building Increment 2 of the functionality and the Design resources on designing Increment 3 (their respective primary focus) which may lead to misalignment of objectives due to resource clashes that are physical constraints.

Another problem controlling this version of IID is the management of change required on multiple versions of a component in multiple increments so that they inter-operate correctly; for example, a change request implemented on component *a* in increment 1 has to be replicated to the component in increment 2; similarly, changes in component *b* in increment 2 need to be replicated on the component in increment 1. A project in ABC comprises the Design, Build, and Test stages. Each stage comprises phases - see Figure 3.5.



| | Test environment provided | | Test data available on Test environment | | Connection to Peer supplier Test environment available | |
|---|---|---|---|---|---|---|
| Test environments | Planned occurrence date | Actual occurrence date | Planned occurrence date | Actual occurrence date | Planned occurrence date | Actual occurrence date |
| Test environment 1 | 14.Feb | 14.Feb | 15.Feb | 15.Feb | 16.Feb | |
| Test environment 2 | 14.Feb | 14.Feb | 15.Feb | | 16.Feb | |
| Test environment 3 | 14.Feb | 14.Feb | 15.Feb | | 16.Feb | |

**Figure 3.5 ABC's Project composition**

Figure 3.5 shows that an increment encompasses the following phases: FD, FD/TD Transition, TD, Code, AT, and IT (Table 3.4 describes the phases). A phase contains events, each of which has two attributes: 'Planned occurrence date' and 'Actual occurrence date'. An event is synonymous to a flag/status to mark the completion of an activity or phase in contrast to milestone, which usually refers to the end of phases where there is higher level management scrutiny to see if the products of the phase are acceptable and to authorise work to start on the next phase.

| **Phase** | **Phase acronym** | **Description** | **Number of people** | **Planned duration (weeks)** | **Location** |
|---|---|---|---|---|---|
| Functional Design | FD | Involves documenting detailed designs of how the future software would be required to operate in an FD document. This was produced by the ABC's Design team with considerable input from the Client. In addition to functional knowledge, the Client had some technical expertise in the existing systems | 3-6 | 4 | Client site (UK) |
| FD/TD Transition | FD/TD Transition | The FD team (UK) handed over the FD document to the TD team (India), which involved walking through the FD document to ensure that the TD team are clear of the design statements made in the FD documents | | 1 | ABC development site (UK); ABC development site (India) |
| Technical Design | TD | The functional designs within the FD documents were converted to technical designs within TD documents | 3-5 | 2 | ABC development site (India) |
| Code | Code | Involved coding the application components, including the necessary interfaces required to interoperate with various enterprise system components. Code also involved Unit Test (or Component Test) which verified whether the developed code matched the TD document. UT involves testing the service and process components (see Figure 3.1 in section 3.2) within their corresponding layers. Stub components which replicated the behaviour of the | 12-16 | 4 | ABC development site (India) |

| **Phase** | **Phase acronym** | **Description** | **Number of people** | **Planned duration (weeks)** | **Location** |
|---|---|---|---|---|---|
| | | Authentication and Component catalogue tools took the place of the actual tools | | | |
| Assembly Test | AT | AT was managed in two distinct phases (i) AT Plan and Preparation (ii) AT Execution where actual verification of the developed software was carried out. AT is similar to UT in that, it involves testing the service and process components (see Figure 3.1 in section 3.2) within their corresponding layers against stub components that replicate the behaviour of the Authentication and Component catalogue tools. However, AT also tests the Service and Process components with the Legacy components developed by the Peer supplier | 4-5 | Plan and preparation 4<br><br>Execution 4 | ABC development site (India) |
| Integration Test | IT | IT means testing the developed components across application layers and across suppliers to ascertain the workability of the overall enterprise system. IT was managed in two distinct phases (i) IT Plan and Preparation (ii) IT Execution where actual verification of the developed software was carried out. IT is similar to AT in that it involves testing the service and process components (see Figure 3.1 in section 3.2) with the Legacy components developed by the Peer | 6-8 | Plan and preparation 4<br><br>Execution 4 | ABC development site (India) |

| **Phase** | **Phase acronym** | **Description** | **Number of people** | **Planned duration (weeks)** | **Location** |
|---|---|---|---|---|---|
| | | supplier. However, IT differs from AT in that, it also tests the Service and Process components with the real Authentication and Component catalogue tools rather than stubs | | | |

**Table 3.4 ABC project phases**

## 3.3 Managing project execution

ABC projects adopted a Component-based development approach (see section 2.4.4 - Chapter 2). The projects investigated by this research developed enterprise middleware systems rather than conventional applications, which adds a lot of complexity to the development effort (Hustad & Lange, 2014) since it requires interaction and embedding with existing products and an increased number of stakeholders as was seen in Figure 3.1 (section 3.2.1). In this model, three spheres can be distinguished - Client, Peer supplier, and ABC - within each of which are further sub-processes - see Figure 3.6.

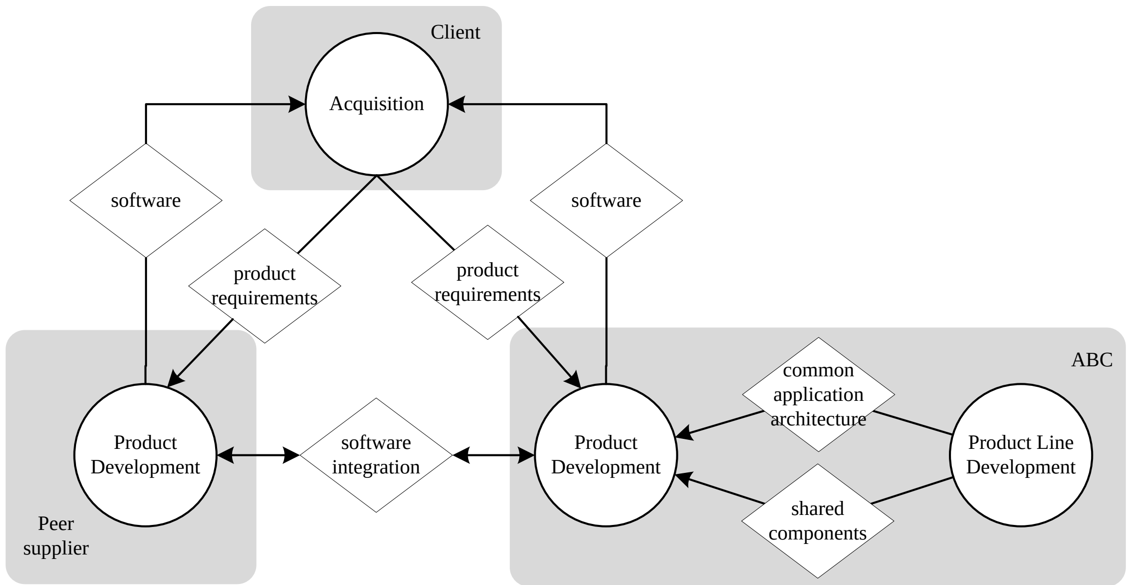


**Figure 3.6 ABC delivery model**

The Client’s Acquisition process provided product requirements to the Peer supplier and ABC. The Peer supplier’s Product Development process supplied software to the Client and liaised with ABC to ensure product integration since they both delivered parts of the overall system. The ABC’s Product Development (PD) process supplied software to the Client, utilising the ABC’s common application architecture and shared components, which was built by the Product Line Development process (PLD). The PLD developed reusable components and managed the enterprise software system.

A detailed view of the ABC development activities is presented in sections 3.3.1 - 3.3.7 (the inner workings of PD), sections 3.3.9 and 3.3.11 (the workings of PLD), and sections 3.3.8 and 3.3.10 (the interaction among PD and PLD). These process models represent the underlying physical system and show the interaction among various project participants during project execution. This context was drawn from the contents of the project performance reports, which were the source of the research data used to track progress of phase execution (see section 3.4).

Recall from section 3.2.1 that, two types of software were developed; service components and process components. The service components are all custom components specifically developed to meet the client's needs; while some of the process components are custom and others are modifications of package software (Summerville, 2011, pages: 6-7) obtained from external providers. The development process of both the service and process components took the incremental approach described earlier in Figure 3.4 (section 3.2.3). The configuration of the package software involved performing gap analysis of the offered functionality by the package product against the client needs prior to project execution and configuring the features of the package software to enable integration with existing systems.

The preceding section (Table 3.4) introduced the phases; the following sub-sections get into the details of the events occur during the execution of the phases. For description of the functions and products appear in these sub-sections, see Tables 3.6 and 3.7 respectively at the end.

As will be seen throughout these sub-sections that, each phase was dependent on the products produced, and/or services provided, by other phases or functions external to the PD, in order to carry out and progress the activities comprising the phase. The complexity of such interactions and interdependencies were noted in Chapter 2 (sections 2.4.1, 2.4.3, 2.3.3, and 2.3.4), and the ability to see the physical details at the level provided here gives a clearer idea of what is going on, which helps in putting the analyses we conducted in the subsequent chapters into the context presented here.

### 3.3.1 Functional Design

The key events occur in producing FD document is shown in Figure 3.7.

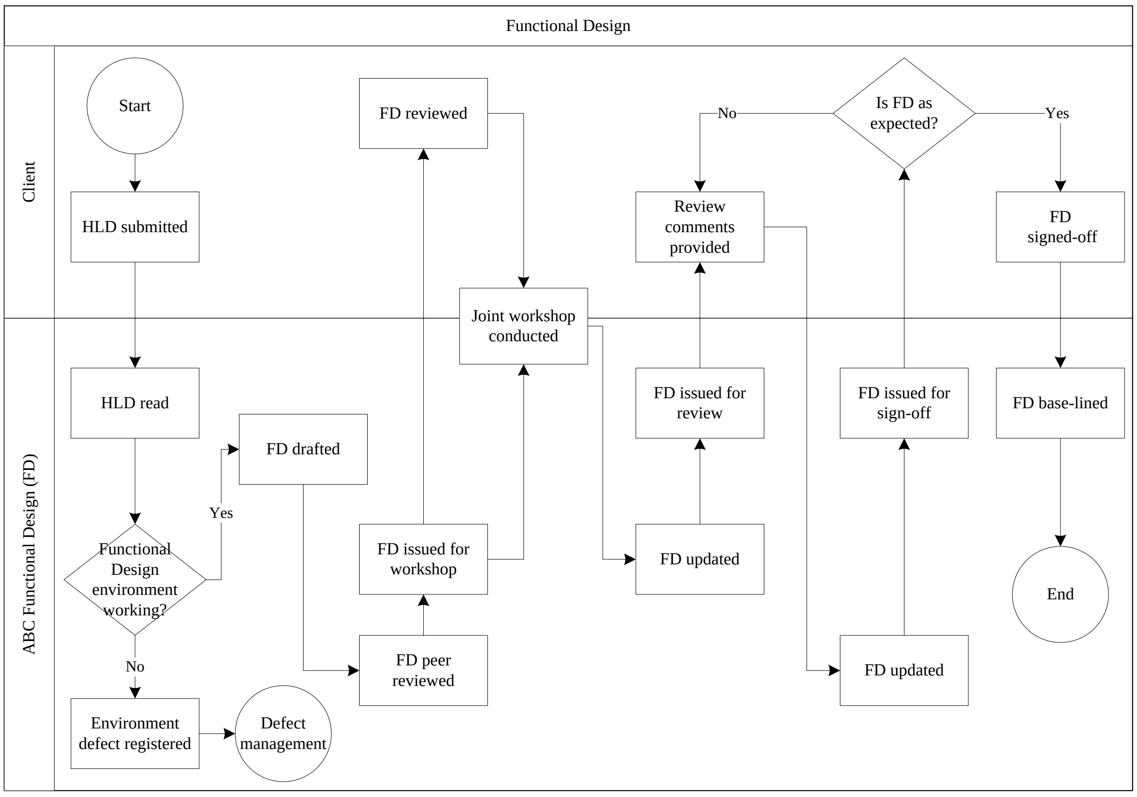


**Figure 3.7 Functional Design event sequence**

At the start of the phase the FD team, under the leadership of the Design Manager, read the HLD (High Level Design) and the Design environment is checked for workability; if it is not working, an Environment defect is registered on the Defect management system (see section 3.3.8), otherwise, the FD documents are drafted, peer reviewed and issued for Client review prior to the joint workshop (face-to-face) to discuss and agree the details of the design. Following the workshop, the FD document is updated to reflect the requirements captured in the workshop and issued for Client review. The Client feedbacks are incorporated and the FD issued for sign-off. If the designs do not reflect the Clients requirements at this stage, appropriate changes are made upon feedback, and the FD is reissued for sign-off. This cycle continues until the FD is to the satisfaction of the Client, in which case it is signed-off and base-lined. Changes on the designs after this point would have to be made according to the Change Control procedure (section 3.3.11). The Design manager reported progress of work activities (performance report) weekly to the Project manager.

### 3.3.2 FD/TD Transition

The key events occur during the FD/TD Transition is shown in Figure 3.8.

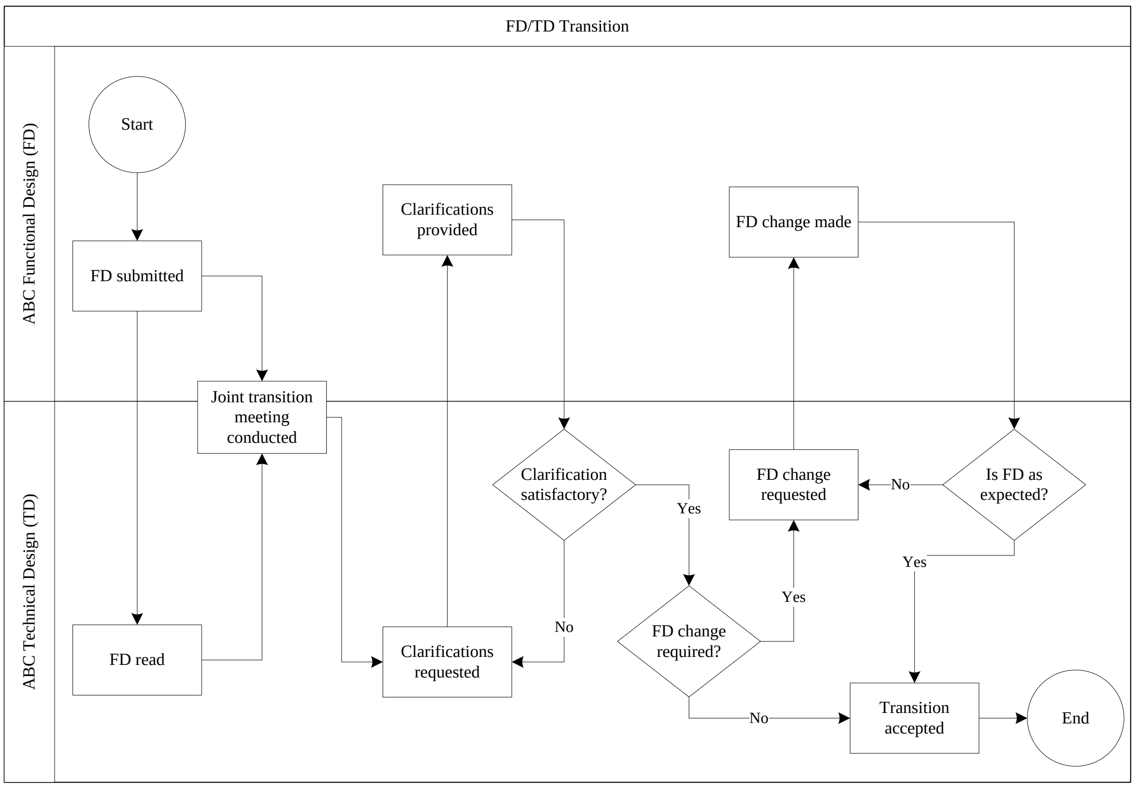


**Figure 3.8 FD/TD Transition event sequence**

The FD/TD Transition phase starts typically with the signed-off FD document being submitted by the FD team (located in the UK) to the TD team (located in India), who after reading the FD convene with the FD team in joint transition meeting, typically over video conference facilities; walking through the FD documents to ensure that the TD team are clear of the design statements made in the FD documents. Following the meeting, the TD team may seek further clarifications, which would be provided. If the clarifications are not satisfactory, the request for clarification goes around until they meet the satisfaction of the TD team. If no further FD changes are required as a result of these clarifications, FD transition is accepted, the decisions made are recorded in an FD/TD Transition document, and the phase ends. All FD changes are subjected to the Change control procedure (section 3.3.11). The Build manager reported progress of work activities weekly to the Project manager.

Section 3.2.2 noted communication challenges across sites, which can be seen here when the FD team attempt to explain the functional design statements (based on the context of business functions) to the TD team (who lack that context due to lack of training) through video screens, and the email follow ups to clarify specific points in order to bring the FDs to the satisfaction of the TD team. These challenges may be further exacerbated by the, limited, 3.5 overlapping work hours across the sites (see Figure 3.3).

### 3.3.3 Technical Design

The key events occur in producing TD document is shown in Figure 3.9.

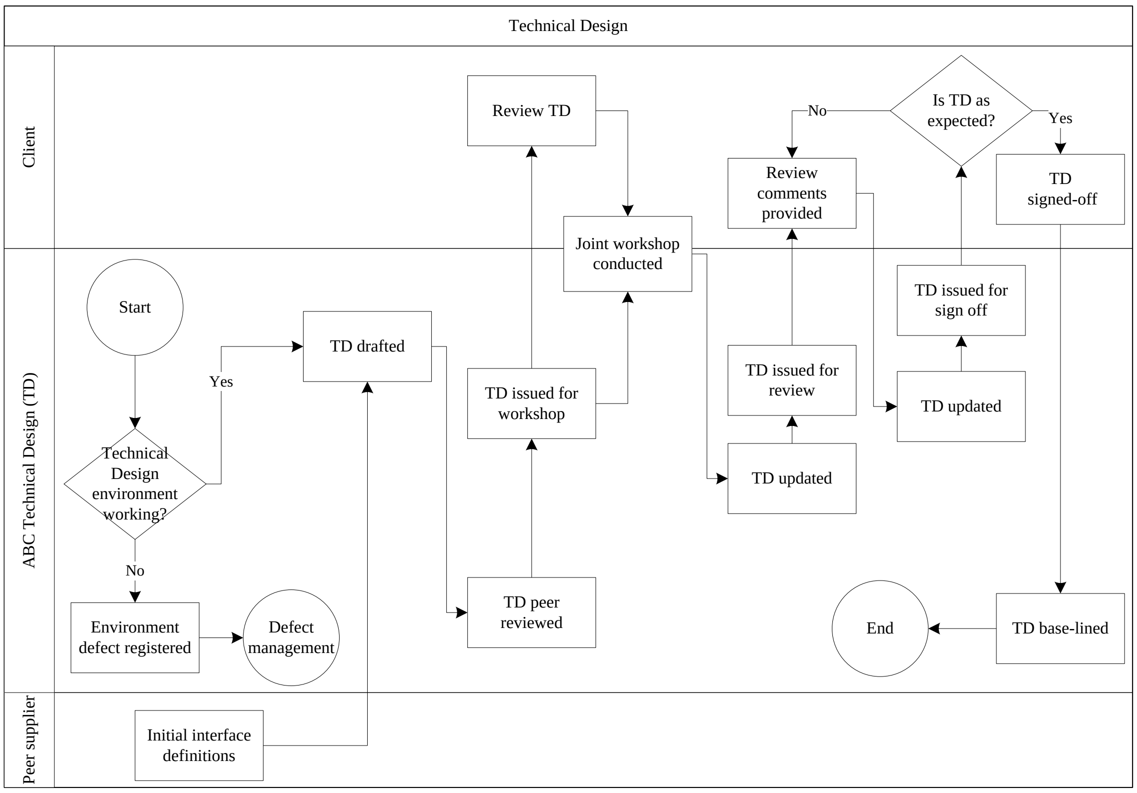


**Figure 3.9 Technical Design event sequence**

At the start of the TD phase, the Design environment is checked for workability, if it is not working, an Environment defect is registered on the Defect management system (section 3.3.8); otherwise, the TD document is drafted. If a TD is drafted based on a yet-to-be signed-off FD, rework might be required. At the time of drafting the TD the initial interface definitions are received from Peer supplier, which will be used to align the TD with the Peer supplier components to ensure integration across suppliers. The TD is peer reviewed, and issued for Client review prior to the joint workshop (via video conference), where the details are discussed and agreed upon. Technical oriented Client representatives from the same Client group attended the FD workshop will be reviewing and attending the TD workshop (no FD representatives would be present).

Following the workshop, the TD is updated and issued for review, when further comments may be provided, and the TD is further updated and issued for sign-off. If the TD is not to the satisfaction of the Client, for example their comment is not incorporated accurately, the review and update cycles continue until agreement is reached; at which point the TD is signed off and base-lined. It is possible that as a result of this process the FD could be amended. The Build manager reported progress of work activities weekly to the Project manager.

### 3.3.4 Code

The key events occur during the Code phase is shown in Figure 3.10.

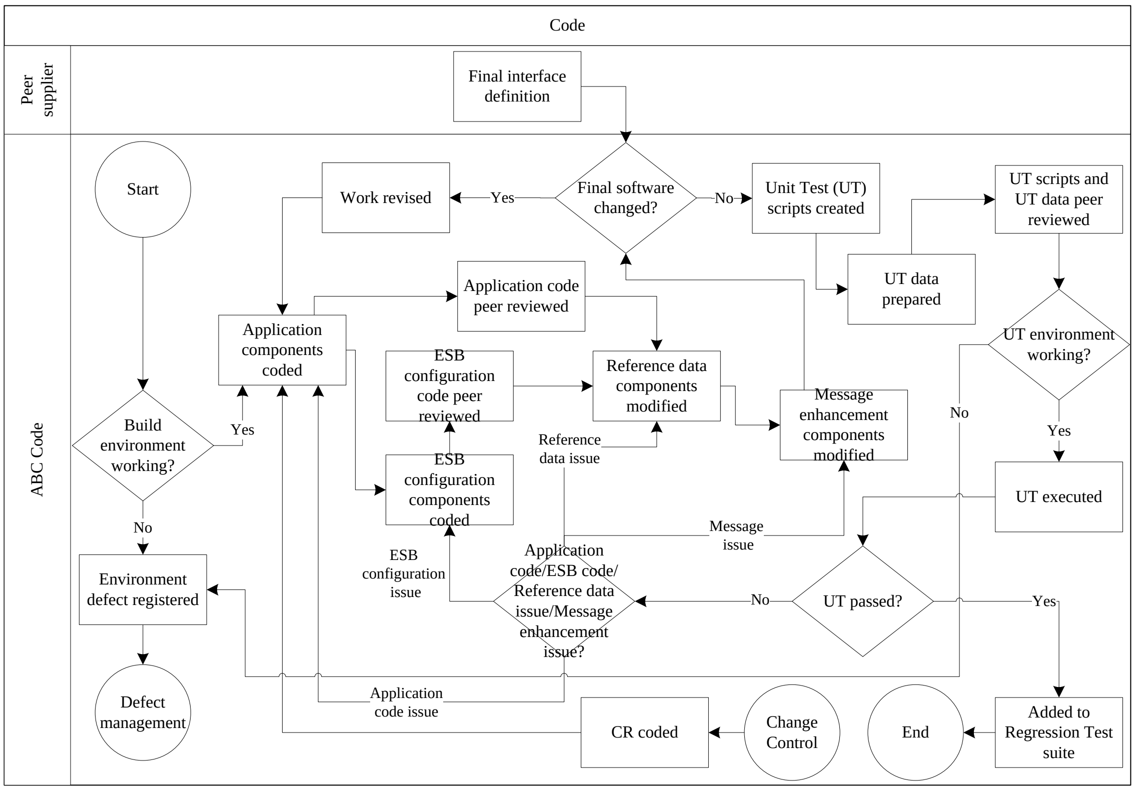

Code
Peer supplier
ABC Code
Final interface definition
Start
Work revised
Yes
Final software changed?
No
Unit Test (UT) scripts created
UT scripts and UT data peer reviewed
UT data prepared
Application code peer reviewed
Application components coded
ESB configuration code peer reviewed
Reference data components modified
Message enhancement components modified
UT environment working?
No
Yes
Build environment working?
Yes
Reference data issue
ESB configuration components coded
No
Environment defect registered
ESB configuration issue
Application code/ESB code/ Reference data issue/Message enhancement issue?
Message issue
No
UT passed?
Yes
UT executed
Defect management
Application code issue
CR coded
Change Control
End
Added to Regression Test suite

**Figure 3.10 Code event sequence**

At the start of Code activities, the Build environment is checked for workability; if it is not working, an Environment defect is registered on the Defect management system (section 3.3.8) for the Technical environment manager to resolve, otherwise coding activities commence.

Developing application software to integrate with the existing enterprise system in ABC, involved: (i) coding the application components (ii) coding the ESB (Enterprise Service Bus) configuration components (iii) coding reference data components (iv) coding message enhancement components. Coding the application components (Service and Process) involved, converting the TD to software components. Coding the ESB configuration components involved enabling the application components to communicate with the ESB (see Figure 3.1 - section 3.2.1). The codes in the application and ESB configuration components were then peer reviewed. This was followed by modifying the enterprise reference data to include the newly required items (e.g. for a data item with code 'UoB' might be decoded as 'University of Brighton'), and modifying the enterprise message enhancement components to include the response messages specified by the Client (e.g. an error message automatically output from the system might be 'ISM09873645' and generate an 'enhanced message' that the user can understand such as 'Name must be specified to access the individual's record').

Towards the end of the phase, the Peer supplier releases to the Build team their final interface definitions to align artefacts for integration. These were compared to the initial versions provided during the TD phase. If changes were discovered, the ABC Code was revised to integrate the two sets of software. Otherwise, work started on Unit Test (UT) scripts. Unit Test (or Component Test) verified whether the developed code matched the TD document. Unit Test in ABC is part of the Code activities performed by the Build team.

Developers then prepared UT data replicating the data in the operational environment. UT scripts (this was recorded on the UT script document that records the test conditions and expected/actual results) and Test data were then peer reviewed, UT environment checked for availability, and UT scripts were executed with the stub components if the UT environment was operational, otherwise environment defect was registered.

After successful unit testing, software components were added to the Regression Test suite for future Regression testing. If defects were detected, the four areas of component development were analysed to ascertain the location of the defect, and the coding cycle was repeated. Finally, during or towards the end of the Code phase, Change requests (CR) might arise. The request for coding a CR comes from the Change Control function (section 3.3.11), and results in those changes to be incorporated in the current development activity. The Build manager reported progress of work activities weekly to the Project manager.

### 3.3.5 Assembly Test Plan and Preparation

Figure 3.11 shows the events of AT Plan and Preparation phase, where some activities produced and obtained approval for the Test approach, and other activities arranged for Test environment setup and Test data preparation needed for Test execution.

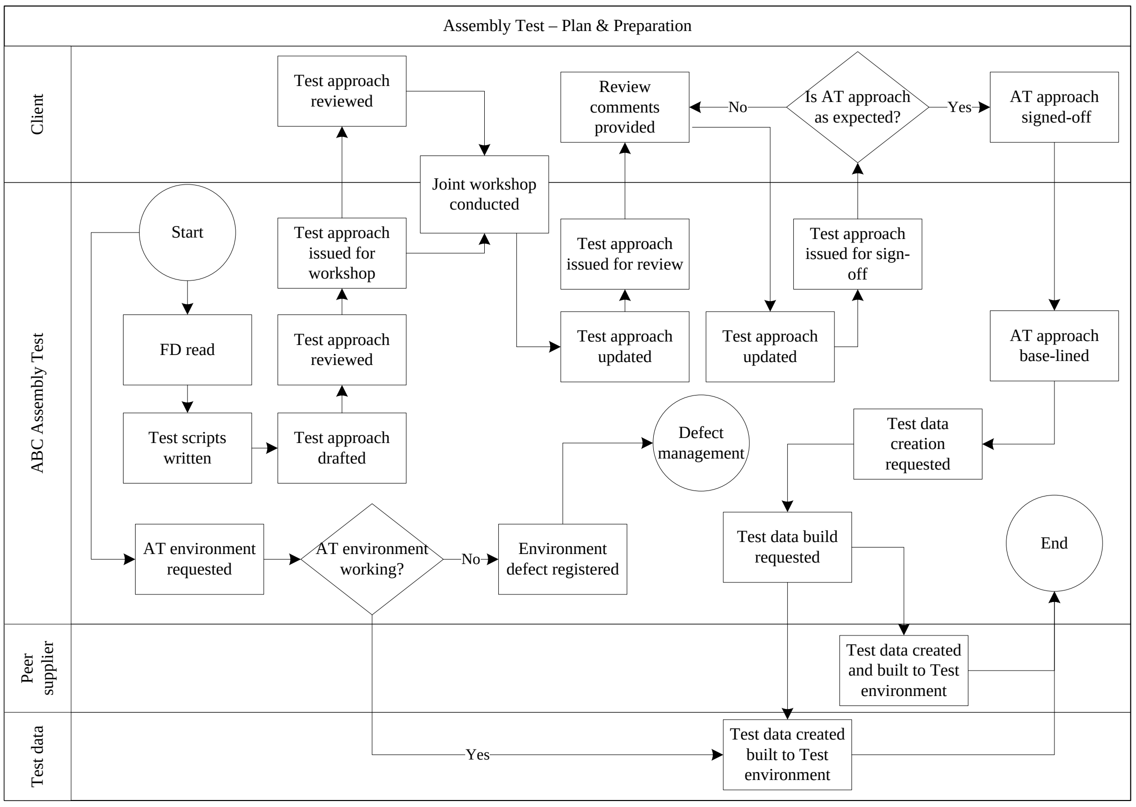


**Figure 3.11 AT Plan and Preparation event sequence**

The AT Test scripts were written based on the FD documents. The Test approach was then drafted, reviewed by Project manager, and issued for Client review prior to a joint workshop, where the content of the document was discussed in detail and agreed. The Test approach was updated to reflect the meeting outcome, and issued for Client's review. The document was then updated with comments provided, and issued for sign-off. If the document was not to the Client's expectation, for example a Test scenario does not accurately reflect how the system would be used to carry out business functions, it would go through the review and update cycle until it is satisfactory, at which point the approach was signed-off and base-lined.

An AT Test environment was requested, and if it was working, requests for Test data creation (based on the AT approach) and build to the Test environment are made. Otherwise an environment defect was registered. The Test data manager then created Test data and built it to the ABC Test environment, in coordination with the Technical environment manager. The Peer supplier also created Test data and built it to their own Test environment. Test data was typically copied from operational data, so that testing can replicate the operational environment as closely as possible. The Test manager reported progress of work activities weekly to the Project manager.

### 3.3.6 Assembly Test Execution

The key activities of the Assembly Test Execution phase are shown in Figure 3.12.

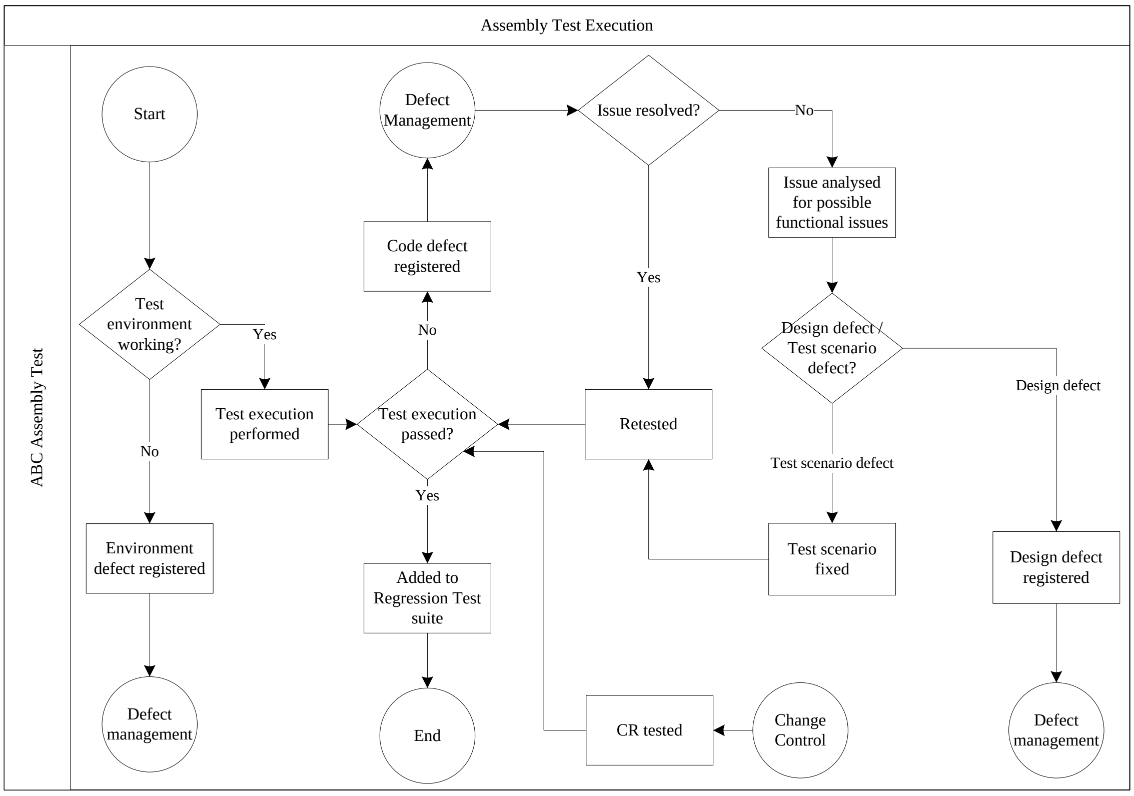


**Figure 3.12 Assembly Test Execution event sequence**

At the start, the AT Test environment is checked for workability, if not working, an Environment defect is registered on the Defect management system (section 3.3.8) for the Technical environment manager to fix; otherwise Test execution activities commence. Test execution involves the manual running of a program - XMLSpy - that runs the developed components in the right message routing order through the different application layers (as described in Table 3.2 - section 3.2). If the test is successful, it is added to the Regression Test suite to enable future automatic retesting. Otherwise, Code defects are registered for the Build manager to fix (Defect Management - section 3.3.8). Defect correction and retesting is repeated until the component passes testing.

However, if the defect was not a Code issue (i.e. the Code was implemented according to the Technical/Functional designs), then the defect is analysed (by the Test manager/team) for possible functional issues. For example, the Design might be flawed, or the Test scenario might be misunderstood or vague and thus require clarification. If it is a Test script defect, the defect is fixed and the component is retested, otherwise a Design defect is registered for the Solution architecture manager to fix (Defect Management - section 3.3.8). It is possible that during the AT Execution, need arises to test coded Change requests. Changes made to requirements during testing require change to the Design documents, change to the Code, and the changed Code to be tested. The Test manager reported progress of work activities weekly to the Project manager. At the end of the Test execution phase, the Test manager submitted a Test completion report to the Client for approval.

### 3.3.7 Integration Test

The events that take place in the plan and preparation activities for IT are the same as the AT Plan and Preparation phase described in section 3.3.5, the difference would be in the type of data and functional scenarios that would be prepared for testing. Thus, with exception of replacing the term AT in section 3.3.5 with the term IT (integration test), there is no value in repeating the details here. Similarly, the events occur during IT execution are the same as the ones occur in AT execution (described in section 3.3.6), the difference would be in testing the developed code with the Authentication and Component catalogue tools as stated in Table 3.4 (section 3.2.3) and the creation of a daily report of test execution results sent from the Test manager to the Client detailing the scenarios passed/failed and blockers to progress. The remainder of the process is the same as for AT. The IT team are the same staff who carried out the AT execution. The Test manager reported progress of work activities weekly to the Project manager. The Test manager submitted a Test completion report to the Client at the end of the Test phase.

### 3.3.8 Defect management

Defect management include registering, prioritising, and resolving defects, using the IBM’s Clear Quest tool, which can be accessed by the teams in the UK and India. Registered defects were given priority rating (i.e. the ‘severity’ which indicates level of urgency for resolution) within the tool; the relative priority of defects with the same severity level was established outside the tool; an extract (spreadsheet) of active defects is taken at the end of the day (UK time) and circulated for prioritisation on the ‘Daily defect call’.

The daily defect call was a daily teleconference meeting took place at 9.15am UK time (see Figure 3.3 - section 3.2.2) to consider and prioritise defect reports of code and technical environment and design queries. Hence, this meeting is attended by the Technical environment manager (UK), Fix manager (one in the UK, another in India), and the Solution architecture manager (UK); and were then dealt with through the processes described in sections 3.3.9, 3.3.10, and 3.3.11 respectively. Unresolved defects appear on the next extract (spreadsheet) of active defects, considered at the next Daily defect call.

Section 3.2.2 noted potential challenges in coordinating activities across sites; for example, a test environment defect registered 9:00 India time (see Figure 3.3) which may be obstructing execution of Test scripts, means that the Test team in India have to wait until 14:30 (9am UK time) for the UK based Technical environment team to start looking at the issue; i.e. more than half a working day is already lost in India at least.

### 3.3.9 Technical Environment

Located in the UK, the key events occur within the Technical environment function is shown in Figure 3.13.

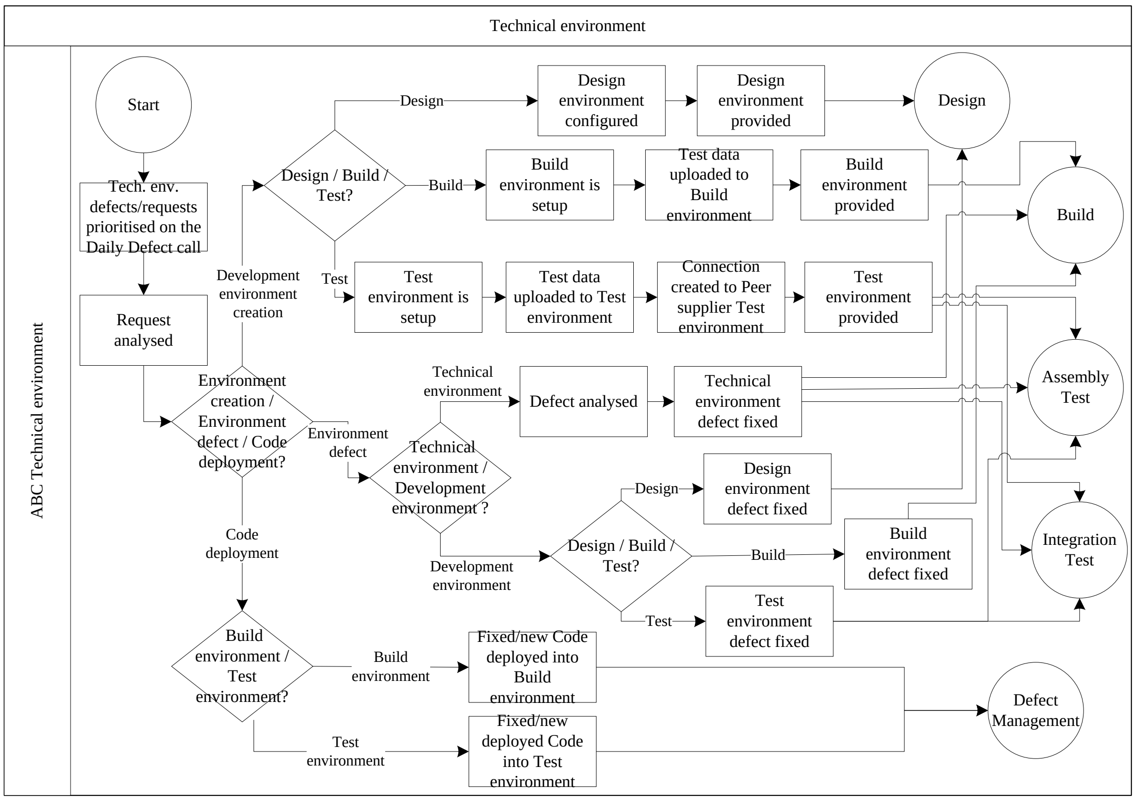

Technical environment
ABC Technical environment
Start
Tech. env. defects/requests prioritised on the Daily Defect call
Request analysed
Development environment creation
Design / Build / Test?
Design
Design environment configured
Design environment provided
Design
Build
Build environment is setup
Test data uploaded to Build environment
Build environment provided
Build
Test
Test environment is setup
Test data uploaded to Test environment
Connection created to Peer supplier Test environment
Test environment provided
Assembly Test
Environment creation / Environment defect / Code deployment?
Environment defect
Technical environment
Defect analysed
Technical environment defect fixed
Technical environment / Development environment ?
Development environment
Design / Build / Test?
Design environment defect fixed
Build environment defect fixed
Test environment defect fixed
Integration Test
Code deployment
Build environment / Test environment?
Build environment
Fixed/new Code deployed into Build environment
Test environment
Fixed/new deployed Code into Test environment
Defect Management

**Figure 3.13 Technical environment event sequence**

Three types of environment defects/requests arise from the prioritised issues on the Daily defect call. Firstly, request for setting up a development (Design, Build, or Test) environment. Setting up the Test environment includes building test data on the environment and establishing connection with the Peer supplier’s test environment for integration testing.

Secondly, request to fix environment defects which could be defects in the development environments or defects in the Technical environment. Defects in the development environment may need coordination with the relevant teams. For example, defects in the Test environment may be due to inconsistency in the Test data specified by the Test team. However, coordinating this activity may be challenging across sites (see section 3.2.2) due to the limited overlapping work hours (3.5) across sites.

Thirdly, request from the Build manager to deploy a build (code) into the Build and/or Test environments at the end of India day (12:30 UK time - see Figure 3.3). This means that if the build lacked certain packages needed to complete the deployment, the Technical environment manager has to wait until next day 14:30 India time (9:00 UK) to get the relevant packages and restart the Code deploy process; i.e. more than half a day lost in India waiting for the code availability. The Technical environment manager reported progress of work activates weekly to the PD Project manager and to the management of the Technical environment function.

### 3.3.10 Fix management

Fix management refers to fixing defects arising from testing the developed Code; it is carried out by the Fix team, who form a subset of the Build team that developed the components and have reasonable knowledge of them - see Figure 3.14.

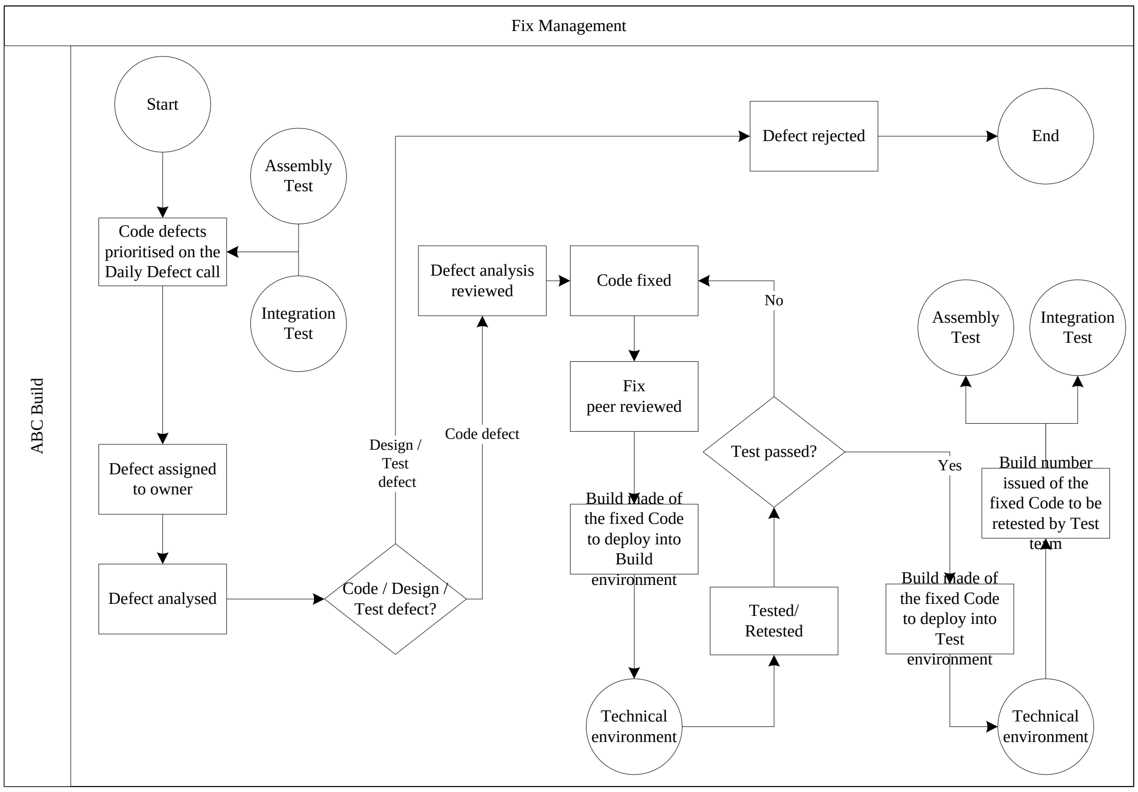


**Figure 3.14 Fix Management event sequence**

The prioritised code defects from the Daily defect call are analysed, and a defect is rejected if it is not considered a code defect as the Code correctly implements the Technical designs; i.e. treated as design or test script defects. The Client only intervenes in this process if change in the requirements/design is needed. The analysis of the Code defect is reviewed by the Fix manager to ensure that this is truly is a code defect, then the defect will be fixed, peer reviewed, and a build is made of the fixed code and deployed into the Build environment for testing, using the Code deploy tool within the

Technical environment function. The fixed code is then tested by the Fix team, and if it fails, it goes through the cycle of fixing. When the code passes retest a build is made and deployed into the Test environment using the Code deploy tool. Finally, the build number of the fixed code is issued to the Test team to be used for actual AT or IT Test execution. The Build manager reported progress of work activities weekly to the Project manager.

As the fix activities are carried out in India and deploying code fix is carried out using the Code deploy tool which is managed by the UK team, coordination challenges can be amplified by time zone difference when the code deploy process fails to complete successfully. For example, the India team typically deployed fixes (it was possible to trigger the tool from India) at the end of India day (see Figure 3.3) for the code to be ready for testing the following day. However, if the tool fails to complete the process overnight, the India team has to wait until 14:30 the day after for the Technical environment team to first look at the issue at 9:00 UK time; i.e. more than half a day is lost in India - Gorton & Motwani (1996) noted that time zone differences can hinder progress, if not managed correctly.

### 3.3.11 Change Control

The Solution architecture function within ABC is located in the UK and is responsible for controlling changes to software under development during project execution. Each project in ABC has its own dedicated Solution architecture team that manages change control procedures and ensures the sound application architecture of the software. The Solution architecture team take ownership of the FD documents straight after sign-off. Changes required to the software under development, including Design defects corrections are managed through a formal Change Control procedure - see Figure 3.15.

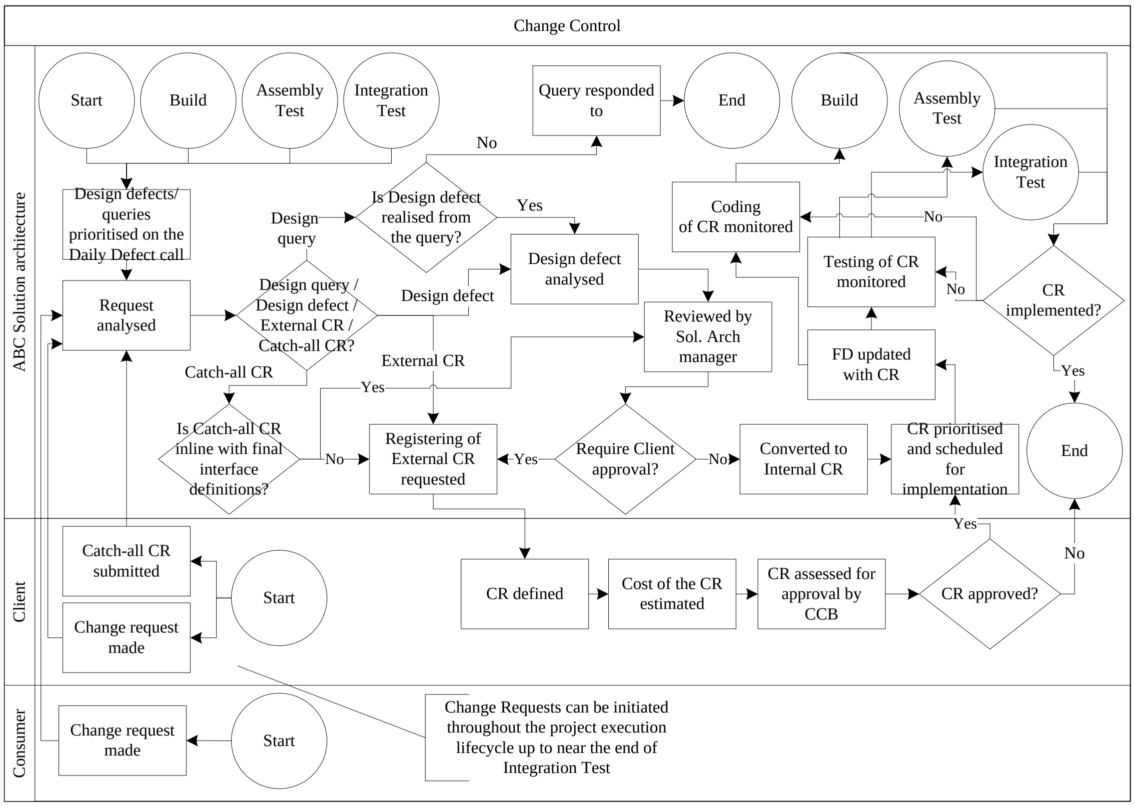

Change Control
ABC Solution architecture
Client
Consumer
Start
Build
Assembly Test
Integration Test
Design defects/ queries prioritised on the Daily Defect call
Request analysed
Design query
Is Design defect realised from the query?
No
Yes
Query responded to
End
Design query / Design defect / External CR / Catch-all CR?
Design defect
Design defect analysed
External CR
Catch-all CR
Is Catch-all CR inline with final interface definitions?
Registering of External CR requested
Require Client approval?
Reviewed by Sol. Arch manager
Converted to Internal CR
CR prioritised and scheduled for implementation
Coding of CR monitored
Testing of CR monitored
FD updated with CR
CR implemented?
Catch-all CR submitted
Change request made
CR defined
Cost of the CR estimated
CR assessed for approval by CCB
CR approved?
Change Requests can be initiated throughout the project execution lifecycle up to near the end of Integration Test

**Figure 3.15 Change Control event sequence**

Change request is essentially a request for changing the software during project execution. There are five types of change request, and sources may vary - see Table 3.5. Design queries my result in the identification of design defects, and if they do not require Client approval for the change, because for example the design does not conform to the original requirements, it triggers creation of internal CR which will be prioritised and scheduled for implementation (i.e. fixed in the relevant FD, coded, and tested). If Client approval is required, it triggers creation of external CR, which requires going through the formal change control board (CCB) procedure from the Client side to authorise budget - before the CR is scheduled for implementation. A catch-all CR submitted by the Client and which does not conform to the final interface definitions triggers the creation of external CR because it incurs a cost.

The process of CR implementation involves agreeing with the Build and Test managers a timeline, within the existing project schedule, to implement the changes. The Solution architecture team update the FD documents with the changes (as owner of the base-lined FDs), monitors/track progress of implementing the changes in the Build and Test phases, and any delays are reported by the Solution architecture manager to the UK

Project manager in the weekly project performance meetings. Coordinating the effort of coding and testing of the CRs across sites (the Solution architecture manager located in the UK, and each of the Build and Test managers located in India) would be difficult due to communication challenges and limited overlapping work hours as was noted in section 3.2.2.

| Type of change | Description | Possible source |
|---|---|---|
| Design query | Question about a design statement in the FD documents seeking clarification on the intention or interpretation of the requirements | Build manager;<br>Test manager;<br>Solution architecture manager |
| Design defect | Fault identified in the Functional design after the FD document was signed-off | Build manager;<br>Test manager;<br>Solution architecture manager |
| External CR | Change request initiated by the Client - external to the ABC's Product Development team; and/or change need to be made but which was not in the original requirements | Client<br>Consumer |
| Catch-all CR | Change request from the Client capturing all the changes already made to the Design and Build artefacts and agreed via email between the Client and ABC PD team, but which have yet to be approved by the Change control board (CCB) due to the time it takes to go through the CCB process. This is aimed at rebase-lining project artefacts to the latest changes | Client |
| Internal CR | Change request that does not require Client approval, and can be implemented prior to informing the Client of the change. This is because the change does not constitute actual modification of the original requirements | Solution architecture manager;<br>Build manager;<br>Test manager |

**Table 3.5 Type and source of change**

| Function | Description | Representative |
|---|---|---|
| Assembly test | As described in Table 3.4 (section 3.2.3) | Test manager |
| Build | As described in Table 3.4 (section 3.2.3) | Build manager |
| Client | Represents the Client's information system function, who commissioned the development of the software product | Client |
| Code | As described in Table 3.4 (section 3.2.3) | Build manager |
| Consumer | Represents the ABC projects that develop the consumer components in Figure 3.1 (section 3.2.1) | Consumer |
| Functional design | As described in Table 3.4 (section 3.2.3) | Design manager |
| Integration Test | As described in Table 3.4 (section 3.2.3) | Test manager |
| Peer supplier | Represents the Peer supplier's information system function, who provide software to the same Client as the ABC company; and whose software (legacy components in Figure 3.1) must integrate with ABC's software | Peer supplier |
| Solution architecture | Represents ABC's Solution architecture function that ensures sound application architecture and, through change control procedures, safeguards against arbitrary changes to the developed software detrimental to existing functionality | Solution architecture manager |
| Technical design | As described in Table 3.4 (section 3.2.3) | Build manager |
| Technical environment | Represents ABC's Technical environment, a supporting function that creates live-like systems' environments for various product development projects to use in their development work | Technical environment manager |
| Test data | Represents ABC's Test data management function that creates test data and arranges for uploading them on the Test environment (i.e. Test data build) for various product development projects | Test data manager |

**Table 3.6 ABC development functions**

| **Product** | **Description** |
|---|---|
| Build environment | The integrated development environment in which writing of the code is carried out |
| Code | The code developed by the Build manager and is being tested by the Test manager and their team |
| Code deploy tool | Custom developed software, used to automate the process of deploying the newly developed code (a Build) to the Test environment for testing. This tool has a web interface and is used by the Build manager remotely as well as the Technical environment manager (who originally developed and manages the tool), to perform code deploy activities |
| Defect | The malfunction of project artefacts; e.g. Design, Code, Code deploy tool, Technical environment, or Test script defects |
| Defect management system | The IBM’s Clear Quest tool used for registering, prioritising, and resolving defects |
| Development environment | A generic term referring to the Design, Build, or Test environment. The development environment is a combination of hardware and software dedicated to a particular project that supports Design, Build, or Test activities |
| FD/TD Transition document | A document created by the Build manager and maintained in collaboration with the Design manager to record key transition discussions/decisions made between the FD and TD teams and any changes that was made on the FD documents as part of completing the transition |
| Fix | The fix of product defects; e.g. fixing the Design, Code, Code deploy tool, Technical environment, and Test script defect |
| Functional design document | A document describing how the functional requirements are to operate when the software is built |
| Functional Design | A CASE (Computer-aided Software Engineering) tool used to generate the FD documents. The tool allows creation |

| **Product** | **Description** |
|---|---|
| environment | of TD documents from the FD documents following certain changes that need to be made by the TD team. The tool is typically expected to be established in around two weeks, necessary emails to request such environments go out prior to starting phase execution |
| HLD | An input to the FD phase produced by the Client; it stands for High level design and documents the Clients' requirements of the system to be developed |
| Interface definition | A machine-readable description of a service component of how it can be called, what parameters it expects, and what data structures it returns |
| Performance report | The progress status of executing planned activities produced by a phase manager and submitted to their management; e.g. the Design, Build, or Test activities produced by the corresponding phase manager to the Project manager |
| Regression Test tool | A custom developed tool used to automate the regression testing activities of the developed code, to ensure that subsequent fixes of the code has not broken the correct parts of the code |
| Technical Design document | A document describing how the software will be developed |
| Technical Design environment | Part of the same CASE tool that was used during the FD phase (see 'Functional Design environment') used to create the TD documents. The tool is expected to carry forward the FD details |
| Technical environment | The combination of hardware and software that integrates a particular development environment with the rest of the enterprise system with which the development effort needs to interact with during the development process, such as the Enterprise service bus |
| Test approach | A document produced at the beginning of the Test phase detailing what and how the testing would be carried out |

| **Product** | **Description** |
|---|---|
| Test completion report | A document produced at the end of the Test phase detailing what testing was carried out and what defects are carried over to the production environment |
| Test data | Data manufactured and used to perform testing of the developed code; the data is manipulated by the developed code to replicate the functioning of the live/production system. This include the Test data prepared by ABC’s Test team as well as the Test data prepared by the Peer supplier's Test team to allow for integration testing of the code developed by both ABC and Peer supplier |
| Test data build | The uploading of the manufactured Test data into the Test environment and executing some test scenarios to determine the correct behaviour of the Test data (pipe-clean) prior to the actual start of the Test execution phase. This include the Test data build by ABC’s Test team as well as the Test data build by the Peer supplier's Test team |
| Test environment | The hardware and software on which testing of the developed code will be carried out, and any other software with which the developed code interacts with during testing of the new and/or the fixed code. This includes UT environment, AT environment, and IT environment |
| Test scripts | Test conditions and expected results written to guide the Test execution activities; e.g. UT scripts, AT scripts, and IT scripts |

**Table 3.7 ABC development products**

## 3.4 Reporting project performance at ABC

This section describes the composition of ABC's project performance reports, because following the completion of the projects, these reports were used as data source for the quantitative and qualitative analyses conducted by this research. Thus, the project performance data were collected by and for project participants rather than specifically for this research; and that the data were used for assessing project progress against the plan by ABC project management, rather than for project post implementation reviews such as the one advocated by Doll et al. (2003).

The managers of the Design, Build, and Test stages within PD (UK and India) presented schedule performance data against baseline schedule (see section 2.2 - Chapter 2) on their work progress at the weekly project performance meeting with project management (UK and India); a project monitoring and control mechanism in ABC. These meetings were held face to face at each location and through teleconferencing connecting UK and India. Schedule performance was considered the key measure of delivery success within the PD; and so all effort was focused on meeting schedule targets. Figure 3.16 shows an example performance report for a phase; one of these was produced by each of the phase managers every week.

Textual explanation

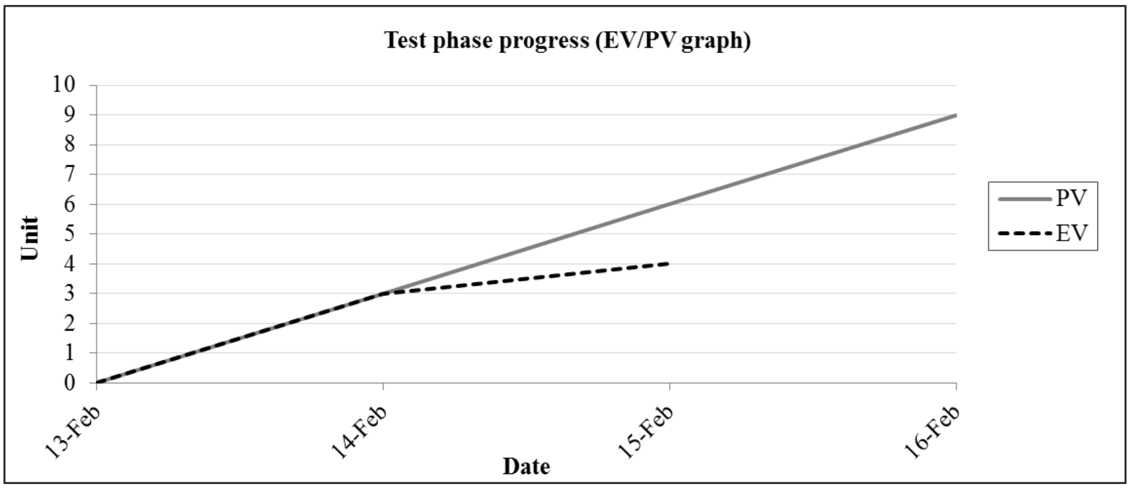
| |
|---|
| Integration Test started Wk1 and all components blocked by defect XXXXNNNNNNNN. On Shore Test Lead recovering from Surgery. Code Fix is required for Operating System X L Integration. |

| | Components provided | | Components tested with Legacy components | | Components tested with Authentication and Component catalogue tools | |
|---|---|---|---|---|---|---|
| Test scenarios | *Planned occurrence date* | *Actual occurrence date* | Planned occurrence date | Actual occurrence date | *Planned occurrence date* | *Actual occurrence date* |
| Test scenario 1 | 14.Feb | 14.Feb | **15.Feb** | 15.Feb | 16.Feb | |
| Test scenario 2 | 14.Feb | 14.Feb | 15.Feb | | 16.Feb | |
| Test scenario 3 | 14.Feb | 14.Feb | 15.Feb | | 16.Feb | |

Event tracking

| Metric | Value |
|---|---|
| EV | 4 |
| PV | 6 |
| BAC | 9 |
| SPI | 0.67 |
| PC | 44.44% |

Numeric performance indicators



Graph

**Figure 3.16 Project performance report in ABC - example**

In this report, four elements can be distinguished:

## 3.4.1 Textual explanation

This part of the report contained textual narrative, a free text area for the Test manager to explain the causes of lack of progress and highlight issues that needed higher management support/attention. The textual information of ABC project performance reports became the source data for the qualitative analysis in this research (see section 7.2.2 - Chapter 7).

A close reading of the textual explanation in the example report (Figure 3.16) indicates that the writer (i.e. Test manager in this case) assumes that the readers (i.e. Project managers) have considerable background knowledge of the projects and omits generally known contextual information within ABC projects. For example, in the final statement 'Code Fix is required for Operating System X L Integration' it is not clear what role (who?) is responsible for providing the code fix. This is because, the textual information was written for an audience of busy senior Project managers of ABC and the Client, both extremely familiar with the contextual background of the projects since many projects had been undertaken within the same programme for the past decade of their relationship. Consequently, these texts used a shorthand means of conveying messages.

### 3.4.2 Event tracking

Event tracking was used to capture basic progress data; an event tracking sheet (see Figure 3.16) was created by the Test manager to track planned and actual achievement of events in the phase. For example, if the event 'Components tested with Legacy components' for 'Test scenario 1' was planned to occur on the 15th Feb (in bold), and today is actually 15th February; the Test manager would enter the value of 15th Feb in the 'Actual occurrence date' if the work was completed, otherwise it was left blank indicating an uncompleted activity. The first event on the tracking sheet typically tracked submission of project artefacts from the previous phase which this phase was dependent on as input to carry out its activities. Figure 3.16 shows that this was the event 'Components provided' by the Code phase so that they can be tested.

The focus of the project management was on tracking the actual finish date of activities; through marking the event as achieved, rather than their start date or mid progress. An activity was only marked as finished when it was 100% complete (see section 2.2.3 - Chapter 2). Hence even if the activity or phase was started it would not appear as such on the tracking sheet. It would instead be stated on the Textual part of the report that the phase was started. Thus, the 'Planned occurrence date' and the 'Actual occurrence date' of the first event on the Event tracking sheet, and the 'Planned occurrence date' and the 'Actual occurrence date' of the last event on the sheet became the scheduled start day, actual start day, scheduled finish day, and actual finish day of a phase, and that were used as variables in the quantitative approach in this research (see section 5.3.2 - Chapter 5) alongside the dependency/precedence relationships among the phases indicated by the first event on the tracking sheet.

### 3.4.3 Numeric performance indicators

The basic progress data captured above could now be turned into performance indicators that were more meaningful to the managers of the project. These were used to show the rate of progress against the planned targets on a weekly basis. Of these, SPI is relevant to this research as it described target achievement of the schedule (see section 5.3.3 - Chapter 5). Schedule Performance Index (SPI) was used to establish what had actually been achieved against what was planned to achieve weekly, by calculating the ratio of EV to PV (SPI = EV divided by PV) - see section 2.2.3 (Chapter 2). The practice at ABC was for PV (planned value) to be a count of the completion dates that should have been met on a particular day. The EV (earned value) was a count of the completion dates that had actually been achieved. An SPI value greater than 1 meant that project progress was ahead of schedule, an SPI less than 1 indicated that progress was behind schedule. Hence an SPI of 1 indicated that progress was on schedule. BAC (budget at completion) is the overall estimated project cost, that is, the total PV for the project; and PC (percentage complete) is the amount of work that has been completed relative to the overall estimated project cost.

### 3.4.4 Graph

The date entries of the Event tracking sheet are converted into units, which are then counted to generate the EV/PV graph for visual comparison. Note that ABC calculated EV and PV differently from the conventional way (see section 2.2.3 - Chapter 2), by using count of events rather than financial value, so they actually are event PVs and event EVs; consequently, the SPI value calculated in the previous section would be more accurately called ‘event SPI’; i.e. a locally-tailored SPI.

## 3.5 The position of ABC’s work practices

This chapter has provided context to the ABC organisation, aiming to position the ABC’s approach to software delivery in light of the standard practices (discussed in Chapter 2) and facilitate putting the subsequent analyses in perspective. The material in this chapter suggests that the ABC’s methodology to control projects varies from the standard model, and understandably for good reasons:

- ABC focused on *events* delivery in performance meetings as in this way the cost will take care of itself (section 3.4.2).

- ABC focused on *speed* of delivery using a version of the iterative and incremental development model with intensive parallel development (section 3.2.3).

- ABC used *narratives* to explain performance indices since the quantitative data alone was not sufficient to guide project participants of what is happening (section 3.4.1).

Furthermore, the ABC’s management of project/phase execution (section 3.3) reveal some notable challenges that may cause schedule delay, for example:

- In some cases, phase events appear to be highly dependent, some of which are outside the particular phase’s control. The implication is that a small delay in one event may trigger a chain of undesired events/delays that might require time unaccounted for and which may delay the overall schedule. Section 3.3.3 is a case in point, where potential rework would be required of a drafted TD document that was based on a not yet signed-off FD (e.g. if sign-off was delayed in the first place), because as the FD gets signed-off, the TD has to be revisited to ensure any potential last changes is reflected in the TD.

- Some events may appear as potential bottleneck during project execution, where the smooth flow of phase execution is disrupted; for example, the Client (section 3.3.1) not signing-off a phase product unless it absolutely reflects their ‘version of truth’, may delay base-lining the FD, delay starting off the FD/TD Transition, and in turn delay in starting the TD...etc.

- The time zone differences in globally distributed teams, although perceived to work to the advantage of the teams as providing 24 hour software development effort;

showed signs of actually hindering progress if not managed properly, as the example, of the Code deploy process failing overnight, presented in section 3.3.10.

The above observations sounded interesting and a more detailed investigation was required in order to further identify causes of schedule delay in software projects. The next chapter puts forward the research methodology employed in this thesis to examine ABC projects more closely.

# 4 An appropriate research strategy

## 4.1 Introduction

Chapter 3 described the software development processes at ABC and the mechanisms ABC used to analyse their project performance data and control their projects. Chapter 2 described the mechanisms proposed in the reviewed literature on how project performance could be analysed. This chapter now describes the methodology used in this research to analyse the project data of ABC.

This research adopted a mixed methods research (MMR) approach to enquiry; combining quantitative (QUAN) and qualitative (QUAL) approaches. This chapter presents the MMR approach along with literature reviewed on methods of software project management research. The QUAN and QUAL approaches, their analyses and results are presented collectively in chapters 5, 7 and 8 respectively for easy reference.

The chapter starts with exploring methods of project management research in light of our understanding of the nature of project or project management in order to devise suitable research methodology to investigate it. This is then followed by the works of this research programme in its research methodology and design. Finally, the methodological position of this research is summarised.

## 4.2 Methods of project management research

Chapter 2 (section 2.5) argued that despite the contribution of the traditional approaches to project management research - surveys, interviews or case studies (see Table 4.1 for examples - which classifies some of the studies surveyed in Chapter 2 according to the research methods used) - academic research should adopt research approaches beyond the conventional ones in order to enhance our understanding and address the complexity inherent in managing projects; quoting Muller et al. (2013) that ‘if we always do what we always did then we should not be surprised that we always find what we always found’ (page: 24).

As such, useful approaches to project management research may include: the development of models which illuminate the complexity of projects such as the interactions between human and nonhuman actors (Muller et al. 2013; Winter et al. 2006); develop concepts that reflect the lived experiences of practising project managers (Cicmil et al. 2006; Winter et al. 2006; Muller et al. 2013); employ different research approaches that examine the collective influence of the internal and external factors on project outcome (Clegg, 2013; Muller et al. 2013; Winter et al. 2006). Table 4.2 summarises some of these approaches proposed by researchers.

| # | Traditional methods | Studies |
|---|---|---|
| 1 | Questionnaire | Cerpa & Verner (2009); Tarawneh et al. (2008); Sauer et al. (2007); Procaccino & Verner (2006); Procaccino et al. (2006); Yeo (2002); Carmel (1995) |
| 2 | Interview | Pertersen et al. (2014); Tarawneh et al. (2008); Moløkken-Østvold & Jørgensen (2005); Taylor (2006); Rainer (1999); Abdel-Hamid & Madnick (1991); Curtis et al. (1988) |
| 3 | Past documents | Patanakul (2014); Rainer (1999); Ebert (2007) |
| 4 | Theme development | Taylor (2006); Curtis et al. (1988) |
| 5 | Causal relationships/ frameworks | Patanakul (2014); Cerpa & Verner (2009); Lehtinen et al. (2014); Sauer et al. (2007); (Ebert, 2007); Lu et al. (2010); Yeo (2002); Abdel-Hamid & Madnick (1991); Curtis et al. (1988); Malcolm (1990); Alberts & Dorofee (2010); Dhlamini et al. (2009); Fan & Yu (2004); Rabbi & Mannan (2008); Marvin et al. (1993) |
| 6 | Content / numeric analysis | Patanakul (2014); Cerpa & Verner (2009); Sauer et al. (2007); Ebert (2007); Rainer (1999); Tarawneh et al. (2008); Yeo (2002) |
| 7 | Case study | Patanakul (2014); Pertersen et al. (2014); Moløkken-Østvold & Jørgensen (2005); Patanakul (2014); Lehtinen et al. (2014); Ebert (2007); Rainer (1999); Abdel-Hamid & Madnick (1991); Curtis et al. (1988); Malcolm (1990); Ebert & Neve (2001) |
| 8 | Literature  survey | McLeod & MacDonell (2011); Nasir & Sahibuddin (2011); Lu et al. (2010); Bakker et. al (2010) |
| 9 | More than one method | Lehtinen et al. (2014); Tarawneh et al. (2008); Rainer (1999) |
| 10 | Retrospectives | Nelson (2007); Ebert (2007) |

**Table 4.1 Traditional methods of software project management research**

| **Useful methods** | **Acronym** | **Description (source)** |
|---|---|---|
| Actor-network theory | ANT | Used to study the influence of human and nonhuman actors, which interact to achieve some goal. ANT has been described as 'a useful methodology to organizational research', and that ANT's unique approach 'enables it to shed light on complex ties that so far escaped organization theory', and ANT 'offers significant potential in exploring how projects are managed' (Er et al. 2013, page: 164-165). The ANT's approach in avoiding applying existing classifications to the research problem can help to ensure that research findings more accurately mirror the experience of practitioners (Er et al. 2013) |
| Activity theory | AT | Used to study human activity, where a subject (individual or group) performs activity to achieve an object (purpose). The focus of AT is on the link (activity) between the subject and the object which could be carried out by tools; since, the tool gives clue as to how the activity was carried out; AT has been used to study human computer interaction (Er et al. 2013) |
| Action research | AR | An approach that helps the practitioner to generate knowledge about a social system while, at the same time, attempting to change it (Er et al. 2013, page: 177). It can be useful in project management research where there is a need to investigate social implications from project processes; working collaboratively to find practical solutions to problems arising in a project; and in situations where major project changes need to be implemented (Er et al. 2013) |
| Grounded theory | GT | A set of procedures used for analysing empirical data in order to develop categories or theory of process, sequence, and/or interaction of the area being investigated, from the participants' perspective (Glaser & Strauss 1967, page: 114). The GT techniques can be useful in project management research to develop categories of phenomena that emerge during project execution, and reveal the processes by which these phenomena influence project outcome |
| Simulation | SM | Defined as the 'abstraction of reality for a purpose' (Leigh 2013, page: 200), has been described as useful in project |

| Useful methods | Acronym | Description (source) |
|---|---|---|
| | | management research where interfering with project activities for the purpose of research impedes the normal course of project progress (Leigh, 2013). Leigh (2013) argues that the ‘system dynamics’ approach (see next) grew out of specific application of simulation, focusing on certain structural and conceptual features of simulation that remain the same (page: 199). Computer simulation offers ‘the ability to experiment and the generation of insight into the dynamic performance of complex systems’ (Williams 2005, page: 456) |
| System dynamics | SD | A set of conceptual tools that enable understanding of the structure and dynamics (i.e. interdependence, mutual interaction, information feedback and circular causality) of complex systems (Oorschot, 2013). This approach is thought to be useful to research multiple and interacting processes, finding correlation between possible causal factors and performance outcomes (Oorschot, 2013; Lyneis & Ford, 2007; Robertson & Williams, 2006) |
| Mixed methods research | MMR | An approach to research that combines multiple methods in a single design. It can be argued that in the case where a research program investigates complex phenomena; need arises to use research methodologies that match that complexity in order to generate knowledge of any use. Leigh (2013) noted that ‘Operational contexts of contemporary organisations are becoming increasingly complex, uncertain and turbulent, creating unfamiliar research challenges for which familiar research methodologies do not offer appropriate support’ (page: 199). Furthermore, it has been argued that ‘a complex phenomenon often needs more than one method to investigate it adequately’ (Cameron & Sankaran 2013, page: 383) |

**Table 4.2 Useful methods of software project management research**

The research described in this dissertation used a MMR (see Table 4.2) approach combining both quantitative (project metrics) and qualitative (see GT and ANT in Table 4.2) methods. The literature reviewed on using MMR in software project management research is summarised in section 4.2.1 (next). The literature reviewed relating to metrics used by project managers to control schedule, as well as the use by researchers in empirical studies of past projects was summarised in Chapter 2. The literature reviewed on using GT and ANT in software project management research are reserved for sections 6.4 and 6.5 (Chapter 6) respectively. The rationale for the choice of methods is provided in section 4.4 (research design).

This research can be seen as an example of empirical software engineering research (ESER), the study of the application of software engineering principles in the real world and its influence on the complex interaction of the people, processes, and technology. ESER scholars such as Sjøberg et al. (2007) and Basili et al. (2006) encourage researchers to use a variety of research methods and techniques, including the combining of qualitative and quantitative approaches to enquiry that Ciolkowski & Briand (2006) and Münch (2006) also advocate. (Leszak, 2006) addresses the need to examine software project management issues of developing large software systems with globally development teams from a granular level using real project data. Kitchenham, (2006) calls for using case studies of genuine industrial software engineering projects in order to address the issues being investigated. Empirical studies adopt both quantitative and qualitative approaches to enquiry, though it has historically been associated with the former (Basili et al. 2006).

Some past empirical research appeared to be particularly relevant. Lehtinen and his colleagues' work, published in series of papers (Lehtinen et al. 2014; Lehtinen & Mäntylä, 2011; Lehtinen et al. 2011), investigated Agile software engineering projects of various sizes in four medium-size software companies in Finland. The research analysed problems including schedule delay and developed models that depict the causes of software project failure. The multiple case study approach used root cause analysis (RCA) to collect data. Existing categories of causes were borrowed from literature (such as 'process area' or 'cause type'), and were then combined using Grounded theory techniques to identify further categories of interconnected causes that crossed over process areas. These causes were analysed quantitatively to identify feasible process improvements. The researchers acknowledged their use of RCA was prone to the subjective judgment of case participants. The researchers attempted to triangulate the data with interviews to confirm causes were ‘correct and accurate’. The authors contended that the research methods used allowed them to ‘construct the story behind the data’ and concluded that ‘more empirical research is needed to better understand the complicated mechanisms and relationships of causes leading to software project failures’ (page 642). Although Lehtinen and colleagues’ study does not claim to have used a mixed method approach; it would appear to have integrated quantitative and qualitative approaches to enquiry in order to make sense of the complexity inherent in software projects.

Lehtinen and colleagues’ study is similar to the works presented in this thesis: both studies investigate software development projects, combine case study research with GT analysis, and examine causes of software project delay. The differences between the two studies, however, can be seen in that: the Lehtinen and colleagues’ study investigated Agile projects, developing medium size software, and in collocated team environment in one country; whilst the work presented in this research investigated software projects that adopted the iterative and incremental model, developing large software systems, with globally distributed teams. Furthermore, Lehtinen and colleagues’ study used interviews to collect textual data for their subsequent qualitative analysis, and questionnaires to collect numeric data for their subsequent quantitative analysis. Contrasted with this research, both the numeric and textual data were parts of the same project performance reports that were created by and for project participants in ABC, and that mixed method approach was used to analyse the data. Hence, this research extends Lehtinen and colleagues’ findings by offering insight into a set of factors influencing schedule delay in a different project context.

### 4.2.1 Mixed methods in software project management research

The use of mixed methods research (MMR) in empirical research is not new (Tashakkori & Teddlie 2010, pages: 804 and 808; Creswell, 2009) and it has been used in various disciplines, such as education, health and medicine, and management studies (Ivankova & Kawamura 2010, page 593). The use of MMR in empirical software engineering research is not new either: Ivankova & Kawamura (2010) noted that MMR has been specifically referred to as early as 2002. However, earlier studies advocated combining the QUAN and QUAL methods in empirical software engineering research (Seaman, 1999), which is also advocated by Sjøberg et al. (2007).

In a literature survey, of studies published between the years (2000 - 2008) on the adoption of MMR in empirical research, Ivankova & Kawamura (2010) report that the use of MMR was beneficial and produced meaningful and credible findings. One example was the use of MMR to analyse the activities of the users of virtual reality application software (Feldon & Kafai, 2008). This quantified the behavioural patterns of the study participants, and showed the frequencies and types of interactivity by the participants, at the same time using a qualitative approach to provide the context of the interactions (Ivankova & Kawamura 2010, page: 600).

A reason for considering the MMR to study a phenomenon could be its complexity; necessitating the use of a wider range of methods rather than one method alone (Ivankova & Kawamura, 2010). Managing the development of large software systems with globally distributed teams is a complex undertaking that involves not only technology, but people, processes, and its environments. For the researcher to make sense of what is happening during project execution they may need to integrate many different methods and techniques (Coleman & O’Connor, 2007; Seaman, 1999). Earlier, this thesis (section 2.4.1- Chapter 2) noted the complexity inherent in developing software systems due to the number and variety of project elements, interactions, interdependencies, and the rate of change of the project context. Table 4.3 lists example empirical studies that used MMR in software project management research. The remainder of this chapter describe this research’s methodology and design.

| **Purpose of use** | **How it was used (source)** |
|---|---|
| Study the degree of adoption of project risk management and the barriers to its use by information technology project managers | A two phase research approach was adopted. An exploratory phase to develop an understanding of the research problem through semi-structured interviews then coding the qualitative data and generating categories, followed by an explanatory phase using a survey to confirm or refute the findings (Kutsch & Hall, 2009) |
| Investigate coordination and communication activities in globally distributed teams; examining the influence of the individuals' locus of control, i.e. an individual's perception of their control in a work situation in terms of the degree to which their effort does actually affect work outcome | Used a survey questionnaire to test the effect of locus of control on team member perception of role conflict, followed by case study interviews to gain an understanding of the issues identified in the survey facing individuals. The author reported that the integrative approach helped understanding the individual's locus of control orientation and its impact on the team member's motivation in a distributed work environment (Lee-Kelly, 2006) |
| Investigate the perceived benefits of implementing standardised project management (STPM) practices in improving project performance | Combined qualitative and quantitative approaches in three phases. A qualitative phase used case study with interviews and observations to develop categories of factors that make STPM effort successful. This was followed by a quantitative phase that used the STPM categories to develop hypotheses and perform hypothesis testing using a survey questionnaire; followed by interviews to enrich and refine the findings qualitatively (Milosevic & Patanakul, 2005) |
| Research the impact of steering committees on project performance and the creation of value | Used case study (to analyse the specific functions of steering committees), followed by multilevel surveys (interviewing senior managers at organisational level and distributing questionnaire to |

| **Purpose of use** | **How it was used (source)** |
|---|---|
| from project management capabilities | project managers at project level). The authors concluded that the complex and diverse nature of the research problem required multilevel research approach (Lechler & Cohen, 2009) |
| Understand how corporate strategy is implemented via projects | Combining case study research, semi-structured interviews, and past documents (Morris & Jamieson, 2005) |
| Update the APM Body of Knowledge 4th edition | Using structured interviews, web questionnaires, and past documents to assess the influence of the relationship between the project manager's leadership style and project type on project success (Morris et al. 2006) |
| Develop models of how corporate strategy is implemented through projects, and investigated the diffusion and adoption of new product development tools in Singapore | Combining qualitative case study (to generate hypotheses) and quantitative survey (to test hypotheses); the researchers reported that new understanding was generated as a result of this combination (Chai & Xin, 2006) |

**Table 4.3 Mixed methods approach in project management research**

## 4.3 Methodology

This research adopted a mixed method (Creswell & Plano Clark, 2011) methodology drawing from both the QUAN and QUAL methods of enquiry to identify the influencing factors on schedule delay. This approach was appropriate because the research data, consisting of project performance reports, contained both numeric data and text. It became clear that one approach alone would not meet the needs of the research. The early research, analysing the numeric data, enabled the identification of the project phases with the most delays, but not the causes of delay. For example, analysing Project 1's numeric data showed that the Integration Test phase in increment 4 - IT (Inc4) contributed most to delaying Project 1; that is, 42 days (81% of the project's 52 days total delay). Analysing the textual data revealed phenomena leading to delays during the execution of the phase; such as the large number of the defects found and delays in their correction. This was followed by the tracing of the interactions among the project actors (human and nonhuman) during project execution which made apparent the influence of project actors on schedule delay. In this way, integrating the QUAN and QUAL approaches enabled identifying the influences on schedule delay.

Several different terms describe research methods that use multiple approaches in a single design: multimethod, multi-strategy, multiple method, mixed methodology, mixed research, and mixed methods - see for example Creswell & Plano Clark (2011), Tashakkori & Teddlie (1998), Teddlie & Tashakkori (2010) and Robson (2011). Teddlie & Tashakkori (2010, page: 19) proposed the following to be a common definition of such an approach, which was developed by Johnson et al. (2007, page: 123):

> 'Mixed methods research is the type of research in which a researcher or team of researchers combines elements of qualitative and quantitative research approaches (e.g., use of qualitative and quantitative viewpoints, data collection, analysis, inference techniques) for the broad purposes of breadth and depth of understanding and corroboration'

Cameron (2013) has neatly distinguished between the variants of this approach. According to Cameron, multiple-method designs can be categorised as either 'Multimethod research' or 'Mixed methods designs' (page: 78):

- Multimethod research is where the researcher adopts *one approach* to enquiry in a single study, *either* Quantitative or Qualitative. Then within this approach more than one method of data collection or analysis is used: for example; in a Qualitative study, both interviewing participants and searching archived documents.

- Mixed methods designs mean *more than one approach* to enquiry is adopted in a single study, *both* Quantitative and Qualitative: for example using Case study (QUAL) and Non-Experimental statistical (QUAN) approaches in the same study. This design has further two sub-types:

    - Mixed model research - mixing of the QUAN and QUAL can occur *in many or all* stages of the study (i.e. forming research questions, selecting methods, data collection, analysis, interpretation).

    - Mixed method research - mixing of the QUAN and QUAL occurs *only* at the *selecting methods* stage of the study; where the data collection and analysis according to both the QUAN and QUAL approaches take place in sequence or in parallel; i.e. in the other stages of the study the QUAN and QUAL remain separate.

This research falls under the mixed method research - i.e. last category (in its sequential approach). There are a number of reasons for mixing research methods - see Table 4.4 for examples (adapted from Creswell & Plano Clark 2011, pages: 62-63; Venkatesh et al. 2013).

| # | **Reason for mixing** | **Description** |
|---|---|---|
| 1 | Triangulation | Seeks convergence and corroboration of results from the different methods |
| 2 | Complementarity | Seeks elaboration, enhancement, illustration, and |

| # | Reason for mixing | Description |
|---|---|---|
| | | clarification of the results from one method with the results from the other method |
| 3 | Completeness | Bringing together a more comprehensive account of the area of enquiry |
| 4 | Explanation | One method is used to help explain findings generated by the other method |
| 5 | Offset | Combining the quantitative and qualitative methods to offset the weaknesses in each of the methods and draw on the strengths of both |

**Table 4.4 Reasons for mixing research methods - example**

The mixed method methodology in this research was to explain the findings generated by the quantitative approach with the one generated by the qualitative approach.

The overall approach in this research was exploratory, remaining open to any insights the empirical data might provide rather than developing or testing hypotheses. The decision to adopt MMR emerged (Creswell & Plano Clark 2011, page: 54) during the course of the study, rather than at the outset. Early work revealed that adopting a QUAN method alone provided only a partial understanding of the area; hence a QUAL strand of enquiry was added to explain the QUAN results (due to part of the research data being numeric and the other part textual). As will be seen, a variety of methods and techniques were integrated in order to interpret both the QUAN and QUAL results in light of the contextual data about ABC practices. Teddlie & Tashakkori (2010, page: 8) recommend employing variety of methods as needed to answer research questions that evolve as the study unfolds.

## 4.4 Design

Several mixed method designs have been proposed - see for example Creswell & Plano Clark (2011), Tashakkori & Teddlie (2008), Collins (2010), Onwuegbuzie & Combs

(2010), and Creswell (2009). During research design, consideration has to be given to (Creswell & Plano Clark 2011, page: 63-66):

(i) The order of conducting the QUAN and QUAL methods - whether sequential, concurrent, or multiphase. The order in this research was sequential; the QUAN method first, followed by the QUAL method.

(ii) The priority (relative importance) of the QUAN and QUAL methods for answering the research questions - whether equal priority or one method given more weighting during design. Both methods in this research played an equally important role in addressing the research questions.

(iii) Integration of the methods - whether to connect one of the methods to the other, embed one method within the other, or keep the methods independent and mix only at conclusion. The QUAL method in this research was dependent on the results of the QUAN method, and so they were connected.

Table 4.5 shows examples of such designs (adapted from Creswell & Plano Clark 2011, page: 73-76). This research applied the Explanatory sequential design (#2 in Table 4.5) for its suitability to the journey undertaken by this research. The Explanatory sequential design started with the collection and analysis of numeric data. Results from the QUAN analysis identified cases for the QUAL analysis of textual data to explain the QUAN results (Creswell & Plano Clark 2011, page: 71; Creswell et al. 2008).

| # | **Design type** | **Description** | **Purpose of use** |
|---|---|---|---|
| 1 | Convergent parallel | Methods implemented in parallel: Quantitative and qualitative data collected concurrently, analysed separately, and merged during interpretation | Validate and corroborate quantitative scales |
| 2 | Explanatory sequential | Methods implemented sequentially: Quantitative data collection and analysis first (phase 1), followed by qualitative data collection and analysis (phase 2) which | Explain quantitative results |

| # | Design type | Description | Purpose of use |
|---|---|---|---|
| | | builds on phase 1 | |
| 3 | Exploratory sequential | Methods implemented sequentially: Qualitative data collection and analysis first (phase 1), followed by quantitative data collection and analysis (phase 2) which builds on phase 1 | Test or measure qualitative exploratory findings |
| 4 | Embedded | Implementation of a mini concurrent or mini sequential design within (before, during, or after) a major concurrent a or major sequential design | Preliminary exploration before an experimental trial |
| 5 | Transformative | Framing the concurrent and/or sequential implementations within a theoretical framework that guide the methods decisions | Conduct research that identifies and challenges social injustice |
| 6 | Multiphase | Combining the implementation of concurrent and/or sequential designs over multiple phases of a programme of study | Address program objectives, such as program development and evaluation |

**Table 4.5 Mixed method designs**

Figure 4.1 depicts the Explanatory sequential design of this research; the flow chart was adapted from Creswell & Plano Clark (2011). The notation within the flow chart elements was adapted from Cameron’s MMR notation system (Cameron, 2012). Cameron calls for the combining of the textual notation with the flow chart to improve the reporting of mixed methods studies and this has been done. The acronyms used in Figure 4.1 are shown in Table 4.6. A description of the research design (Figure 4.1) and the rationale for the choices mode are provided following Figure 4.1.

| Acronym | Description |
|---|---|
| DS | Data source (2ndy: secondary) |

| **Acronym** | **Description** |
|---|---|
| S-SIZE | Sample size |
| INST | Data collection instrument |
| VAR | Variables |
| ANAL | Analysis technique |
| UOA | Unit of analysis |
| QT | Quantitative |
| QL | Qualitative |
| n | Number |

**Table 4.6 Research design acronyms**

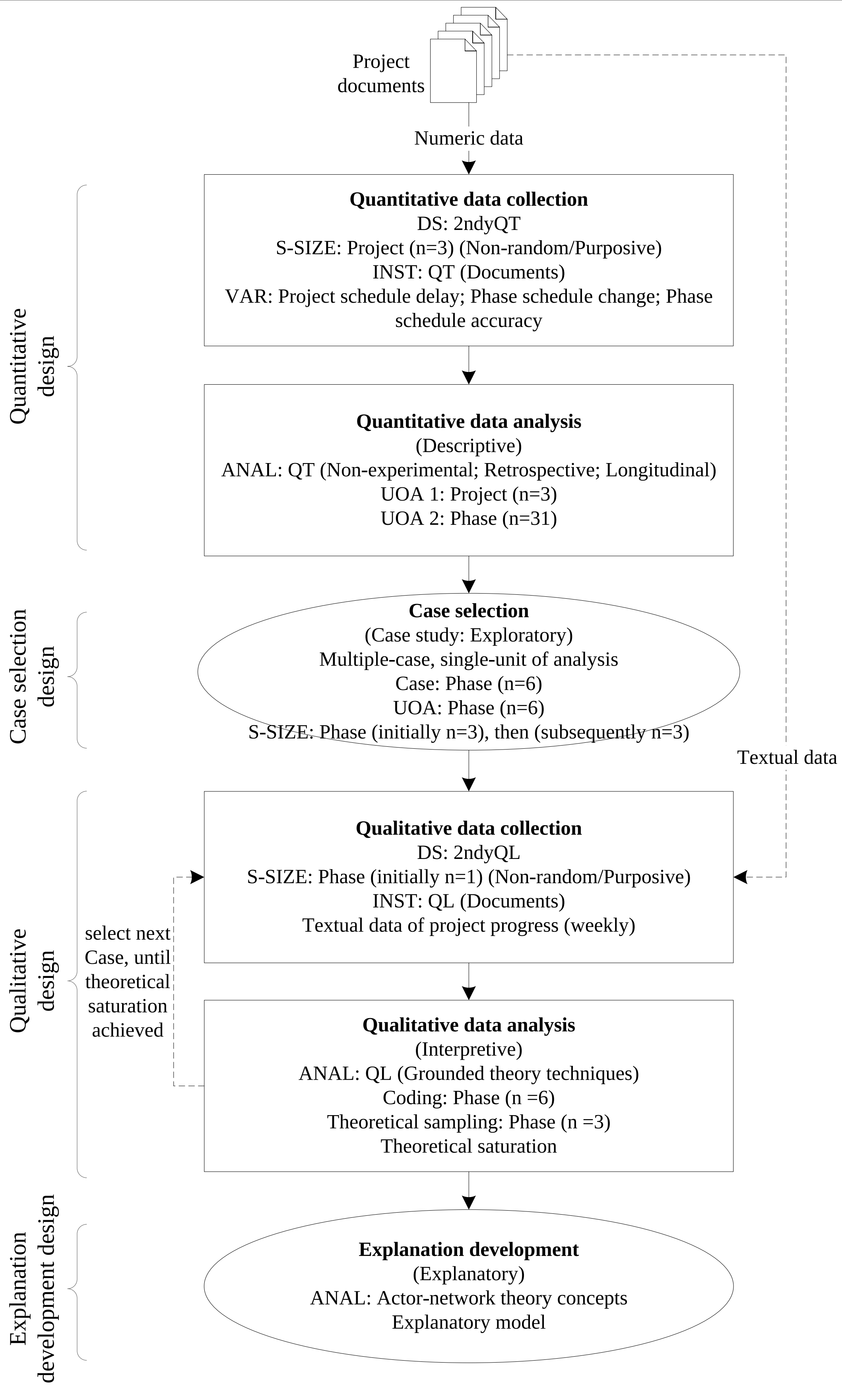

Project documents
Numeric data
Quantitative data collection
DS: 2ndyQT
S-SIZE: Project (n=3) (Non-random/Purposive)
INST: QT (Documents)
VAR: Project schedule delay; Phase schedule change; Phase schedule accuracy
Quantitative data analysis
(Descriptive)
ANAL: QT (Non-experimental; Retrospective; Longitudinal)
UOA 1: Project (n=3)
UOA 2: Phase (n=31)
Case selection
(Case study: Exploratory)
Multiple-case, single-unit of analysis
Case: Phase (n=6)
UOA: Phase (n=6)
S-SIZE: Phase (initially n=3), then (subsequently n=3)
Textual data
Qualitative data collection
DS: 2ndyQL
S-SIZE: Phase (initially n=1) (Non-random/Purposive)
INST: QL (Documents)
Textual data of project progress (weekly)
select next Case, until theoretical saturation achieved
Qualitative data analysis
(Interpretive)
ANAL: QL (Grounded theory techniques)
Coding: Phase (n =6)
Theoretical sampling: Phase (n =3)
Theoretical saturation
Explanation development
(Explanatory)
ANAL: Actor-network theory concepts
Explanatory model
Quantitative design
Case selection design
Qualitative design
Explanation development design

**Figure 4.1 Research design**

### 4.4.1 Quantitative design

This step of the research process attempted to develop answers to the first research question:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

In order to answer RQ1, the research investigated project performance data of three completed projects which ABC had collected and analysed to track progress during project execution. Analysing past project documents meant that the research had no control over the behavioural events of the projects and could not intervene (Yin, 2009). Thus, the enquiry at this stage was descriptive; the design was non-experimental and retrospective (studying completed projects) and longitudinal (examining phenomenon over multiple data points in time) (Robson, 2011; Kumar, 2011; Sjøberg et al. 2007). Non-experimental designs offer the benefit of dealing with things in their natural settings without disturbing them. Examples of applying quantitative method in empirical studies include Lipke et al. (2009) and Kim et al. (2003).

The focus of the analysis of the numerical data was schedule behaviour (schedule delay, schedule change, and schedule accuracy) with the aim of learning the extent to which the mechanisms used to control schedule duration identified the causes of delay. The schedule metrics in the QUAN study were descriptive, and not used as statistical techniques to confirm any hypotheses (Robson, 2011). The output of this stage was identification of project phases most contributing to project delay. Further details of this analysis are provided in Chapter 5.

### 4.4.2 Case selection design

The QUAN analysis, as will be seen in Chapter 5, revealed that small number of the Test phases contributed most to project delay. Thus the possible reasons for delays in these phases were of particular interest, and a case study (Eisenhardt, 1989) was designed to examine the textual data in the project reports of the Test phases. The analysis of the case study data would be qualitative, leading to a mixed method research strategy (Creswell & Plano Clark 2011, page: 54) where quantitative methods (Chapter 5) and qualitative methods (here) are linked to make sense of the studied domain. The purpose of the research at this stage was to explore the Test phases in more detail in order to answer the remaining two research questions through qualitative analysis and explanation:

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

The reliance on past project documents as data, which constrained the study from exercising intervention, supported the choice of case study research. Case study design (Yin, 2009) allows (i) examination of naturally occurring phenomena, in one or few areas, in considerable depth and over time to gain maximum learning (ii) investigation of causal effects of hidden processes (Hammersley & Gomm, 2000). Besides, 'Conducting case studies is a standard method of empirical study in management and related disciplines such as organization development and information systems (IS) research' (Sjøberg et al. 2007, page: 4).

Case study research is appropriate for research questions concerned with the 'why' and 'how' aspects of an empirical setting, and where the investigator has no control over the events (Yin 2009, page: 8; Eisenhardt 1989, page: 542; Verner et al. 2009). Benbasat et al. (1987) noted that case study research is appropriate for practice based problems where the experience of the actors is important and the context of action is critical. Case study research is also useful for studying information systems in their natural setting, and for understanding the nature and complexity of the processes taking place, as operational links can be traced over time. In studying organisations, Remenyi (2013) suggests that, case study research should be used to investigate organisations in an appropriate industry sector, of appropriate size, and sufficiently complex in nature to be interesting (page: 30). Further details of this analysis are provided in Chapter 6.

### 4.4.3 Qualitative design

This step of the research process attempted to develop answers to the second research question:

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

The aim of the investigation at this stage was to determine, from the textual reports, what phenomena emerged during the execution of the Test phases that influenced schedule duration; thus, the nature of the enquiry at this stage was interpretive; interpreting the participants' views of what was happening within the Test phases of the projects being examined.

The approach to textual data analysis was that of Grounded theory (Glaser & Strauss, 1967). Corbin & Strauss (2008) noted that Grounded theory (GT) practices can be used to construct theory, develop thick and rich descriptions, develop concepts, and pull out themes (page 162). Urquhart asserted that GT can be (and has been) used for purposes other than theory generation but the researcher needs to state when this is the case. This can be to support coding (in the sense of identifying categories) or data analysis (Urquhart, 2013). This research used GT techniques to identify categories of phenomena that influence schedule duration. This took advantage of the rigour of GT coding techniques compared to others like thematic analysis (Miles & Huberman, 1994; Robson, 2011). Compared to other types of coding in qualitative research, GT codes are not imported from pre-existing ideas in the literature, but emerge from analysing the data. They are applied to a detailed level of data rather than large chunks of text (Urquhart, 2013).

The research then collected textual data, from the same past project documents used in the QUAN phase, for one case selected purposively (Kumar, 2011; Robson, 2011). The content of this case was analysed qualitatively using GT techniques (see section 6.4 -

Chapter 6), and the emergent types of factors (i.e. ‘categories’ in GT term) that appeared (i.e. ‘emerged’ in GT term) from the case gave rise to the selection of the next case (using the Theoretical sampling approach of GT where the next case is selected to develop and/or to generalise categories - see section 7.2.1 - Chapter 7). This process continued until selecting more cases produced no more new categories of phenomena from the ones emerged already (Corbin & Strauss, 2008, page 148). This approach seemed appropriate because the aim was to identify the common causes across the six cases that influence schedule duration. The output was 5 categories of phenomena emerged from six Test phases (including the three Test phases selected in the previous step) across the three projects, and a narrative schema. The narrative schema is a visual representation of the analysed content of the textual data of the Test phase performance reports, showing the relationship among the various phenomena emerged during the Test phase execution.

Although the narrative schema answered RQ2, it had two limitations. It was a representation of the contents of textual data in the Test performance reports, reported by the Test manager, and thus represented a particular perspective of what was happening. It also lacked the contextual information surrounding the Test phases of which project participants would be aware, but which was not explicitly mentioned. To obtain a more complete understanding of the influences on schedule delay, the contextual information was brought in and further input from project participants were sought. However, the resultant picture was very local to ABC (further details of this analysis are provided in Chapter 7). Further analyses were needed to take our understanding to higher level of abstraction, as is explained in the next section.

### 4.4.4 Design of developing explanation

This step of the research process attempted to develop answers to the third research question:

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

As will be seen that, the narrative schema in Chapter 7 revealed that the factors directly affecting the task completion related not just to human performance but also to nonhuman intermediaries, such as software tools and platforms, and intermediate products passed between human actors. This suggested that the project could be seen as a network of actors, both human and nonhuman, each of whom can, to varying degrees, empower or constrain others. The picture that emerged was aligned to that expected by Actor-network theory (ANT) (Law, 2012; Callon, 2012; Latour, 2005), and therefore the ability of ANT (see section 6.5 - Chapter 6) to illuminate what was happening demanded attention.

The aim of the investigation at this stage was explanatory: to understand the interactions that developed during project execution, and how they influence schedule delay. An explanatory study was appropriate since they focus on tracing operational links over time (Yin, 2009; Maxwell, 1992; Guba & Lincoln, 1994; Robson, 2011). This involved applying ANT concepts to the emergent categories of phenomena from the GT analysis in the previous research step. This then culminated in an ANT model of schedule delay; the explanation of which can be applied to a broader range of experiences/contexts of other researches. Further details of this analysis are provided in Chapter 8.

## 4.5 Rationale

The motivation of this research, as was articulated in Chapter 1, stems from the need to have better understanding of the project behaviours that influence software project progress. This chapter argued that we need to adopt research approaches beyond the conventional ones in order to obtain such understanding and address the complexity inherent in managing projects. This section discusses the practical implications of the decisions made about the research approach.

The research obtained case study data through personal contact of the researcher that led to two of the ABC Managing Directors to make available the projects’ reports to be analysed by the research. An implication of this is that the decision about research methods is going to be affected by what data is available. The researcher’s past experience, as professional software engineering and project management consultant for a number of years, in similar projects and organisations to the ones investigated in this research offered practical benefits and contextual knowledge to enable interpretation of the project data closer to reality compared to not having such knowledge. A limitation may be seen in such situation is that the researcher may make assumptions based on their past project experience which may not actually apply to the new situation. However, some researchers argue that ‘Context not only ground concepts, but also minimise the chances of distorting meaning and/or misrepresenting intent’ (Corbin & Strauss 2008, page: 57). Furthermore, others (see for example Coleman & O’Connor, 2007; Fitzgerald, 1998; Anderson, 2006) have argued that having the knowledge of the cultural insider - i.e. having prior expertise and practical knowledge of the domain enables better understanding of the data and the quality of the knowledge created. Still, the author of this dissertation diligently sought maintaining a degree of impartiality and distance throughout the research process.

The key research data were records of past project progress reports produced by and for the project team purely for internal use rather than being specifically collected for this research. The research analysed the contents of these weekly archival documents. The value of this approach is in part that it provides access to the details of day to day execution of the projects as perceived by project team and thus it gives insights that other approaches such as observation and participant interviews (see for example Lehtinen et al. 2014) do not provide; for e.g. comprehensiveness, avoidance of retrospective reconstruction of history. The focus on analysing source documents allows a more rigorous, more objective picture to emerge (Deephouse et al. 1996) compared to interviews which would be prone to the subjective judgment of case participants. Limitations of this approach - common to most content analysis - are that the research is constrained by the scope of the project reports, and the need for additional, contextual, information in order to understand content produced purely for internal communication. In common with other approaches using historical records it may be difficult to go back and clarify certain aspects of the data. This is where a researcher’s personal knowledge about a domain is useful. Much of this data about organisational structures and process flows (Chapter 3) are factual (and were clarified by the project team); e.g. software tools deployed and these were used to provide context to the Test phases to obtain a more complete picture of what was happening.

Researchers have proposed various ways in evaluating mixed methods research (see for example Mingers, 2001; O'Cathain et al. 2008; Creswell & Plano Clark, 2011; O'Cathain, 2010; Tashakkori & Teddlie, 2008; Onwuegbuzie & Johnson, 2006). However, Venkatesh et al. (2013) offer a useful guideline for conducting mixed methods research in IS, and is used in this research to evaluate the approach adopted. According to Venkatesh et al. (2013), researchers should consider the following factors when conducting a mixed methods study - each of these factors is followed by what was actually done in this research separated by a hyphen (-) for contrast:

- The appropriateness of a mixed methods approach to the research questions, objectives and context - the general appropriateness of mixed methods in this research was established earlier in the chapter as a necessity; the research data was a mix of numeric and text, and using only one method of enquiry (i.e. quantitative or qualitative) did not support answering the research questions and would have provided a partial view of what was happening. Furthermore, the context of the investigated domain was complex requiring analysis of different data types (numeric and text) in order to make sense of what was happening and obtain a fuller view of the influencing factors on schedule delay.

- Development of meta-inferences (i.e. theoretical statements, narratives, or a story inferred from an integration of findings from quantitative and qualitative strands of mixed methods research) - this research developed an explanatory model (Chapter 8) which is essentially a meta-inference drawn from an integration of the quantitative results (Chapter 5) and qualitative findings (Chapter 7). According to the 'meta-inference analysis path' suggested by Venkatesh et al. (2013, page: 39), the path in this research was: quantitative findings > qualitative findings > metainferences.

  The choice of Grounded theory (GT) techniques to analyse the textual data instead of a literature informed case study, given prior work, stems from the need of the research to make sense of what is happing in the project reports, i.e. grounded in the data, rather than what existing frameworks might inform a priori. Chapter 2 (section

2.5) argued that existing work do not address complexity inherent in real projects, and that it does not provide neat explanations of the causes of project failure and so a literature informed analysis would yield the same findings. One value of GT is that there are different levels of analysis - and each one can be valuable in its own way. At particularly the higher levels there is degree of subjectivity, but the reader can be reasonably sure that the concepts can be linked back to the data - hence the 'grounded'.

The choice of Actor-network theory (ANT) to develop explanation of schedule delay instead of other social theories, such as Structuration theory or Sociomateriality, was driven by the nature of the outcome of the textual analysis of the project progress reports (Chapter 7); i.e. the narrative schema. The narrative schema revealed various types of actors (human and nonhuman), their relationships, and the interactions among them influencing schedule duration. This composition meant that a suitable decision had to be made as to the most appropriate approach to make sense of what was in hand. Although the three approaches can be seen similar, there are some minor differences. For example, considerations of using Structuration theory (Giddens, 1984) for making sense of the narrative schema would have underrepresented the influence of technology on schedule delay (Allison 2004: page, 74; Allison & Merali, 2007), because technology theory is underdeveloped in Structuration theory (Greenhalgh & Stones 2010, page: 1287) and is not within the Giddens' sphere of interest (Leonardi 2011, page: 150); however, in the narrative schema humans and nonhumans appear equally influencing schedule duration. The potential use of Sociomateriality to understand the narrative schema would have meant that consideration of human intention be accounted for in the analysis; although Sociomateriality recognises the influence of material as well as human, it distinguishes between them as the latter involves the *intention* of doing something (Leonardi, 2013); however, the narrative schema, representing what was reported in the project progress reports, surfaced factors relating to obstacles to progress rather than human intentions of how to go about removing them; the latter although discussed in the weekly status meetings it was not reported. Therefore, ANT appeared more suitable to make sense of the narrative schema given its approach of not distinguishing between human and nonhuman during analysis in order to makes sense of what might actually be happening rather than framing them within existing concepts. Besides, the attempts made by researchers to combine Strong structuration theory - the application of Giddens' theory to empirical situations, and Actor-network theory illustrates this need in order to obtain a fuller understanding of a studied domain - see for example Greenhalgh & Stones (2010).

- Assessment of the quality of meta-inferences presented in Chapter 8, which involve:

    - Addressing validation of the quantitative strand and potential threats and remedies - see Chapter 8: section 8.4.1.

    - Addressing validation of the qualitative strand and potential threats and remedies - see Chapter 8: section 8.4.2; the narrative schema developed in Chapter 7 was validated with the project team to ensure appropriate reflection of their perception of the events that took place - i.e. 'member checking' as contended by Maxwell (1992).

    - Addressing validation of meta-inference(s) and potential threats and remedies - see Chapter 8: section 8.4.3.

Addressing validation from a research design point of view - see Chapter 8: section 8.4.3.

# 5 Quantitative approach

## 5.1 Introduction

This chapter describes the data collection and analysis of the quantitative (QUAN) approach and presents its findings. These are the first and second steps in the research process outlined in Chapter 4: Figure 4.1 and section 4.4.1. The chapter attempted to answer the first research question:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

To do so, the characteristics of the three ABC projects investigated (Project 1, 2, and 3) are presented first, followed by a description of the QUAN analysis. This includes the illustration of the approach through the analysis of the data for Project 1, followed by the results of analysing the remaining projects. Finally, the works in this chapter is summarised.

## 5.2 Project characteristics

This section provides the context to the three projects investigated, identifies their similarities and differences, and describes their dependency during project execution.

### 5.2.1 Context of the three Projects

All three projects were carried out at about the same time, within a range of two years - after 2010, for the same client. The ABC programme management, senior managers, and managers were very familiar with the application domain and had an established relationship of 10 years standing with the client providing software development and system integration services.

**Project 1**

This developed the service middleware layer shown in Figure 3.1 (Chapter 3). This software existed between two layers, one which requested services of the application, and another called to carry out services. Although, ABC had a long relationship with the client through many software development and integration programmes, Project 1 was the first release of a newly started programme; and the Designers, Developers, and Testers lacked the application domain expertise and experience of handling the relationship with this particular Client.

**Project 2**

Project 2 followed on from Project 1; building on the foundational service components developed in Project 1, it added new services and amended others already developed (but not yet released) in Project 1. The two projects were run by the same Project manager, phase managers, and team members. It was hoped Project 2 would benefit from the experience, relevant contextual knowledge and technical skills gained in Project 1. It was reasonable to expect speedier delivery and a better quality product. However, this was not always the case.

Project 2 started in the middle of Project 1, so development teams were for a time working on the two projects simultaneously. As will be seen, the parallel working resulting from (possibly) aggressive task compression, added a range of conflicting resource constraints.

**Project 3**

Project 3 developed the process middleware layer shown in Figure 3.1 (Chapter 3). Some software components were custom developed, others configured from a package application. Some Project 3 components depended on the components/services developed by Projects 1 and 2. Project 3 had a different project manager, phase managers, but all experienced in the technology and familiar with the client from other projects.

### 5.2.2 Similarities and differences among the three projects

Table 5.1 shows the acronyms used for various project phases in this chapter, followed by Tables 5.2 and 5.3 describing the similarities and difference among the three projects respectively.

| **Acronym** | **Phase name** |
|---|---|
| FD | Functional design |
| FD/TD Transition | Functional design to technical design transition |
| TD | Technical design |
| Code | Programming |
| AT | Assembly test |
| IT | Integration test |
| DBT | Design, Build, and Test |

**Table 5.1 Phase acronym**

| # | Characteristic | Project 1 | Project 2 | Project 3 |
|---|---|---|---|---|
| 1 | Project composition (in phases) | FD; FD/TD Transition; TD; Code; AT; IT | FD; FD/TD Transition; TD; Code; AT; IT | TD; Code; AT; IT |
| 2 | Level of reporting progress performance | Phase: e.g. IT | Phase | Increment: e.g. DBT |
| 3 | Progress tracking | Phase | Phase | Increment |
| 4 | Nature of functionality | New | New and modified | New and modified |
| 5 | Platform | Service middleware | Service middleware | Process middleware |
| 6 | Delivery method | Custom development | Custom development | Custom and package development |
| 7 | Delivery duration (weeks)<br>(months - approx.) | 35<br>(9) | 38<br>(9.5) | 21<br>(5) |
| 8 | User of the project's deliverables | Client and the Consumer layer shown in Figure 3.1 (Chapter 3) | Client and the Consumer layer shown in Figure 3.1 (Chapter 3) | Projects 1 & 2, and the Consumer layer shown in Figure 3.1 (Chapter 3) |
| 9 | Research focus<br>(project phase) | IT- Execution - with Authentication Tool (Inc1)<br><br>IT - Execution - without Authentication Tool (Inc1); | Integration Test - Plan & Preparation;<br><br>Integration Test - Execution | DBT (Inc6) |

| # | Characteristic | Project 1 | Project 2 | Project 3 |
|---|---|---|---|---|
| | | IT (Inc4) | | |
| 10 | Size of DBT team (number of people) - approx. | 15 | 25 | 8 |
| 11 | Size of management team (number of people - approx.); i.e. Project managers and Design, Build, and Test managers | 5 | 7 | 3 |

**Table 5.2 Differences among the three projects**

| # | Characteristic | Comments |
|---|---|---|
| 1 | Management team (Project manager and phase managers) | Projects 1 and 2 had the same management team<br>Project 3 had a different management team from Projects 1 and 2 |
| 2 | Project setup | Projects 1 and 2 were different releases of the same project; they are called as such for easier distinction in this research. Projects 1, 2 and 3 belonged to the same programme; and had 4, 1, and 8 increments respectively |
| 3 | Organisation | All three projects were carried out within the same organisational unit in ABC, delivering application systems to the same external Client |

| # | Characteristic | Comments |
|---|---|---|
| 4 | Project execution chronology | Project 2 started in the middle of, and overlapped with, Project 1. Project 3 was executed at the same time as both projects 1 and 2 |
| 5 | Integration dependency | Project 2 and 3 were dependent on Project 1 for functional integration of the products |
| 6 | Delivery method | All projects used the ABC's proprietary delivery method |
| 7 | Delivery model | All projects employed onshore (UK)/offshore (India) delivery model where the Design was done onshore; Build and Test were largely done offshore with onshore Build and Test management |
| 8 | Supporting functions | All projects drew from shared/same supporting functions in ABC. These functions were the various teams within the Product Line Development process described in section 3.3 (Chapter 3) |
| 9 | Relationship among the projects | Project 2 was the next release of Project 1; developing the Service middleware layer of the application architecture. Project 3 was developing/amending a different layer from Projects 1 and 2; the Process middleware layer |
| 10 | Frequency of project performance report | All projects reported progress on weekly basis |
| 11 | Type of product | All projects were delivering large software system within an enterprise architecture environment |
| 12 | Driver of delivery duration | Project end date driven - imposed by client |

**Table 5.3 Similarities among the three projects**

### 5.2.3 Dependency among the three projects during project execution

The three projects developed parts of an integrated product (as described in section 3.2.1 - Chapter 3,); they scheduled specific milestones to align their developed artefacts during project execution for easy integration afterwards. This involved submitting/releasing at agreed times project artefacts produced by the projects that contribute to the integrated end product - see Figure 5.1 (dotted lines) for the key events taking place.

The Client submitted the system requirements to the Peer supplier and ABC (Projects 1 & 2) teams who carried out their Design, Build, and Test activities. Figure 5.1 shows ABC project 1 and project 2 within one area for easy reading; however they were separate projects as described in Tables 5.2 and 5.3. The Peer supplier released their initial interface definitions for their software to ABC (Projects 1 & 2) at the end of their Design stage, and their final interface definitions after their Test phase. The ABC Projects 1 & 2 aligned their project artefacts, under development, with the received Peer supplier artefacts.

The ABC Projects 1 & 2 released a number of artefacts, at the end of their project phases, to ABC Project 3 (which were received at specific points in ABC Project 3 phases as indicated by the position of the arrows in Figure 5.1) including: their FD at the end of the FD/TD Transition phase; their TD and initial interface definitions at the end of the TD phase; their updated FD, TD, and interface definitions at the end of each of the Code and Assembly Test phases, and their final FD, TD, and interface definitions at the end of the Integration Test phase. The ABC Project 3 aligned their project artefacts, under development, with the received ABC Projects 1 & 2 artefacts.

The complexity of interdependency among the three projects can be seen in the dependency of a project on the preceding project for delivering of the needed artefacts in the agreed times. For example, delays in the Peer supplier provision of the initial or final interface definitions would affect progress of work activities in Projects 1 & 2, which in turn affect the progress of Project 3 activities. The chain of delays may have constrained project progress.

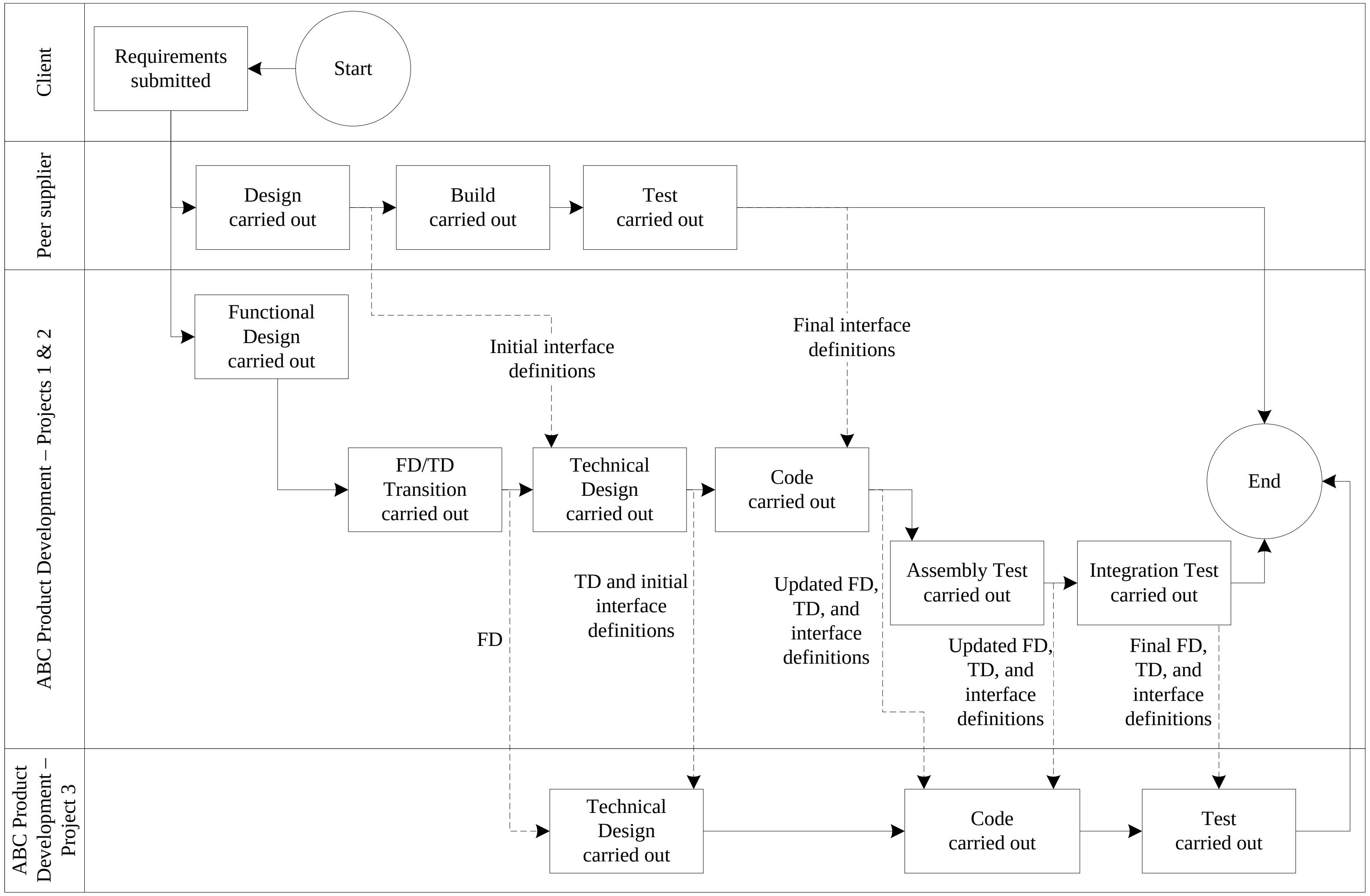

Client
Requirements submitted
Start
Peer supplier
Design carried out
Build carried out
Test carried out
ABC Product Development – Projects 1 & 2
Functional Design carried out
Initial interface definitions
Final interface definitions
FD/TD Transition carried out
Technical Design carried out
Code carried out
End
Assembly Test carried out
Integration Test carried out
FD
TD and initial interface definitions
Updated FD, TD, and interface definitions
Updated FD, TD, and interface definitions
Final FD, TD, and interface definitions
ABC Product Development – Project 3
Technical Design carried out
Code carried out
Test carried out

**Figure 5.1 Dependency among the three projects**

## 5.3 Research process

This section describes the QUAN analysis, using the Project 1 data to illustrate the approach. Note that the key ‘Research data’ in Figure 5.2 refers to the management information collected by project participants for tracking project progress in ABC, rather than data specifically being collected for this research.

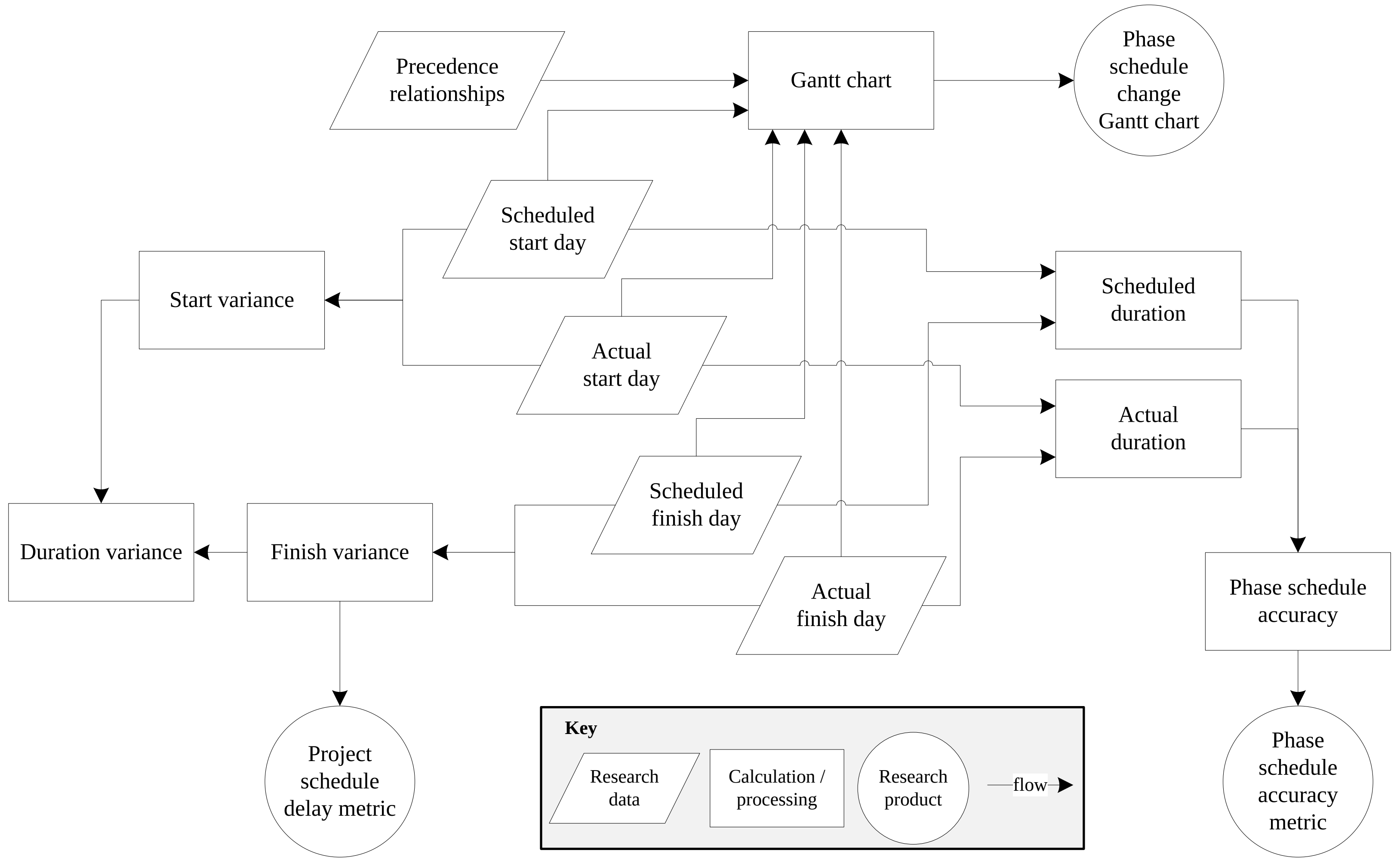


**Figure 5.2 Approach to QUAN analysis**

### 5.3.1 Sampling strategy

The approach to sampling for the QUAN analysis was non-random/purposive (Kumar, 2011; Robson, 2011). The choice of the three projects investigated was primarily based on the availability of the data, and time constraints, but they were also perceived as being representative of ABC developments.

### 5.3.2 The data collected

This research examined ABC's project management information, used for tracking the performance of their projects, rather than collecting data especially for this research. The QUAN approach extracted the project phases' planned and actual start and end dates and precedence relationships from the weekly project performance reports (described in section 3.4 - Chapter 3). Five base variables were created to hold these values - see Table 5.4.

| **Base variable** | **Definition** | **Source data (Event tracking sheet section 3.4)** |
|---|---|---|
| Scheduled start day | The day number which the project/phase was scheduled to start | 'Planned occurrence date' of the first event of the phase |
| Actual start day | The day number which the project/phase actually started | 'Actual occurrence date' of the first event of the phase |
| Scheduled finish day | The day number which the project/phase was scheduled to finish | 'Planned occurrence date' of the last event of the phase |
| Actual finish day | The day number which the project/phase actually finished | 'Actual occurrence date' of the last event of the phase |
| Precedence relationships | The type of the dependency relationships of the phase | First event of the phase |

**Table 5.4 Base variables**

The collected data items were entered into a spreadsheet (Microsoft Excel) used for subsequent analysis. Having checked for correct transcription, they were used to calculate the schedule metrics described in the next section. Tables 5.5, 5.6, and 5.7

show the schedule data for Project 1, 2, and 3 respectively. It may be noted that a start day of 0 means that the project started at the end of day 0 (i.e. the start of day 1).

| # | Project 1 Phases | Scheduled start day | Actual start day | Scheduled finish day | Actual finish day |
|---|---|---|---|---|---|
| 1 | FD (Inc1) | 0 | 0 | 20 | 44 |
| 2 | FD (Inc2) | 49 | 49 | 60 | 63 |
| 3 | FD (Inc3) | 74 | 86 | 95 | 98 |
| 4 | FD/TD Transition (Inc1) | 21 | 21 | 34 | 39 |
| 5 | FD/TD Transition (Inc2) | 60 | 60 | 65 | 70 |
| 6 | TD (Inc1) | 35 | 35 | 60 | 60 |
| 7 | TD (Inc2) | 70 | 70 | 78 | 77 |
| 8 | Code (Inc1 & Inc2) | 36 | 37 | 90 | 109 |
| 9 | FD/TD Transition+TD+Code (Inc3) | 98 | 118 | 132 | 144 |
| 10 | TD and Code (Inc4) | 165 | 165 | 172 | 178 |
| 11 | AT - Plan & Preparation (Inc1) | 27 | 35 | 90 | 90 |
| 12 | AT - Execution (Inc1) | 91 | 91 | 118 | 125 |
| 13 | IT- Execution - with Authentication Tool (Inc1) | 119 | 126 | 144 | 166 |
| 14 | IT - Execution - without Authentication Tool (Inc1) | 142 | 142 | 151 | 166 |
| 15 | IT (Inc4) | 168 | 178 | 189 | 241 |

**Table 5.5 Project 1 schedule data**

| # | Project 2 Phases | Scheduled start day | Actual start day | Scheduled finish day | Actual finish day |
|---|---|---|---|---|---|
| 1 | FD | 0 | 0 | 55 | 56 |
| 2 | FD/TD Transition | 55 | 55 | 65 | 65 |
| 3 | TD | 67 | 67 | 107 | 107 |
| 4 | Build | 106 | 105 | 135 | 146 |
| 5 | Assembly Test - Plan & Preparation | 103 | 103 | 130 | 186 |
| 6 | Assembly Test - Execution | 138 | 138 | 165 | 209 |
| 7 | Integration Test - Plan & Preparation | 137 | 137 | 165 | 253 |
| 8 | Integration Test - Execution | 165 | 209 | 214 | 268 |

**Table 5.6 Project 2 schedule data**

| # | Project 3 Increments | Scheduled start day | Actual start day | Scheduled finish day | Actual finish day |
|---|---|---|---|---|---|
| 1 | DBT (Inc1) | 0 | 0 | 112 | 105 |
| 2 | DBT (Inc2) | 0 | 0 | 112 | 119 |
| 3 | DBT (Inc3) | 81 | 109 | 116 | 146 |
| 4 | DBT (Inc4) | 108 | 119 | 136 | 146 |
| 5 | DBT (Inc5) | 136 | 147 | 185 | 174 |
| 6 | DBT (Inc6) | 31 | 34 | 66 | 112 |
| 7 | DBT (Inc7) | 108 | 108 | 136 | 167 |
| 8 | DBT (Inc8) | 136 | 147 | 185 | 187 |

**Table 5.7 Project 3 schedule data**

It may be noted that, in Project 3 progress was tracked at the level of increments rather than phases and increments. These increments all followed the same ABC DBT lifecycle model used in the previous 2 projects.

### 5.3.3 Techniques used: Schedule metrics

The QUAN analysis measured three attributes of schedule: delay, change, and accuracy (Kitchenham et al. 1995). The schedule metrics: Project schedule delay, Phase schedule change, and Phase schedule accuracy each measured an attribute of the project schedule at the project and phase levels of analysis. This was based on the composition of a project within the Product Development in ABC where a project comprised multiple phases as was described in Figure 3.5 (section 3.2.3 - Chapter 3). Table 5.8 summarises the schedule metrics, described in more depth in the next section.

It is worth noting that the research had also investigated the project schedule at event level, examining the trend of achieving weekly event targets within project phases. This involved producing trend charts of SPI (Schedule performance index), the values of which were originally calculated by ABC to track progress of their projects, to compare achieving event targets against the planned targets. This was motivated by the fact that SPI was used by ABC as the key measure for monitoring and controlling schedule progress - though it was a locally-tailored SPI (see section 3.4.4 - Chapter 3). However, proceeding further with this line of enquiry was unfruitful due to: the limitations of SPI in providing accurate measure of schedule progress (see section 2.2.3 - Chapter 2); the SPI indicating phase completed, and not phase completed on time (i.e. phases having 1.0 SPI at end but overrun on duration recorded); and time constraints.

| Entity | Attribute | Definition of attribute | Unit of measurement | Schedule metric (acronym) | Technique | Unit of analysis | Description of unit of analysis |
|---|---|---|---|---|---|---|---|
| Schedule | Delay | The degree of mismatch between the scheduled duration of a project from its actual duration | Count of number of days | Project schedule delay (PSD) | Calculate Project schedule delay metric to measure extent of delay | Project | A software project encompassing all the phases within it as a whole |
| Schedule | Change | The degree of mismatch between the precedence relationships of a phase from its actual relationships | Precedence relationship type | Phase schedule change (PSC) | Develop Gantt chart that shows planned and actual schedule with their precedence relationships | Phase | A phase within the software project - this is a subset of the Project level of analysis |
| Schedule | Accuracy | The degree of mismatch between the estimates of the duration of a phase from its actual duration | Percentage value (days) which the duration was over or under estimated | Phase schedule accuracy (PSA) | Use the conventional measure Magnitude of relative error (MRE) | Phase | A phase within the software project - this is a subset of the Project level of analysis |

**Table 5.8 Summary of schedule metrics**

#### 5.3.3.1 Project schedule delay

This metric measured the extent of delay in the project schedule.

The Project schedule delay (PSD) metric used four derived variables (below) to calculate schedule delay. These variables were based on the first four base variables described in Table 5.4. Definitions of the derived variables are provided in Table 5.9. The PSD metric used Microsoft Excel to calculate and present the tabular data:

Start variance = actual start day – scheduled start day (Equation 5.1)

Finish variance = actual finish day – scheduled finish day (Equation 5.2)

Duration variance = Finish variance – Start variance (Equation 5.3)

Project schedule delay = Finish variance of the last phase in the project (Equation 5.4)

| **Derived variable** | **Acronym** | **Definition** |
|---|---|---|
| Start variance | STV | The number of days which the actual start day of a phase/project differs from its scheduled start day. A positive value of STV indicates that the phase was started later than scheduled; a negative value indicates that the phase was started earlier than scheduled; and an STV value of zero indicates that the phase was started as scheduled |
| Finish variance | FV | The number of days which the actual finish day of a phase/project differs from its scheduled finish day. A positive value of FV indicates that the phase was finished later than scheduled; a negative value of FV indicates that the phase was finished earlier than scheduled; and an FV value of zero indicates that the phase was finished as scheduled |
| Duration variance | DV | The number of days which the actual duration of a phase/project differs from its scheduled duration. A positive value of DV indicates that the phase's actual duration was longer than its scheduled duration; a negative value indicates that the phase's actual duration was shorter than its scheduled duration; and a DV value of zero indicates that the phase's actual duration was the same as its scheduled duration |
| Project schedule delay | PSD | The number of days which a project schedule was delayed. PSD is equal to the FV of the last phase in the project (i.e. the phase with the latest actual finish day). A positive value of PSD indicates that the project was delivered late; a negative value of PSD indicates that the project was delivered early; and PSD value of zero indicates that the project was delivered on schedule |

**Table 5.9 Project schedule delay variables**

The project schedule delay (equations 5.1 - 5.3 above) was calculate for each phase of Project 1, 2, and 3 to give the number of days each phase contributed to the overall delay of the project. The results of these calculations can be seen in Tables 5.10, 5.11, and 5.12 for Project 1, 2, and 3 respectively.

| # | Project 1 Phases | Start variance (Actual start day - Scheduled start day) | Finish variance (Actual finish day - Scheduled finish day) | Duration variance (Finish variance - Start variance) |
|---|---|---|---|---|
| 1 | FD (Inc1) | 0 | 24 | 24 |
| 2 | FD (Inc2) | 0 | 3 | 3 |
| 3 | FD (Inc3) | 12 | 3 | -9 |
| 4 | FD/TD Transition (Inc1) | 0 | 5 | 5 |
| 5 | FD/TD Transition (Inc2) | 0 | 5 | 5 |
| 6 | TD (Inc1) | 0 | 0 | 0 |
| 7 | TD (Inc2) | 0 | -1 | -1 |
| 8 | Code (Inc1 & Inc2) | 1 | 19 | 18 |
| 9 | FD/TD Transition+TD+Code (Inc3) | 20 | 12 | -8 |
| 10 | TD and Code (Inc4) | 0 | 6 | 6 |
| 11 | AT - Plan & Preparation (Inc1) | 8 | 0 | -8 |
| 12 | AT - Execution (Inc1) | 0 | 7 | 7 |
| 13 | IT- Execution - with Authentication Tool (Inc1) | 7 | 22 | 15 |
| 14 | IT - Execution - without Authentication Tool (Inc1) | 0 | 15 | 15 |
| 15 | IT (Inc4) | 10 | 52 | 42 |

**Table 5.10 Project 1 schedule delay metrics**

The overall project schedule delay, in this case 52 days, is equal to the finish variance of the last phase in the project, (IT Inc4, the Increment 4 Integration Test). This is the difference between when the last activity was scheduled to finish and when it actually finished. The table shows that the phase with the largest delay was also the Integration Test phase (#15) which was 10 days late starting and then contributed a further 42 days of delay.

| # | Project 2 Phases | Start variance (Actual start day - Scheduled start day) | Finish variance (Actual finish day - Scheduled finish day) | Duration variance (Finish variance - Start variance) |
|---|---|---|---|---|
| 1 | FD | 0 | 1 | 1 |
| 2 | FD/TD Transition | 0 | 0 | 0 |
| 3 | TD | 0 | 0 | 0 |
| 4 | Build | -1 | 11 | 12 |
| 5 | Assembly Test - Plan & Preparation | 0 | 56 | 56 |
| 6 | Assembly Test - Execution | 0 | 44 | 44 |
| 7 | Integration Test - Plan & Preparation | 0 | 88 | 88 |
| 8 | Integration Test - Execution | 44 | 54 | 10 |

**Table 5.11 Project 2 schedule delay metrics**

| # | Project 3 Increments | Start variance (Actual start day - Scheduled start day) | Finish variance (Actual finish day - Scheduled finish day) | Duration variance (Finish variance - Start variance) |
|---|---|---|---|---|
| 1 | DBT (Inc1) | 0 | -7 | -7 |
| 2 | DBT (Inc2) | 0 | 7 | 7 |
| 3 | DBT (Inc3) | 28 | 30 | 2 |
| 4 | DBT (Inc4) | 11 | 10 | -1 |
| 5 | DBT (Inc5) | 11 | -11 | -22 |
| 6 | DBT (Inc6) | 3 | 46 | 43 |
| 7 | DBT (Inc7) | 0 | 31 | 31 |
| 8 | DBT (Inc8) | 11 | 2 | -9 |

**Table 5.12 Project 3 schedule delay metrics**

#### 5.3.3.2 The behaviour of schedule delay

- Project 1 was late by 52 days; i.e. 27% of the original schedule (190 days). This compares with the average 20% schedule delay that Sauer et al. (2007) found in a survey of 412 UK based IT projects.

- As noted about the largest contribution to delay in Project 1 was phase #15 - IT (Inc4). This suggests that particular attention needs to be given to this phase obstructing progress.

- The second largest contributor to delay in Project 1 was the first phase #1 Functional Design (Increment 1). By overlapping activities and actually finishing some phases in less than the scheduled durations (see negative duration variances in Table 5.10), the initial delay of 24 days was reduced to just 10 by start of phase #15.

- Project 2 was late by 54 days i.e. 25% of the original schedule (214 days). This compares to 27% for Project 1, when the greater domain and technical expertise might have been expected after Project 1. This also compares with Rainer's PhD thesis, investigating two IBM projects and finding that both projects finished later than scheduled (Rainer 1999, page: 128, 2010, 2011).

- The largest delaying phase in Project 2 was phase #7 (Integration Test - Plan & Preparation). Integration Test phase once again became the point in the project when most delay was experienced (see second bullet point). However, in this case it was the plan and preparation phase, in contrast to the Integration Test *execution* phase in Project 1. The former is concerned with preparing and planning for the subsequent Integration Test execution phase #8 in Project 2.

- Although the last phase #8 in Project 2 was late by 10 days, the accumulation of lateness in the preceding phases meant that phase #8 started with a variance of 44 days later than scheduled, leading to overall project delay of 54 days in total. It was possible for Phase #8 catch up some of the delay of 88 days incurred by phase #7 by overlapping the execution of the two phases, rather than carrying them out one after the other.

- Project 3 was late by 2 days; i.e. 1% of the original schedule (185 days). This project was pretty well on schedule compared to Projects 1 and 2 with 27% and

25% delays respectively. This is compared with Sauer et al. (2007)' findings that the risk (probability) of not achieving scheduled targets decreases for shorter projects (though this does not mean that shorter projects exhibit less percentage delay); Project 3 duration was shorter than Project 1, which in turn was shorter than Project 2. One reason for Project 3's better schedule performance may be that subsequent increments were planned to start as soon as the preceding increments start (see Figure 5.5 - next section).

- The increment with the biggest over-run in Project 3 was phase #6 - DBT (Inc6), but did not actually delay the project because it was not on the critical path - no other increment was dependent on it (see Figure 5.5 - next section).

- Although the last phase #8 in Project 3 started with a variance of 11 days from the accumulation of lateness in the preceding phases, it finished 9 days earlier than scheduled, leading to the overall project delay of only 2 days.

#### 5.3.3.3 Phase schedule change

This metric examined the change of precedence relationships of project phases.

The Phase schedule change (PSC) metric used Gantt charts (PMI, 2008), widely used for project scheduling and control, to enable visual examination of the changes in the precedence relationships among project phases. Gantt charts enabled the comparison of planned and actual phase relationships on one diagram. The research produced a Gantt chart for each of the three projects, using Microsoft Project, based on the five base variables described in Table 5.4.

Activity dependencies (precedence relationships) in a project schedule may be one of four types (Lockyer & Gordon, 2005):

- Finish-to-start (FS), the succeeding activity may not start until the preceding activity has finished.

- Start-to-start (SS), the succeeding activity may not start until the preceding activity has started.

- Start-to-finish (SF), the succeeding activity may not finish until the preceding activity has started.

- Finish-to-finish (FF), the succeeding activity may not finish until the preceding activity has finished.

The Gantt chart for Project 1, 2, and 3 are shown in Figures 5.3, 5.4, and 5.5 respectively.

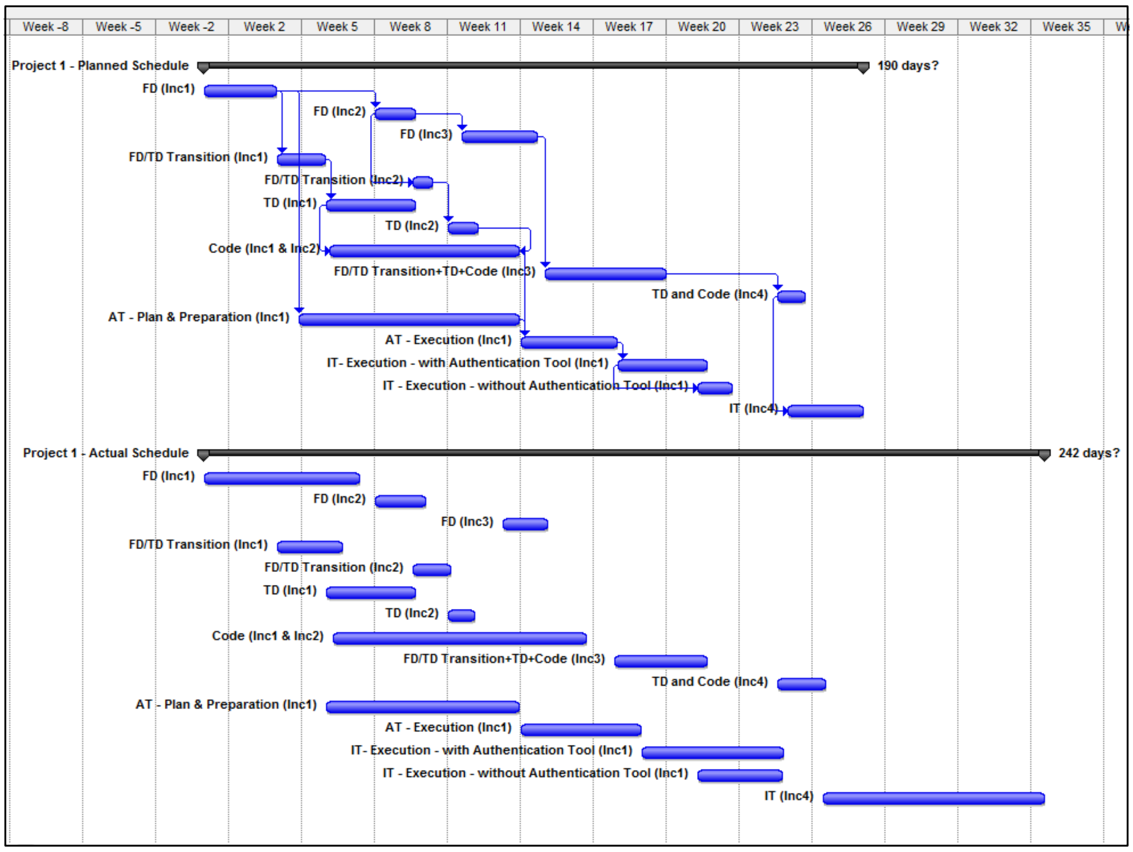


**Figure 5.3 Project 1 Gantt chart**

Figure 5.3 shows the planned schedule prior to execution and the actual schedule post execution. An arrow flowing down represents precedence/dependency of the succeeding phase on the preceding one. Earlier in this section it was noted that, activity precedence can be of type Finish-to-Start, where the subsequent phase should only start when the preceding phase was completed. However, these precedence rules tend to be ignored in practice during project execution and activities/phases overlap due to time constraints, this may influence schedule duration since it potentially involves rework of the activities that were overlapped by the succeeding activity.

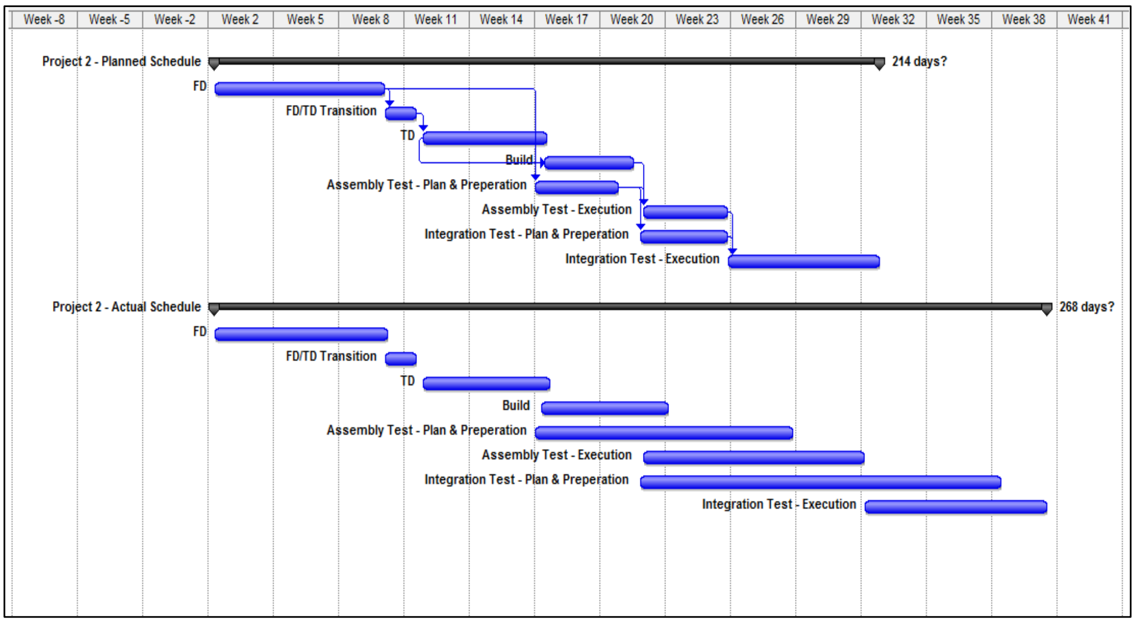


**Figure 5.4 Project 2 Gantt chart**

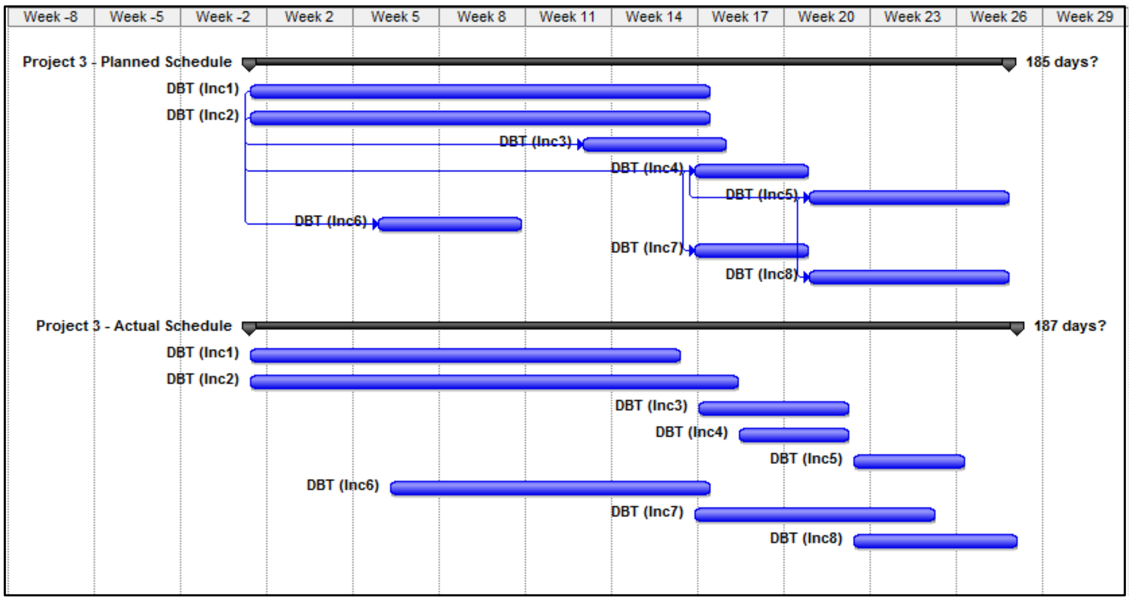


**Figure 5.5 Project 3 Gantt chart**

#### 5.3.3.4 The behaviour of schedule change

The Project 1 Gantt chart (Figure 5.3) indicates that some of the Finish-to-Start (FS) constraints were ignored during project execution; that is, phases planned to be executed sequentially overlapped during execution - see #4, #11 and #12 in Table 5.13. This is compared with Rainer (1999) where the Design, Build, and Test phases of the IBM projects, although planned to occur sequentially (i.e. FS relationship), actually overlapped during execution (page: 128). The practice of overlapping phases appears to have introduced rework (see #4 in Table 5.13). Furthermore, the schedule behaviour of the critical phases (i.e. #1, #2, #3, #9, #10, and #15) indicate that their dependency on one another partly contributed to the overall delay of the project (see phase #15). The majority of the delay was contributed by the phase #15. Thus, it would be of interest to examine the textual reports of phase #15 closely to understand the causes of delay. This was done in Chapter 7.

The Project 2 Gantt chart (Figure 5.4) shows that some of the FS constraints in Project 2 have been ignored during project execution - see phases #6, #7, #8 in Table 5.14. The critical phases (#4, #5, #6, #7, and #8) all finished late. These were mainly due to dependency on external factors (Peer supplier), quality of the code produced (due to resource shortages), and transferring outstanding work (due to delay from Peer supplier) to succeeding phases. Thus, the data does not suggest that delay in these phases were due to their overlapping behaviour, it was rather the dependency on external actors and quality of the work in previous phases that caused the delay in the project. Although the textual information does not report how much rework was undertaken as a result of the overlaps, it is not unreasonable to assume some reworking to have taken place given the large number of defects discovered (see #8 in Table 5.14). The most contributing phase to project delay (#7) was affected by its predecessor, and it affected the succeeding phase too, so it would be of interest to examine its textual information in-depth to understand what might have happened - which was done in Chapter 7.

The Project 3 Gantt chart (Figure 5.5) shows that the precedence relationships among all the increments were of type Start-to-Start (SS). Table 5.15 shows that most of the increments finished late, but because the SS relationships, work started in the succeeding increments without waiting for the predecessor to finish, and that the delays did not affect the final increment. The delay in the final increment (i.e. project delay) was due to unclear change requests. Table 5.15 shows several issues facing phase #6 (the most delayed increment); and it would of interest to examine the textual reports closely to understand what may have caused the delay - which is done in Chapter 7.

| # | Project 1 Phases | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| 1 | FD (Inc1) | | 0 | 24 | 24 | This phase finished later than scheduled due to lack of client input, fluctuating scope, and undefined requirements during phase execution. |
| 2 | FD (Inc2) | 1 FS | 0 | 3 | 3 | Finished later than scheduled; however, it did not overlap with its predecessor (see Figure 5.3). The cause of delay was lack of requirements definition. |
| 3 | FD (Inc3) | 2 FS | 12 | 3 | -9 | Finished later than scheduled; however, it did not overlap with its predecessor (Figure 5.3). The phase finished late due to unconfirmed scope. |
| 4 | FD/TD Transition (Inc1) | 1 FS | 0 | 5 | 5 | Overlapped with its predecessor. The textual reports of this phase indicate the need to rework the transitioned, but not yet signed-off, FDs; i.e. due to this overlap, which explains the 5 day delay in finishing the phase. It is also reported that this phase's progress is delayed due to competing priorities within the FD team resulting in delay in making FD updates as requested by the TD team. |

| # | Project 1 Phases | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| 5 | FD/TD Transition (Inc2) | 2 SS | 0 | 5 | 5 | Although it started as soon as its predecessor started, it still finished late, because the FD team delayed making changes requested by the TD team. |
| 6 | TD (Inc1) | 4 FS | 0 | 0 | 0 | Overlapped with its predecessor. The TDs will almost certainly need to be reworked when the FDs are finally transitioned. However, the TD (Inc1) finished on schedule; hence, any rework may have been carried out through working overtime. |
| 7 | TD (Inc2) | 5 FS | 0 | -1 | -1 | |
| 8 | Code (Inc1 & Inc2) | 6 SS, 7 FF | 1 | 19 | 18 | The dependencies of this phase on its predecessors were such that a TD can be coded as soon as it was completed; i.e. without waiting for all the TDs to complete. However, the phase completed late due to resource constraints and Technical environment issues. |
| 9 | FD/TD Transition+TD+Code (Inc3) | 3 FS | 20 | 12 | -8 | Finished later than scheduled; however, it did not overlap with its predecessor (see Figure 5.3). Unfortunately, no progress reports exist for this phase to be examined. |

| # | Project 1 Phases | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| 10 | TD and Code (Inc4) | 9 FS | 0 | 6 | 6 | Finished later than scheduled; however, it did not overlap with its predecessor (see Figure 5.3). The textual reports indicate that the Unit test environment was unavailable to test the developed codes until the last week of executing the phase, which caused the delay. |
| 11 | AT - Plan & Preparation (Inc1) | 1 FS | 8 | 0 | -8 | Overlapped with its predecessor; however, delay in starting the phase was due to unexpected resource absence. |
| 12 | AT - Execution (Inc1) | 11 FS, 8 FS | 0 | 7 | 7 | Overlapped with its predecessor #8, and finished later than scheduled due to delays in providing the code in #8 as well as Test environment issues. The phase finished with 21% of the scenarios being transitioned to #13 due to being blocked by issues (the reports do not specify what these issues were). |
| 13 | IT- Execution - with Authentication Tool (Inc1) | 12 FS | 7 | 22 | 15 | Started later, not because of its predecessor (#12) finishing late but because of delay in providing Test environment by the Technical environment manager. |
| 14 | IT - Execution - | 13 SS | 0 | 15 | 15 | Finished late because of Technical environment unavailability. |

| # | Project 1 Phases | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| | without Authentication Tool (Inc1) | | | | | |
| 15 | IT (Inc4) | 10 SS | 10 | 52 | 42 | Started late due to Test environment unavailability and delay providing code from #10 due to Unit test delays in #10. Thus, the precedence relationship contributed partly to the delay in starting this phase, because #10 was unable to provide early code for testing in this phase (it had it is own problems - see #10). The phase finished considerably late (52 days), the majority of the delay in this phase was due to its own problems; unavailability of integration test environment and discovery of code defects, as reported in the textual reports. |

**Table 5.13 Project 1 behaviour of schedule change**

| # | Project 2 Phases | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| 1 | FD | | 0 | 1 | 1 | |
| 2 | FD/TD Transition | 1 FS | 0 | 0 | 0 | |
| 3 | TD | 2 FS | 0 | 0 | 0 | |
| 4 | Build | 3 SS | -1 | 11 | 12 | Despite starting ahead of schedule, this phase still finished late; mainly due to resource shortage, but also due to schedule not being baselined and Code deploy tool issues. |
| 5 | Assembly Test - Plan & Preparation | 1 FS | 0 | 56 | 56 | Finished considerably late mainly due to dependency on the Peer supplier to carry out their Test data build. The phase closed transferring all test scenarios, which data cannot not be prepared for them (due to Peer supplier delay in Test data build), to phase (#7). |
| 6 | Assembly Test - Execution | 5 FS, 4 FS | 0 | 44 | 44 | Overlapped with both of its predecessors (#5 and #4). The phase inherited the delay from phase #5 related to providing Peer supplier Test data build. The phase also identified large number (40) of code defects (phase #4), some of which were Peer supplier code defects. The phase closed transferring a number of the |

| # | Project 2 Phases | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| | | | | | | outstanding defects to phase (#8). |
| 7 | Integration Test - Plan & Preparation | 5 FS | 0 | 88 | 88 | Overlapped, and ran in parallel for some time, with its predecessor (#5). Finished significantly late mainly due to inheriting delays from phase #5; i.e. the Peer supplier's Test data build, which was resolved by the end of the phase. |
| 8 | Integration Test - Execution | 7 FS, 6 FS | 44 | 54 | 10 | Overlapped with #7. Finished considerably late due to inheriting several outstanding code defects from #6 delaying its progress, and suffered from delays in Peer supplier integration Test data build. The phase also identified large number of code defects (94), as well as receiving a number of change requests during phase execution. |

**Table 5.14 Project 2 behaviour of schedule change**

| # | Project 3 Increments | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| 1 | DBT (Inc1) | | 0 | -7 | -7 | |
| 2 | DBT (Inc2) | | 0 | 7 | 7 | |
| 3 | DBT (Inc3) | 2 SS, 1SS | 28 | 30 | 2 | Started late due to resource constraints (who were working on both the predecessor increments), and which #2 was delayed. Scope increased during execution and weekend work was carried out to recover. The phase still finished late; however, it did not affect the project end date because no later increment was dependent on it. |
| 4 | DBT (Inc4) | 1 SS | 11 | 10 | -1 | Started and late due to resource constraints, but recovered due to weekend work. Finished late due to technical challenges in developing one of the components. |
| 5 | DBT (Inc5) | 4 SS | 11 | -11 | -22 | Started late due to resource constrains. Finished earlier because some of the change requests turned out not impacting this increment. |
| 6 | DBT (Inc6) | 2 SS | 3 | 46 | 43 | Finished late due to numerous issues: Technical environment issues, Unit test delays, which in turn caused by Technical environment issues, Build environment unavailability, and Design gaps. However, phase delay did not affect the project end date because no other increment was dependent on it |

| # | Project 3 Increments | Predecessor # and type of dependency | Start variance | Finish variance | Duration variance | Schedule change behaviour |
|---|---|---|---|---|---|---|
| 7 | DBT (Inc7) | 4 SS | 0 | 31 | 31 | Finished late due to resource constraints, technical challenges in developing some components, and Technical environment issues; however, the delay did not affect the project end date because it was not on the critical path |
| 8 | DBT (Inc8) | 5 SS | 11 | 2 | -9 | Started and finished late due to unclear change requests, which required clarification, being put forward to the increment. |

**Table 5.15 Project 3 behaviour of schedule change**

#### 5.3.3.5 Phase schedule accuracy

This metric measured the degree to which estimate of the duration of a phase differed from its actual duration.

The Phase schedule accuracy (PSA) metric employed the conventional Magnitude of Relative Error (MRE) measure (Conte et al. 1986). MRE, originally a measurement of estimate accuracy, is usually used in research to evaluate the degree of inaccuracy in an estimate of *effort*. MRE is used in this research to determine the extent to which the *duration* estimate of a phase schedule differs from actual duration, which might indicate presence of unforeseen factors delaying execution, or in rarer cases accelerating it. As MRE detects poor estimates that are both over and under estimates, a large MRE does not necessarily mean an activity has been delayed. The low MRE indicates low uncertainty (i.e. more accuracy) in the estimate (Stensrud et al. 2002; Hughes, 2000). MRE measures only the accuracy of a single task. Therefore, researchers more usually use MMRE, which is the average of the MRE, for a range of task estimates. MMRE often measures the effectiveness of an estimating method.

MRE is measured by calculating the difference between the estimate and the actual, of the aspect being measured (e.g. duration), and divide the result by the actual:

MRE = absolute (actual – estimated) / actual (Equation 5.5)

For example, if a project phase was scheduled to complete in 100 days (estimate), and after completion it turned out that it took 150 days to complete (actual), i.e. it took 50 days more than was estimated (underestimate) then:

MRE = absolute (150 – 100) / 150 = 0.3

This indicates that the estimate was wrong by 30%. Thus, MRE can be used to determine the error in estimating the duration of phase schedule, by calculating the difference between the ‘scheduled duration’ - which is effectively an estimate of the duration, and ‘actual duration’, and divide the result by the ‘actual duration’ giving an absolute value, which represents the degree of inaccuracy in estimating the duration of the phase schedule.

A limitation of MRE reported is that, it does not treat underestimates and overestimates equally (Moløkken-Østvold & Jørgensen, 2005). Thus, extending the example above, if the actual duration turned out to be 50 days; i.e. it took 50 days less than was estimated (overestimate), then MRE = absolute (50 – 100) / 50 = 1; indicating that the estimate was wrong by 100%.

The implication is that, whilst in the case where the number of days underestimated is same as the number of days overestimated; one might expect the MRE to be the same; the above example shows that an underestimate by 50 days is not being treated equally with an overestimate by 50 days (i.e. MRE is 0.3 in the former, and 1 in the latter). Whilst, this inequality in treating underestimates and overestimates has been reported as shortcoming of MRE in the reviewed literature in this section, it appears to be a strength. Since, whilst numerically it is acceptable to expect equal treatment of underestimates with overestimates, in reality the underestimate should be treated more seriously (i.e. MRE should read higher) given that a supplier might be unable to fulfil an underestimated contract (Hughes, 2000). Yet, it can be argued that if the estimate was used as the basis of a bid for work, an overestimate could mean that a contract could be lost unnecessarily - which would be serious too. The limitation of MRE is then appear to be in showing a more favourable MRE for the underestimate (0.3) compared to the overestimate (1.0) in the example above; i.e. the underestimate is being treated less seriously.

As a result of the above perceived shortcoming of MRE, an alternative measure have been proposed, called Balanced relative error (BRE) (Miyazaki et al.1991), which is intended to compensate for the MRE's perceived shortcoming above by balancing out the overestimates with underestimates (Moløkken-Østvold & Jørgensen, 2005). However, as can be seen below that the BRE's behaviour, in relation to treating overestimates and underestimates differently, is no different from MRE:

BRE = absolute (actual – estimated) / min (actual, estimated) (Equation 5.6)

Thus, for the underestimate example above BRE = (150 – 100)/ 100 = 0.5

And, for the overestimate example above BRE = (50 – 100)/ 50 = 1.0

Thus, it can be said that using MRE to measure the extent of uncertainty in estimating a single task can produce more reliable measurement, than using it to compare underestimates with overestimates. This research used the MRE for individual project phases in order to identify the phases which are most difficult to estimate duration for, which may indicate causes of schedule delay.

The PSA metric used three derived variables (below), based on the first four base variables in Table 5.4. Definitions of the derived variables are provided in Table 5.16:

Scheduled duration = scheduled finish day – scheduled start day (Equation 5.7)

Actual duration = actual finish day – actual start day (Equation 5.8)

PSA = absolute (actual duration – scheduled duration) / actual duration (Equation 5.9)

| **Derived variable** | **Acronym** | **Definition** |
|---|---|---|
| Scheduled duration | SD | The number of days which a phase/project was planned to expend from start to finish. SD is expected to be a positive value, since the scheduled finish day should always be later than the scheduled start day of a phase/project |
| Actual duration | AD | The number of days which a phase/project actually expended from start to finish. AD is expected to be a positive value, since the actual finish day should always be later than the actual start day of a phase/project |
| Phase schedule | PSA | The percentage value which the duration estimate of |

| Derived variable | Acronym | Definition |
|---|---|---|
| accuracy | | a phase schedule was wrong; this indicates the extent to which the duration of the phase schedule was under-estimated or over-estimated |

**Table 5.16 Phase schedule accuracy variables**

The Scheduled duration and Actual duration variables above relate to duration rather than effort; for example, if there is only one developer working on each activity, then the duration and effort would be more or less the same.

To measure the schedule accuracy for each phase in the three projects, the phase's scheduled and actual durations were calculated first (see equations 5.7 and 5.8). The Phase schedule accuracy measure was then calculated using conventional MRE measure (see equation 5.9). The results are shown in Tables 5.17, 5.18, and 5.19 for Project 1, 2, and 3 respectively. Note that this treats errors in over-estimating duration on the same basis as under-estimating.

| # | Project 1 Phases | Scheduled duration (Scheduled finish day - Scheduled start day) | Actual duration (Actual finish day - Actual start day) | Phase schedule accuracy (PSA = absolute (Actual duration - Scheduled duration) / Actual duration) |
|---|---|---|---|---|
| 1 | FD (Inc1) | 20 | 44 | 0.55 |
| 2 | FD (Inc2) | 11 | 14 | 0.21 |
| 3 | FD (Inc3) | 21 | 12 | 0.75 |
| 4 | FD/TD Transition (Inc1) | 13 | 18 | 0.28 |
| 5 | FD/TD Transition (Inc2) | 5 | 10 | 0.50 |
| 6 | TD (Inc1) | 25 | 25 | 0.00 |
| 7 | TD (Inc2) | 8 | 7 | 0.14 |
| 8 | Code (Inc1 & Inc2) | 54 | 72 | 0.25 |
| 9 | FD/TD Transition+TD+Code (Inc3) | 34 | 26 | 0.31 |
| 10 | TD and Code (Inc4) | 7 | 13 | 0.46 |
| 11 | AT - Plan & Preparation (Inc1) | 63 | 55 | 0.15 |
| 12 | AT - Execution (Inc1) | 27 | 34 | 0.21 |
| 13 | IT- Execution - with Authentication Tool | 25 | 40 | 0.38 |
| 14 | IT - Execution - without Authentication | 9 | 24 | 0.63 |
| 15 | IT (Inc4) | 21 | 63 | 0.67 |

**Table 5.17 Project 1 phase schedule accuracy**

Table 5.17 shows that, the duration estimate of phase #15 was the largest percentage *under-estimation*. Note that, the duration estimate for phase #3 FD (Inc3) was the largest *over-estimation*. This estimate was based on a provisional scope which was then cut by the client, shortening the duration and allowing the phase to start later than scheduled. The textual data of the progress reports indicates unconfirmed scope of this phase during execution:

FD (Inc3): N_01: 'Scope confirmation session held on Wk1 (delayed from two weeks back due to stakeholder availability). Design work will commence from Wk1. No onward impact from scope delays to Inc3 plan. Inc3 TD/Code on track to commence Wk3 – subject to scope confirmation. Code resources due to onboard ready for Wk3.'

It may be noted that Wk1 in the above narrative is relative to the start of FD (Inc3), which is equivalent to week 12 on the overall project timeline (see Figure 5.3). Similarly, Wk3 in the narrative means Wk15 on the Gantt chart.

| # | Project 2 Phases | Scheduled duration (Scheduled finish day - Scheduled start day) | Actual duration (Actual finish day - Actual start day) | Phase schedule accuracy (PSA = absolute (Actual duration - Scheduled duration) / Actual duration) |
|---|---|---|---|---|
| 1 | FD | 55 | 56 | 0.02 |
| 2 | FD/TD Transition | 10 | 10 | 0.00 |
| 3 | TD | 40 | 40 | 0.00 |
| 4 | Build | 29 | 41 | 0.29 |
| 5 | Assembly Test - Plan & Preparation | 27 | 83 | 0.67 |
| 6 | Assembly Test - Execution | 27 | 71 | 0.62 |
| 7 | Integration Test - Plan & Preparation | 28 | 116 | 0.76 |
| 8 | Integration Test - Execution | 49 | 59 | 0.17 |

**Table 5.18 Project 2 phase schedule accuracy**

| # | Project 3 Increments | Scheduled duration (Scheduled finish day - Scheduled start day) | Actual duration (Actual finish day - Actual start day) | Phase schedule accuracy (PSA = absolute (Actual duration - Scheduled duration) / Actual duration) |
|---|---|---|---|---|
| 1 | DBT (Inc1) | 112 | 105 | 0.07 |
| 2 | DBT (Inc2) | 112 | 119 | 0.06 |
| 3 | DBT (Inc3) | 35 | 37 | 0.05 |
| 4 | DBT (Inc4) | 28 | 27 | 0.04 |
| 5 | DBT (Inc5) | 49 | 27 | 0.81 |
| 6 | DBT (Inc6) | 35 | 78 | 0.55 |
| 7 | DBT (Inc7) | 28 | 59 | 0.53 |
| 8 | DBT (Inc8) | 49 | 40 | 0.23 |

**Table 5.19 Project 3 phase schedule accuracy**

#### 5.3.3.6 The behaviour of schedule accuracy

- Project 1 phase schedule accuracy (Table 5.17) shows that the second highest under-estimating error of the schedule duration was that of another Integration Test phase #14: IT Execution - without Authentication Tool (Inc1). This phase was split from the phase before it, phase #13, during phase execution, in order to simplify testing - i.e. the duration estimate for phase #14 was produced during project execution, not before it. Table 3.1 (sections 3.2.1 - Chapter 3) noted that the Authentication tool was managed by the PLD team. In order to distinguish between defects relating to the Authentication tool which were the responsibility of the Technical environment manager (part of the PLD team), and the defects in the developed code (including interfaces between the developed software and the tool) which were fixed by the Build manager, the Test execution phase was split into two streams of activities running in parallel. Phase #13 tested the integration of the developed components with the Authentication tool, whilst phase #14 tested the integration of the developed components in isolation from the Authentication tool.

- Table 5.17 shows wide variations in the change of duration estimates for the FD phases. FD is particularly vulnerable to scope changes and other external factors.

- Project 2 phase schedule accuracy (Table 5.18) shows that, the phase which its duration estimate was exceeded most was phase #7 (Integration Test - Plan & Preparation).

- The duration estimates of the Test plan and preparation phases in Project 2: i.e. phase #7 (Integration Test - Plan & Preparation) and phase #5 (Assembly Test - Plan & Preparation) are less accurate than their Test execution counterpart phases: i.e. phase #8 (Integration Test - Execution) and phase #6 (Assembly Test - Execution). The project performance reports show that the delay in phase #7 was largely due to external factors to the project team: the Peer supplier delayed providing Test data for several weeks. Performance reports also spoke of external factors causing delay in phase #5, which was the Technical environment manager delay in building Test data into the Assembly Test environment. These indicate who was responsible but not why they were late.

- Project 3 phase schedule accuracy (Table 5.19) shows that the duration estimate of increment #5 - DBT (Inc5) was the most over-estimate, with the increment delivered 22 days earlier than scheduled. Under-estimates are more damaging to schedules than over-estimates so the inaccurate estimate for increment #6 - DBT (Inc6) which was 43 days late - was of more concern.

- The preceding results show that the Test phases contributed most to project delay. This is not surprising since the study by Lehtinen et al. (2014) (see section 4.2 - Chapter 4) found that the most common causes of delay in the studied projects occur in the Design, Build, and Test phases; with the Test phase containing the highest number of causes leading to project delay. They also found that only 2.5% of the overall 648 causes were related to tools. Rainer's investigation of two IBM projects found that the Design, Build, and Test phases for both projects were completed later than scheduled, (Rainer 1999, page: 128, 2010, 2011). Finally, Nelson (2007) conducted 99 retrospectives in 74 organisations involving 502 participants in projects that developed large, medium, and small and size projects. The study found that 54% of the projects suffered from poor estimation/and or scheduling, and that when projects got behind schedule, Testing was one of the first areas that got cut by eliminating Test planning and performing minimal Testing. The extent of delay in the projects or their phases was not studied in Nelson (2007)'s investigation.

## 5.4 Summary

The thesis in this chapter set out to answer the first research question:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

The QUAN approach revealed many empirical insights regarding project schedule behaviour at project and phase units of project work. The findings informed the research that Projects 1, 2, and 3 were delayed by 27%, 25%, and 1% respectively. The research also showed that the phases contributing most to project delay were the Test phases: IT (Inc4) in Project 1 (42 days), Integration Test - Plan & Preparation in Project 2 (88 days), and DBT (Inc6) in Project 3 (43 days).

ABC's project control was focused on delivering on schedule, and its use of a locally-tailored SPI measurement did not provide reliable management information to enable appropriate control. Furthermore, the approach to project execution in overlapping phases and the parallel incremental approach, intended to minimise overall project delay, appear to have contributed to increasing the delay. These findings add to what is already known in software project management field.

Nevertheless, the mechanisms used to control schedule duration although successfully identified problematic points in the projects (the Test phases) which obstruct progress; needed the support of the textual data of the progress reports in order to enable identifying the causes of schedule delay (the quest of RQ1). This begs questions like: 'what' might be influencing schedule duration across the Test phases, and 'how' these influences cause schedule delay? Chapter 6 reports on the approach taken to select specific Test phases for further in-depth examination.

# 6 Case selection

## 6.1 Introduction

This chapter describes the selection of specific cases from the QUAN approach for further in-depth examination using the QUAL approach; the third step in the research process outlined in Chapter 4: Figure 4.1 and section 4.4.2.

This chapter first assesses the extent to which the QUAN results were able to answer the first research question, followed by an outline of the next steps needed to develop answers to the remaining two research questions. The case study approach is then presented, followed by literature review of the two approaches adopted for analysing case study data: Grounded theory and Actor-network theory and how they were applied in this study. The characteristics of the phases that were selected for further in-depth examination are then presented. The actual examination of these cases/phases, however, is carried out in Chapters 7 and 8. Finally, the works in this chapter is summarised.

## 6.2 Selection process

### 6.2.1 The research progress thus far

The QUAN approach attempted to answer the first question of the research:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

Chapter 5 identified a particular phase in each of the three projects examined that was the major contributor to schedule delay in the project - see Table 6.1.

| Project | Original schedule duration (days) | Project schedule delay (days) | Test phase most contributing to project delay | Phase schedule delay (days) |
|---|---|---|---|---|
| 1 | 190 | 52 | IT (Inc4) | 42 |
| 2 | 214 | 54 | Integration Test - Plan & Preparation | 88 |
| 3 | 185 | 2 | DBT (Inc6) | 43 |

**Table 6.1 Schedule delay from QUAN results**

However, this quantitative analysis did not identify why things went wrong in the first place (a major part of the quest of RQ1).

### 6.2.2 Next steps

The research now focused more on understanding why the schedules of the three test phases identified above were delayed; leading to the other two research questions:

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

In order to develop answers to RQ2 and RQ3, a case study design was deemed appropriate (see section 6.3) for examining the three test phases in Table 6.1 in more depth. However, to assess how representative the influences on schedule duration in the three phases were of three projects as a whole, it was decided to sample a few more phases. The selection of the additional phases was guided by the theoretical sampling strategy of the Grounded Theory method (section 7.2.1 - Chapter 7), and led to the analysis of the three phases shown in the last column of Table 6.2.

| Project | Test phase identified from QUAN results | Test phase added from Theoretical sampling of Grounded theory method |
|---|---|---|
| 1 | IT (Inc4) | IT- Execution - with Authentication Tool (Inc1) |

| Project | Test phase identified from QUAN results | Test phase added from Theoretical sampling of Grounded theory method |
|---|---|---|
| | | IT - Execution - without Authentication Tool (Inc1) |
| 2 | Integration Test - Plan & Preparation | Integration Test - Execution |
| 3 | DBT (Inc6) | |

**Table 6.2 Phases selected for QUAL analysis**

Thus the textual data of the progress reports of six Test phases were subjected to a more detailed, qualitative, examination.

## 6.3 Case study approach

A case study is an in depth examination of real-life situations within their context, which may or may not involve researcher intervention in the situation/context. It has been described as an empirical enquiry that investigates a phenomenon when the boundaries of the phenomenon and its context are not clearly evident at the outset of the research, and that it focuses on understanding the dynamics present within single settings (Yin 2009, page: 18; Eisenhardt 1989, page: 534; Benbasat et al. 1987, page: 370). In these situations, case study research may offer greater understanding of the domain investigated (Verner et al. 2009).

It has been argued that case study research can be used equally for QUAN, QUAL, and mixed QUAN/QUAL approaches to enquiry (Yin, 2009; Eisenhardt, 1989). Case study research can be embedded within a larger mixed method approach (Yin 2009, pages 24 and 63; Eisenhardt, 1989, page: 538) which could contain both QUAN and QUAL analyses, and the QUAL strand adopts a case study research approach to analyse and interpret the findings. This could be because the questions for the case study only surfaced after analysis of the QUAN data. The selection of cases might come from those analysed quantitatively and examine them in greater depth (Yin 2009, page: 174; Gomm et al. 2007, page: 107; Hammersley et al. 2007, page: 237). The research carried out here was a case study within one organisation, ABC. The case study approach was embedded within a larger mixed method approach and was used to guide the QUAL part of the analysis.

Case study research can adopt a theoretical proposition at the outset (Yin, 2009; Eisenhardt, 1989), or take an exploratory approach to enquiry (Yin, 2009; Benbasat et al. 1987). Our research was of an exploratory nature and remained open to what emerged from the empirical data. In exploratory case studies, Verner et al. (2009) noted that ‘it is important to use industrially-based cases as the context from which a theory or artifact may emerge’ (page: 313).

### 6.3.1 Case study design

According to Yin (2009), case study research can be designed for a single case, such as a single organisation, or multiple cases, such as a number of organisations. Within either of these designs, further focus can be given to a single unit of analysis (such as one project within an organisation) or multiple units of analysis (e.g. several projects within the same organisation) (Yin 2009, page: 46).

A case can be an individual, an event, an entity, a project phase, a project, or an organisation (Yin, 2009). What defines a case and any further units of analysis is the purpose of the research (Yin, 2009; Benbasat et al. 1987; Remenyi, 2013). Recall from section 6.2.2 that, following the QUAN results the research became interested in understanding why the schedules of the Test phases were delayed. As such, this research defined *phase* as the case to be studied. This means that, answers to the RQ2 and RQ3 were sought in a total of six cases (phases) each with single unit of analysis (the phase). This makes this study a multiple case design with single unit of analysis (Verner et al. 2009).

### 6.3.2 Undertaking case study

It has been suggested that, when sampling for multiple-case research, replication logic could be used to analyse the data and direct the selection of cases (Yin 2009, page: 54); i.e. after analysing the first case in a study, subsequent cases are investigated for either similar or contrasting concepts that have emerged in the first case. This approach is very similar to the theoretical sampling strategy of the Grounded theory (GT) method - see also Eisenhardt (1989), where the investigator follows a particular strategy to select subsequent cases to develop and/or generalise the concepts that have emerged from the first case.

There appear to be no ideal number of cases to be considered as sufficient in multiple-case study research. For example, Yin (2009) suggested 2 or more cases depending on when the investigator believes that the findings support their hypothesis relative to rival explanations. Eisenhardt (1989) suggested 4-10 cases if the objective was to generate theory based in the case study. The number of cases studied in this research was based on the theoretical saturation concept of GT and after analysing 6 cases the emergent categories appeared to have been saturated (see Chapter 7 - section 7.3.2.4).

Although standard approaches have been proposed for the analysis of case study data - see for example Yin (2009) and Eisenhardt (1989), it is possible for researchers to adopt alternative approaches (Yin 2009, page: 136) more suited to the research context. This research used GT techniques to categorise the phenomena influencing schedule duration during the Test phase execution (see section 6.4). Both within-case and across-case analyses were performed on the cases presented in Table 6.2. The result of this analysis was drawn into a GT model, a ‘narrative schema’ showing the relationship among the emergent phenomena. The research then applied Actor-network theory (ANT) concepts (see section 6.5) to the emergent GT categories, and developed an explanatory model to make sense of how interactions among project actors influence schedule delay.

## 6.4 Grounded theory

### 6.4.1 The usefulness of Grounded theory to this case study

Grounded theory practices have gained particular interest in empirical software engineering research in recent years - see for example Coleman & O’Connor (2007), Balaji et al. (2006), Montoni & Rocha (2010), Rose et al. (2007), and Hoda (2010). GT has also been adapted to the circumstances of the information systems (IS) research. Urquhart (2007) noted that ‘IS researchers commonly use grounded theory to generate concepts as opposed to generating theory’ (page: 346). Adapting GT practices has been supported by Grounded theory scholars in the IS field because of the particular characteristics of the IS field involving interaction between technology and people (Urquhart, 2007). Table 6.3 shows example GT studies in software project management research.

| **Area of research** | **Purpose of application (source)** |
|---|---|
| Software project management | Develop conceptual model of the key technological factors affecting success in globally distributed software projects (Qureshi et al. 2004) |
| | Develop categories of best practices used by software project managers to deliver successful projects (Georgieva & Allan, 2008) |
| | Develop theoretical model of competences expected from software project managers in order to deliver successful projects (Rose et al. 2007) |
| Information systems | Develop theoretical frameworks through generating descriptive and explanatory theory of adopting CASE (computer aided software engineering) tools in organisations over time (Orlikowski, 1993) |
| | Develop concepts from the literature of software development methodologies and turning them into questions for interviewing participants on projects developing large software systems (Hansen & Kautz, 2005) |
| Software engineering | Develop theoretical framework that explains when and why software process improvement is undertaken by software companies (Coleman & O’Connor, 2007) |
| | Develop theory of how software project teams operate during the development process in projects that use mixed development methods (Scrum and Waterfall) or selected practices of a particular method (Adolph et al. 2012) |
| Combining GT with other methods | Identify key factors influencing team operation in globally distributed teams through combining GT and Case study research (Casey & Richardson, 2006) |

| **Area of research** | **Purpose of application (source)** |
|---|---|
| | Investigate the impact of variables on the outcome of a newly introduced process in software engineering (PhD thesis), using GT techniques to refine concepts that were borrowed from the literature and form hypotheses, and then test the hypotheses using the traditional methods of empirical studies (Carver & Basili, 2003) |
| | Add rigour and reliability to theory formulation in Action research, integrating some of the GT techniques to Action research (Baskerville & Pries-Heje, 1999). |

**Table 6.3 Grounded theory in software project management research**

Past relevant research by Patanakul (2014) investigated the management of 14 large-scale information systems projects in the public sector (in the UK, US, and Australia) to identify common problems leading to poor project performance and causes of these problems. A case study research approach was adopted using content analysis on past project reports (publicly available government audit reports of these projects). The researcher developed codes (themes) of the problems identified, then grouped these into categories, and in some cases developed diagrams to represent the causal relationships that impacted project performance. A cross-case analysis was then conducted investigating common problems and their causes. The author highlighted the limitation of the approach as the limited generalizability of the findings given the small sampling (14 cases), and the potential subjectivity of the reports since they contain the auditors' interpretation of the events at the time of auditing. Patanakul (2014)'s study is similar to this research: (i) in its investigation of large-scale IT projects in the public sector (ii) its purpose to identify causes of project problems that influence project performance, and doing so across cases (iii) its approach to using case study. Although Patanakul (2014)'s study does not claim to have used Grounded theory techniques, the approach taken in starting with no existing frameworks and coding the data in the project reports indicates some form of category development very similar to the GT approach.

### 6.4.2 The application of Grounded theory in this case study

Earlier in this thesis (section 4.4.3 - Chapter 4), the rationale for using GT techniques was put forward. This section now describes how GT was applied to analyse the six cases in practice; however, the analysis work is reserved for Chapter 7.

The GT approach is a set of procedures used for analysing empirical data in order to develop categories (i.e. put into groups of different types) or theories of process, sequence, and/or interaction within the area being investigated, from the participants' perspective (Glaser & Strauss 1967, page: 114). The GT approach enabled this research to categorise phenomena that emerge during the Test phase execution. The coding process in GT involves making sense of the perceptions that the participants hold (Urquhart, 2013; Guba & Lincoln, 1994) by attaching meaning to them. In the current research this could be perceptions of the events during the Test phase execution of the three projects being examined and which are recorded on the performance reports.

The term GT has been used to refer both to the process of doing GT analysis and to the product of the analysis; i.e. the theory grounded in the data. Bryant (2002) calls the former Grounded theory method (GTM).

The GTM has been developed and clarified over time: there is no unified body of literature as various implementations differ in their approach and the steps of coding and analysis - see Urquhart (2013), Charmaz (2006), Holton (2007), and Corbin & Strauss (2008). The preliminary work of this research applied Charmaz (2006, page: 9)'s implementation of GTM, which advocates coding the 'actions' undertaken by participants in order to develop categories of processes; for example, 'setting schedule target' is an action taken by the Test manager. The textual data in one phase was coded in this research in this way, and a set of useful categories emerged reflecting the 'action' perspective of what was happening during project execution. However, Locke (2001, pages: 41- 42) criticises GT coding procedure for focusing on processes/action over time, and overlooking the structural units of the investigated domain. This research aimed at developing explanatory models that are representative of phenomena present across multiple projects; therefore, the analysis procedure took into consideration the structural elements (constituent parts) present in the data such as 'performance report' or 'product defect' in addition to the process/action took place during project execution. Urquhart (2013)'s implementation of GT, which captures structure as well as process/action was adopted. This is based on Glaser in 'coding the data every way possible' (Glaser, 1978: page 56).

The GT analysis can be distinguished from other types of qualitative analysis in four areas (i) its requirements that preconceived ideas should not influence the development of the emergent categories (ii) theory building is the main purpose of the study (iii) the 'constant comparative method' is used to analyse the research data (iv) the 'theoretical sampling' procedure is used to sample data for analysis (Urquhart et al. 2010, page: 359). The QUAL approach in Chapter 7 employed all above; using GT to create models which are effectively theories based on textual content of the Test phase progress reports. However, this was not a model in the sense that in its current state it could be generalised (see section 4.4.3 - Chapter 4). Hence, further analyses, based on Actor-network theory, were conducted in Chapter 8 to produce a more complete and generalizable model.

Reviewing literature prior to the study and/or acquiring contextual knowledge have been advised against by some (Glaser & Strauss 1967 page: 37; Holton, 2007). Others see this as useful when formulating the research problem (Urquhart, 2013), providing direction during analysis and theoretical sampling (Corbin & Strauss, 2008), and making sense of what might be happening in the text (Charmaz 2006; Coleman & O'Connor, 2007). Yet others see challenges in balancing the emergence of categories from the data and preconceived ideas about the concepts (Kelle, 2007). This research reviewed the literature prior to the study to define the research questions, then allowed categories to emerge from the empirical data, followed by a further literature review to relate the emergent categories with the existing body of knowledge. Walsham (1995) put it succinctly: 'It is possible to access existing knowledge of theory in a particular subject domain without being trapped in the view that it represents final truth in that area' (page: 77).

## 6.5 Actor-network theory

### 6.5.1 The usefulness of Actor-network theory to this case study

Actor-network theory (ANT) has been used in the computer sciences' research in various ways and for different purposes (Walsham, 1997; McLean & Hassard, 2004) - see Table 6.4 for examples. ANT draws attention to the role of technology in organisational management research which may have previously attracted disparate attention by researchers. Orlikowski (2009) usefully categorised management research into studies that (i) recognise the social as having the primary influence on organisations (ii) recognise the technology as having the primary influence (iii) position technology as a product of social interaction (i.e. technology's role can be seen only when used by humans). Orlikowski criticises these approaches for their separation of human from nonhuman components, an 'ontology of separateness', and calls for a 'relational ontology' where neither human nor technology is privileged or treated as separate. On the latter, Orlikowski calls for the use of Actor-network theory or Sociomateriality in management research, since 'contemporary forms of technology and organizing are increasingly understood to be multiple, fluid, temporary, interconnected, and dispersed' (page: 15). Sociomateriality distinguishes between the influences exerted by human (human agency) and the influences exerted by nonhuman (material agency) with respect to intention; while Actor-network theory treats human and nonhuman influences equally (Leonardi, 2011).

| **Area of research** | **Purpose of application (source)** |
|---|---|
| Project management | Understand the process of constructing the project network of building the Skye road bridge in Scotland, where during project execution new actor-networks emerged and influenced schedule duration and caused project failure (Sage et al. 2011) |
| | Interpret the accounts of experienced project managers on how they use project management techniques within uncertain project environments (Blackburn, 2002) |
| | Theory development in the baggage handling information system project in Denver international airport, examining ANT's utility for information systems research (Mahring et al. 2004) |
| | Building software systems in the NHS, examining the stabilisation/failure of project networks (Bloomfield et al. 1997) |
| Information systems | Analyse the development and usage of Geographical Information Systems in India, using ANT to understand the processes of actor-network construction and stabilisation (Walsham & Sahay, 1999) |
| | Explain the causes of deviation from project plan when developing an Enterprise Resource Planning system, using the ANT notion of 'tokens' (Latour, 1987) to interpret the findings (Elbanna, 2008) |
| | Analyse the activities of defining system specifications in the NHS to develop resource management information systems (Bloomfield et al. 1992) |
| | Analyse the success or failure of the development of information technology project in rural communities (Andrade & Urquhart, 2010) |
| | Analyse the implementation of the London Ambulance Service's computer system (McGrath, 2002) |
| Combining ANT with other | Combine ANT with Complexity theory (i.e. the number of different types of components, the number of types of links, and the speed of change of the system) and Reflexivity theory (i.e. the actions taken to stabilise a system can lead to unexpected results |

| Area of research | Purpose of application (source) |
|---|---|
| methods | through emergence of side effects that generate new actions) to understand the complex dynamics inherent in implementing software projects and the way it destabilises the project leading to project failure (Hanseth et al. 2006) |
| | Combine ANT with the Event-based approach (critical events that challenge the execution path of a project) to understand the interaction among various actors during the development of radiology system in hospitals (Cho et al. 2008) |
| | Combine ANT with the Diffusion theory (the spread and acceptance of an idea/practice) to understand the interaction among various actors when evaluating projects in an organisation (Nijland, 2004) for allocating resources/budget. |
| | Combine ANT with the strong structuration theory (empirical application of examining structure and agents in its situated environment) to understand the interaction among various actors in developing large information technology systems in the UK's NHS (Greenhalgh & Stones, 2010) |
| | Combine Actor-network theory with Grounded theory to explain the learning process during decision making under uncertain and complex situations - PhD thesis (Lopes, 2010) |
| | Combine ANT with the Escalation theory (pouring resources into failing projects in an attempt to save the project, which might have the opposite effect) to make sense of the causes of the project failure (Mahring et al. 2004) |

**Table 6.4 Actor-network theory in software project management research**

Past relevant research (PhD thesis) by Lopes (2010) on learning during decision-making under uncertainty and complexity integrated GT with ANT to explain the learning process during decision making. Lopes (2010) used Grounded theory as a research method to categorise data, then mapped them to ANT concepts to explain how decisions are made. Lopes followed Charmaz's (2006) approach when constructing the GT categories and developed three models: conceptual framework, process framework, and behavioural framework. The first emerged from the GT analysis leading to a handful of categories and relationships. As some of the emerged categories were nonhuman, Lopes utilised ANT concepts of inscription, translation, and punctualisation (explained later in Chapter 8 in this thesis) to explain the relationships between the human and nonhuman categories in the process of learning during decision making. This led to the second framework reflecting the decision-making process. The third model then emerged from connecting the conceptual framework to the process framework, and illustrated the behaviour the decision-maker exhibits when making decisions under uncertain and complex situations. Lopes (2010) used ANT concepts to 'enhance and elaborate' on the categories emerged from the GT analysis, and to make 'deeper understanding' of the GT categories to explore 'nonhuman relationships' in the process of making decisions (page: 52). Lopes' (2010) work is similar to this research in applying ANT concepts to GT categories to make sense of a particular situation, and explain the relationship between human and nonhuman elements.

### 6.5.2 The application of Actor-network theory in this case study

The thesis in section 4.4.4 (Chapter 4) presented the rationale for using ANT techniques to develop explanations of schedule delay. This section now presents how ANT was applied; the development of the explanation is addressed in Chapter 8.

ANT is an approach used for describing how things actually are in the area being examined rather than viewing them within existing perspectives (Latour, 2004a) - this should make it compatible with GT which follows this principle too. An ANT study describes the flow of action and interaction between the entities in a project, which can be human or nonhuman (Law, 2012; Callon, 2012; Latour, 2005; Latour, 2004a). As a school of thought, ANT stems from the interdisciplinary field of science and technology studies, which studies the relationships between science, technology, and society in practice (Bijker & Pinch, 2012). ANT can be distinguished from other methods by its (i) approach to analysis with an open-mind as to what influences (e.g. human, nonhuman, or association) may emerge from the empirical data rather than assuming certain influences, usually human ones, will be the key drivers (ii) use of the same vocabulary to describing the interactions of the human and nonhuman in the analysis (Callon, 1986).

The complexity inherent in managing industrial software development projects, as was noted in Chapter 2, requires closer attention to the interdependencies among the constituent parts of the project, and the way they influence one another (Kitchenham, 1987; Abdel-Hamid & Madnick, 1991; Hughes, 2012; Law 2012). The suitability of ANT for studying the complexity of project behaviour comes from the way it makes the researcher pay closer attention to the influences exerted by project participants, both human and nonhuman, during project setup and/or project execution (Law 2012, page: 107; Law, 1991). This is achieved through examining: the activities carried out by project participants/actors (Latour, 1999), the problems present, the actions taken to solve the problems (Callon, 2012), and the success or failure of the project as a result (Walsham, 1997; Akrich et al. 2002). Law (2012) noted that, ANT aims ‘to discover the pattern of forces as these revealed in the collisions that occur between different types of elements’ (page: 108). As will be seen in the course of this study that, ANT enabled (i) a focus on both human and nonhuman elements of a project (ii) interpretation of the interactions through which these elements attempt to control schedule duration and the way this influences schedule delay.

ANT can be used in research as an analytical technique (McLean & Hassard, 2004; Winner, 1993) where ANT concepts are applied to the research findings to *explain* what might be happening in the investigated area. For example, ANT concepts were applied to Grounded Theory categories in this research to make sense of the influences on schedule delay (further examples were provided in section 6.5.1). The outcome of using ANT in this way can be explanatory models (McLean & Hassard, 2004; Dankert, 2011) as was the case in this study modelling the interactions among the actors in the network. This could be a graphical representation of a network comprising sequences of points and lines (Callon 2012 page: 90), that illuminate *why* the project network fails to setup or operate successfully (Walsham, 1997; Law & Callon, 1992; Latour, 1996; Callon & Law, 1989; Latour, 2004a; Latour, 2004b; Dankert, 2009a; Dankert, 2009b).

ANT can also be used as a research method (Walsham, 1997; Dankert, 2011), where the researcher *describes* the associations of the actors in the project network (Law 1991, page: 11; Callon, 1991, page: 154; Walsham 1997, page: 469) - almost reported ANT research results in a description. The outcome of this approach to using ANT can be (McLean & Hassard, 2004; Dankert, 2011) a narrative of *how* project networks are setup or operate (Latour, 1999; Walsham, 1997; Andrade & Urquhart, 2010).

Like other approaches, the way ANT research has been done, have been criticised:

- Social structure: i.e. for lack of distinction between the three aspects of the world: the real, the actual, and the empirical; Mingers & Willcocks (2014, page: 53) elaborate that the real are enduring structures and mechanisms that have particular tendencies and powers generating causal effects in the world. These structures interact with each other and generate actual events that do/and do not occur. Some of these events are observed and experienced, and have the potential to become empirical. The authors contend that ANT analyses only focus on the empirical, which is similar to criticisms raised by other researchers that ANT studies focus on analysing the local (micro) processes of the area being examined, and it fail to consider the wider contextual environment (macro structures) and the way they influence the local (Walsham, 1997; McLean & Hassard, 2004). However, ANT analyses describe what emerges from the studied domain rather than *prior* separation of the context from the content (Latour, 2005, 2004a, 1999, 1991; Monteiro, 2001) and so it is possible for the context to emerge from an ANT analysis if it is present in the data; for example, the works of this research illustrates the influences of the wider context of the studied projects (i.e. the programme) on the delay of projects' schedules.

- Focus and scope of the analysis: i.e. ANT studies focus on selected actor/s in the analysis and lack advice on where to draw the boundary of the network (McLean & Hassard, 2004; Winner, 1993; Star, 1991). However, the research questions and the objective of the investigation should guide the researcher in selecting any actor as a focal actor to start off the analysis, and as to the likely boundary of the network to extend the analysis to (Monteiro, 2001; Law, 1991; Latour, 2004a) as was done in this research where the focus was on the Test manager and the boundary of the analysis included the contextual environment to the projects, such as Technical environment function.

- Symmetry: i.e. treating humans and nonhumans equally (Mingers & Willcocks, 2014; Harbers, 1995; McLean & Hassard, 2004; Walsham, 1997; Monteiro, 2004). On this point, it can be said that ANT does not discount the fact that humans have intentions whilst nonhumans do not; however, the equality view is used during the analysis in order to allow for emergence of what is actually happening in the data rather than enforcing prior assumptions (Latour, 2005; Walsham, 1997; Monteiro, 2001). For example, in this research it was useful to remain open to what might be influencing schedule delay during analysis in order to see that products affect one another, through defect, without human intervention, and which eventually cause schedule delay.

Despite the criticisms above, Winner (1993) praises ANT for ‘its promise to deliver a veritable gold mine of those most highly valued of academic treasures: case studies…its conceptual rigour, its concern for specifics, and its attempt to provide empirical models of technological change that better reveal the actual course of events’ (page: 366-368).

## 6.6 Case characteristics

The following sections provide detailed contextual information on the six cases studied in-depth (Verner et al. 2009), present their similarities and differences, and the dependency among the cases during Test phase execution.

### 6.6.1 Context of the six cases

A Case is essentially a phase of the software project; since the names of some of the examined Test phases were long, a mnemonic acronym was used for easy reference; for example, rather than the phase ‘Integration Test Execution - with Authentication tool (Inc1)’ it was referred to as Case P1-IT-Ex-Au (Inc1). The convention followed was that of ‘<project identifier> - <phase identifier> - <increment identifier>’; for example, P1 indicates Project 1; IT-Ex-Au indicates Integration Test Execution with Authentication tool; and (Inc1) indicates increment 1.

Furthermore, since the six Test phases were subjected to in-depth examination employing a Case study research approach; a phase was treated as a case that was studied in detail. Table 6.5 maps the cases, phases, and the projects they belong to.

| **Project** | **Phase** | **Scheduled duration (days)** | **Duration variance (days)** | **Case** |
|---|---|---|---|---|
| 1 | IT- Execution - with Authentication Tool (Inc1) | 25 | 15 | P1-IT-Ex-Au (Inc1) |
| 1 | IT - Execution - without Authentication Tool (Inc1) | 9 | 15 | P1-IT-Ex-no-Au (Inc1) |
| 1 | IT (Inc4) | 21 | 42 | P1-IT (Inc4) |
| 2 | Integration Test - Plan & Preparation | 28 | 88 | P2-IT-PP |
| 2 | Integration Test - Execution | 49 | 10 | P2-IT-Ex |
| 3 | DBT (Inc6) | 35 | 43 | P3-DBT (Inc6) |

**Table 6.5 Case to phase to project mapping**

## IT- Execution - with Authentication Tool (Inc1)

P1-IT-Ex-Au (Inc1) was the first execution phase of Integration Test in Project 1. It involved testing the integration of the application layers described in Chapter 3 (section 3.3.7). As well as testing the developed software components it included the retesting of code with errors. Recall from section 5.3.3.6 (Chapter 5) that this phase was split into two separate, concurrent, phases due to delays caused by uncertainty about the source of defects requiring rework. One phase tested the functionality without the Authentication tool (a security subcomponent from an external supplier) - see immediately below - and the other with the external component. It may be argued that, it would be logical to test the developed component without the Authentication tool first rather than the opposite.

### IT - Execution - without Authentication Tool (Inc1)

P1-IT-Ex-no-Au (Inc1) was concerned with testing the integration of developed components without Authentication tool in Project 1. The phase had its separate Test execution work stream, separate progress reporting, and separate performance measures tracking its progress. The purpose of this separation was to isolate causes of failing Test scenarios; i.e. to ascertain whether each one was caused by the introduction of the Authentication tool or the developed components were faulty and required fix.

### IT (Inc4)

P1-IT (Inc4) was the last execution phase of Integration Test in Project 1. It was a subsequent Test execution phase to the preceding two cases, and so carried out with all the domain knowledge experiences gained from executing P1-IT-Ex-Au (Inc1) and P1-IT-Ex-no-Au (Inc1). The phase included testing the developed components with the Authentication tool, since the challenges faced initially in setting up the tool and developing appropriate interfaces were now seen as overcome. Nonetheless, section 5.3.3.2 (Chapter 5) showed that P1-IT (Inc4) was the most contributing to the delay of Project 1.

### Integration Test - Plan & Preparation

P2-IT-PP was the planning and preparation phase of Integration Test in Project 2, a subsequent release of the application build in Project 1. Again, this case inherited all the experiences and knowledge learned from the previous project and Test execution phases, and the Test activities were carried out by the same Test manager and Test team.

The Test planning and preparation phase developed a plan that included the functionality to be tested, a schedule, details of components , and a specification of the test data that to be used during subsequent test execution (Sommerville, 2011). Since the plan and preparation were for an Integration Test phase (as opposite to Assembly Test phase), it required coordination with the Peer supplier, in order to test the integration of ABC and Peer supplier components. This involved coordinating data preparation between ABC and Peer supplier to ensure data consistency during Test execution.

### Integration Test - Execution

P2-IT-Ex was the execution phase of the Integration Test in Project 2, and followed P2-IT-PP above. The Test plan and data preparation developed above were implemented to check the functionality in the software components. Again, this case inherited all the experiences and knowledge learned from Project 1 and previous Test execution phases.

### DBT (Inc6)

P3-DBT (Inc6) was a combination of the Design, Build and Test (DBT) phases in Project 3, unlike the preceding cases which only comprised the Test phase. Project 3 was run by a Project manager, phase managers, and project team that were completely different from Projects 1 and 2. In addition to Integration Test, P3-DBT (Inc6) included Assembly Test activities too. Assembly Test excluded testing the developed components with the real Authentication or Component catalogue tools.

## 6.6.2 Similarities and differences among the six cases

Tables 6.6 and 6.7 summarise the similarities and differences among the six cases.

| Characteristic | P1-IT-Ex-Au (Inc1) | P1-IT-Ex-no-Au (Inc1) | P1-IT (Inc4) | P2-IT-PP | P2-IT-Ex | P3-DBT (Inc6) |
|---|---|---|---|---|---|---|
| Phase type | Integration Test Execution | Integration Test Execution | Integration Test Execution | Integration Test Plan & Preparation | Integration Test Execution | Design, Build, Assembly Test, and Integration Test |
| Testing performed with the Authentication tool | Yes | No | Yes | Not applicable | Yes | No in Assembly Test; Yes in Integration Test |
| Delivery duration (in weeks) | 7 | 3 | 11 | 18 | 16 | 8 |
| Located within project | Project 1 | Project 1 | Project 1 | Project 2 | Project 2 | Project 3 |
| Test progress tracking | Event | Event | Event | Event | Event | Increment |

**Table 6.6 Differences among the six cases**

| Characteristic | Comments |
|---|---|
| Phase execution chronology | P1-IT-Ex-Au (Inc1), P1-IT-Ex-no-Au (Inc1), P1-IT (Inc4), P2-IT-PP, and P2-IT-Ex were executed in sequence. P3-DBT (Inc6) was executed in parallel with the above cases |
| Phase type | All cases were Test phases only, with exception of P3-DBT (Inc6) which comprised Test phase in addition to Design and Build |
| Project team | P1-IT-Ex-Au (Inc1), P1-IT-Ex-no-Au (Inc1), P1-IT (Inc4), P2-IT-PP, and P2-IT-Ex were run by the same project team. P3-DBT (Inc6) was run by a different project team |

**Table 6.7 Similarities among the six cases**

### 6.6.3 Position of the six cases within Test execution architecture

Figure 6.1 shows the position of the six cases within the overall Test execution architecture.

The ABC's Test environments hold the Test data and Code to be tested. Uploading the Test data and Code was done through the Technical environment. Projects 1, 2, and 3 arranged for the Test data to be uploaded onto the Test environment. Test execution involved the manual running of a program - XMLSpy - that runs the developed components in the right message routing order through the different application layers described in Table 3.2 (section 3.2 - Chapter 3). When a defect was discovered during the Test execution; the defect was registered by the Test manager, fixed by the designated Build manager, and the fixed code was deployed to the particular Test environment using the Code deploy tool. The Build manager used the Code deploy tool to make a build and deploy to the Test environment; the Code deploy tool used the Technical environment to deploy Code to the Test environment automatically.

On the other side of the common network was the Peer supplier uploading their Test data and Code to their Test environment. The Peer supplier's Technical environment, located at a different physical location, was connected to the ABC's Technical environment to enable the integration test execution messages pass through both systems. A test message is an executable file (combination of Test data and Code) created by the tester, and which represents a functional scenario typically performed by the user, to perform test execution.

If ABC's Code and Test data were according to the expected requirements, the Test message will pass through the Technical environment of ABC to the Technical environment of the Peer supplier in order to reach the Peer supplier's Test environment, provided that the connection works as expected. If the Peer supplier's Code and Test data were according to the expected requirements, the test messages will reach the Mainframe and the resultant message will return back to the Test execution interface using the same path but in reverse order.

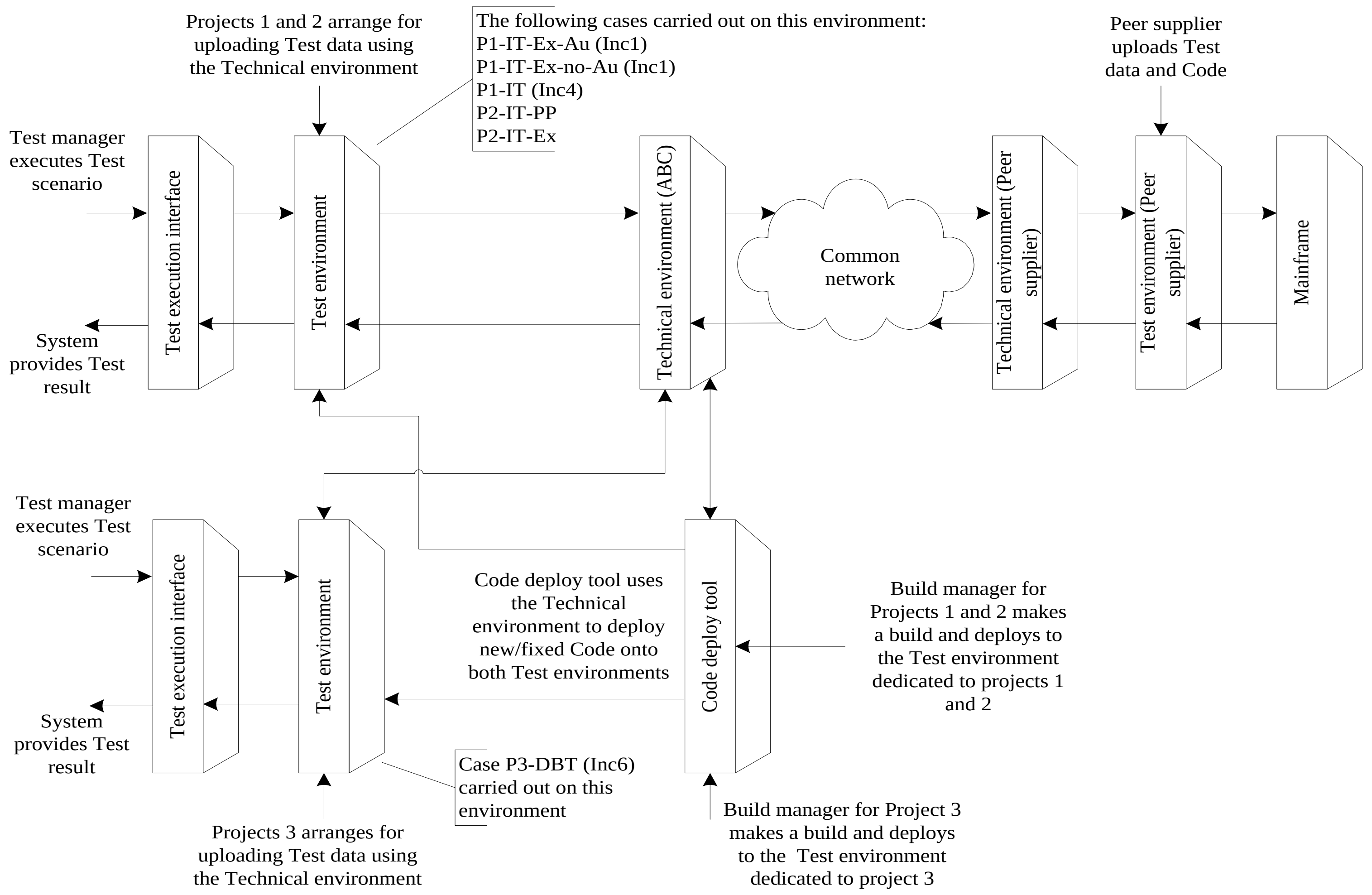

Projects 1 and 2 arrange for uploading Test data using the Technical environment
The following cases carried out on this environment:
P1-IT-Ex-Au (Inc1)
P1-IT-Ex-no-Au (Inc1)
P1-IT (Inc4)
P2-IT-PP
P2-IT-Ex
Peer supplier uploads Test data and Code
Test manager executes Test scenario
Test execution interface
Test environment
Technical environment (ABC)
Common network
Technical environment (Peer supplier)
Test environment (Peer supplier)
Mainframe
System provides Test result
Test manager executes Test scenario
Test execution interface
Test environment
Code deploy tool uses the Technical environment to deploy new/fixed Code onto both Test environments
Code deploy tool
Build manager for Projects 1 and 2 makes a build and deploys to the Test environment dedicated to projects 1 and 2
System provides Test result
Case P3-DBT (Inc6) carried out on this environment
Projects 3 arranges for uploading Test data using the Technical environment
Build manager for Project 3 makes a build and deploys to the Test environment dedicated to project 3

**Figure 6.1 Position of the six cases within Test execution architecture**

## 6.7 Summary

This chapter described the process of selecting three Test phases from the QUAN results for in-depth examination in the QUAL analysis. The chapter also outlined the rationale and approach for sampling for an additional three Test phases. Thus, a total of six Test phases (cases) were selected, for their textual data of the progress reports, to be analysed (see Table 6.2).

The chapter also outlined the Case study approach adopted for the QUAL analysis to enable developing answers to the second and third research questions, using the Grounded theory and Actor-network theory techniques to analyse the textual data of the six cases. The application of GT and ANT will be described in Chapters 7 and 8 respectively. The chapter ended with a detailed description of the characteristics of the six cases.

# 7 Qualitative approach

## 7.1 Introduction

This chapter describes the collection and analysis of the qualitative (QUAL) data and presents its findings. These are the fourth and fifth steps in the research process outlined in Chapter 4: Figure 4.1 and section 4.4.3. The chapter attempts to answer the second research question

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

In order to develop answers to RQ2, the thesis needed to scrutinise the textual data in the Test phases' progress reports, categorise all the different types of phenomena that are present during the Test phases' execution, and make sense of their influence on schedule duration. To do so in a systematic way, however, an established approach was needed; and that was the Grounded theory (GT) method.

The chapter firstly describes the approach adopted in the QUAL analysis. The approach is illustrated through an analysis of case P1-IT (Inc4) data and the results of the process, i.e. categories of phenomena present during phase execution, are shown. The emergent phenomena across the six cases are then produced. A narrative schema depicts the content of the text in these cases is developed and analysed. Finally, the works in this chapter is summarised.

## 7.2 Approach

### 7.2.1 Sampling strategy

The QUAL approach adopted the theoretical sampling strategy of GT (Glaser & Strauss 1967, page: 58). Section 6.2.2 (Chapter 6) argued for sampling a few more cases in addition to the three Test phases most delayed in the three projects in order to strengthen the development of the emergent categories (i.e. types of phenomena) and achieve representativeness across the three projects; and that the Theoretical sampling technique of GT method was employed to select the additional cases.

Theoretical sampling can be used to develop (i.e. enlarge/build) categories and/or to generalise categories (Glaser & Strauss 1967, page: 58).

*Developing categories* involves sampling *similar cases* to the one already analysed. If the data is similar, it results in the accumulation of similar data that make up a given category (i.e. enlargement of the category). If the data is dissimilar, it results in differentiation among the emergent categories (i.e. development of new categories to the ones already emerged). For example, if the category ‘people’ represents ‘test manager’ in case 1 (a Test phase). Analysing case 2 (another Test phase) results in the emergence of ‘test manager’ again, this develops the category people. If analysing case 2 also results in the emergence of ‘performance report’ this results in a new category ‘product’.

*Generalising categories* involves sampling *different* cases to the one already analysed. If the data is similar, it results in the accumulation of varied data that make up a given category (i.e. generalisation of the category). If the data is dissimilar, it increases the applicability of the theory being generated (i.e. generalisation across the cases). For example, if the category ‘people’ represents ‘test manager’ in case 1 (a Test phase). Analysing case 3 (a Build phase) results in the emergence of ‘build manager’, this generalises the category people across the two cases. If analysing case 3 also results in the emergence of ‘code’, this increases the applicability of the emergent theory across the cases.

Table 7.1 shows the theoretical sampling approach in this case study. Section 7.3.2.4 illustrates how this was done in practice.

| **Order of selection** | **Project** | **Case** | **Sampling approach** | **Phase type** | **Rationale** |
|---|---|---|---|---|---|
| 1 | 1 | P1-IT (Inc4) | Purposeful | Integration Test Execution | Contributed most to project delay |
| 2 | 1 | P1-IT-Ex-Au (Inc1) | Theoretical sampling of GT | Integration Test Execution | To develop the emergent categories thus far |
| 3 | 1 | P1-IT-Ex-no-Au (Inc1) | Theoretical sampling of GT | Integration Test Execution | To develop the emergent categories thus far |
| 4 | 2 | P2-IT-PP | Purposeful | Integration Test Plan and preparation | Contributed most to project delay; but which also increases the generalisation of the emergent categories thus far |
| 5 | 2 | P2-IT-Ex | Theoretical sampling of GT | Integration Test Execution | To develop the emergent categories thus far; and to increase the generalisation of the emergent categories thus far |
| 6 | 3 | P3-DBT (Inc6) | Purposeful | Design, Build, and Test | Contributed most to project delay; but which also increases the generalisation of the emergent categories thus far, and increases the applicability of the emergent theory |

**Table 7.1 Theoretical sampling of the case study**

### 7.2.2 The information collected

Textual information from weekly project progress reports was extracted (section 3.4.1 - Chapter 3). The unit of analysis for this qualitative study was project phase. Section 6.6 (Chapter 6) described the characteristics of the six phases selected.

The QUAL data analysis used software NVivo to manage the data during analysis. Identifiable information was made anonymous prior to entering into NVivo, and missing performance reports for particular weeks were marked as [No performance report]. Appendix A (Figure A.2) provides example screenshot of the textual data in NVivo.

### 7.2.3 Techniques used: Grounded theory

The thesis introduced earlier (section 4.4.3 - Chapter 4; section 6.4.2 - Chapter 6) the use of GT to analyse the textual data of the six cases; i.e. to categorise all the different types of factors that appeared during the Test phases' execution and their influence on schedule duration. The next section of this chapter now illustrates the QUAL approach through application of the analysis procedure to case P1-IT (Inc4), followed by the findings across the six cases.

## 7.3 Case P1-IT (Inc4) findings

### 7.3.1 Case P1-IT (Inc4) textual data

Table 7.2 shows the textual data of the weekly progress reports for P1-IT (Inc4) phase. The progress report was written by the Test manager for the Project manager and was discussed in the weekly status meetings attended by all other phase managers in addition to the Project manager. The narratives in Table 7.2 (first column) represent the individual weeks when progress was reported.

| Narrative # | Textual information |
|---|---|
| N_01 | Integration Test complete for Inc3. Inc4 execution not started due to X+ environment setup delays – expected start date Wk1. Inc4 X+ environment not yet available for execution due to XXX connectivity issues and L code has not been deployed. |
| N_02 | Integration Test complete for Inc3. Inc4 execution not started due to P1 XXN environment setup and Unit Test delays – expected start date Wk2. |
| N_03 | Integration Test complete for Inc3. Inc4 execution started late (Wk2). So far 12/17 components ran clean, 13 new Defects have been raised. Inc4 execution started late due to P1 XXN environment setup and Unit Test delays. |
| N_04 | Integration Test complete for Inc3. Inc4 execution started 1.5 weeks late (Wk2). Inc4 execution extended to end of Wk4 package window (Wk5). Environmental issues have significantly hindered progress. |
| N_05 | Integration Test complete for Inc3. Inc4 execution started 1.5 weeks late (Wk2). Inc4 execution extended to end of Wk4 package window (Wk5). Environmental issues have significantly hindered progress. |
| N_06 | Inc4 execution extended to Wk6. Environmental issues have significantly hindered progress and continue to prevent closure of Inc4. |
| N_07 | One Third-party defect remains with component X. Third-party reports this is a live issue and a fix is not going to be provided as this has never been seen in live. A Change Request is being raised by Business design to change the L behaviour in this area which will be scheduled into a future release. Awaiting confirmation from Business Design that Inc4 can be closed with this issue open as live volumes of temporary component X are very low, and from Consumer to confirm they are happy to close Inc4 with this open. |
| N_08 | Awaiting sign off of P1_R1 Integration Test Completion Report. |
| N_09 | Awaiting sign off of P1_R1 Integration Test Completion Report. |
| N_10 | Awaiting sign off of P1_R1 Integration Test Completion Report. |
| N_11 | P1_R1 Integration Test Completion Report signed off. Completed. |

**Table 7.2 Case P1-IT (Inc4) Textual information**

### 7.3.2 Case P1-IT (Inc4) analysis procedure

The overall procedure for coding the textual data is shown in Figure 7.1, which started with coding the textual data for case P1-IT (Inc4).

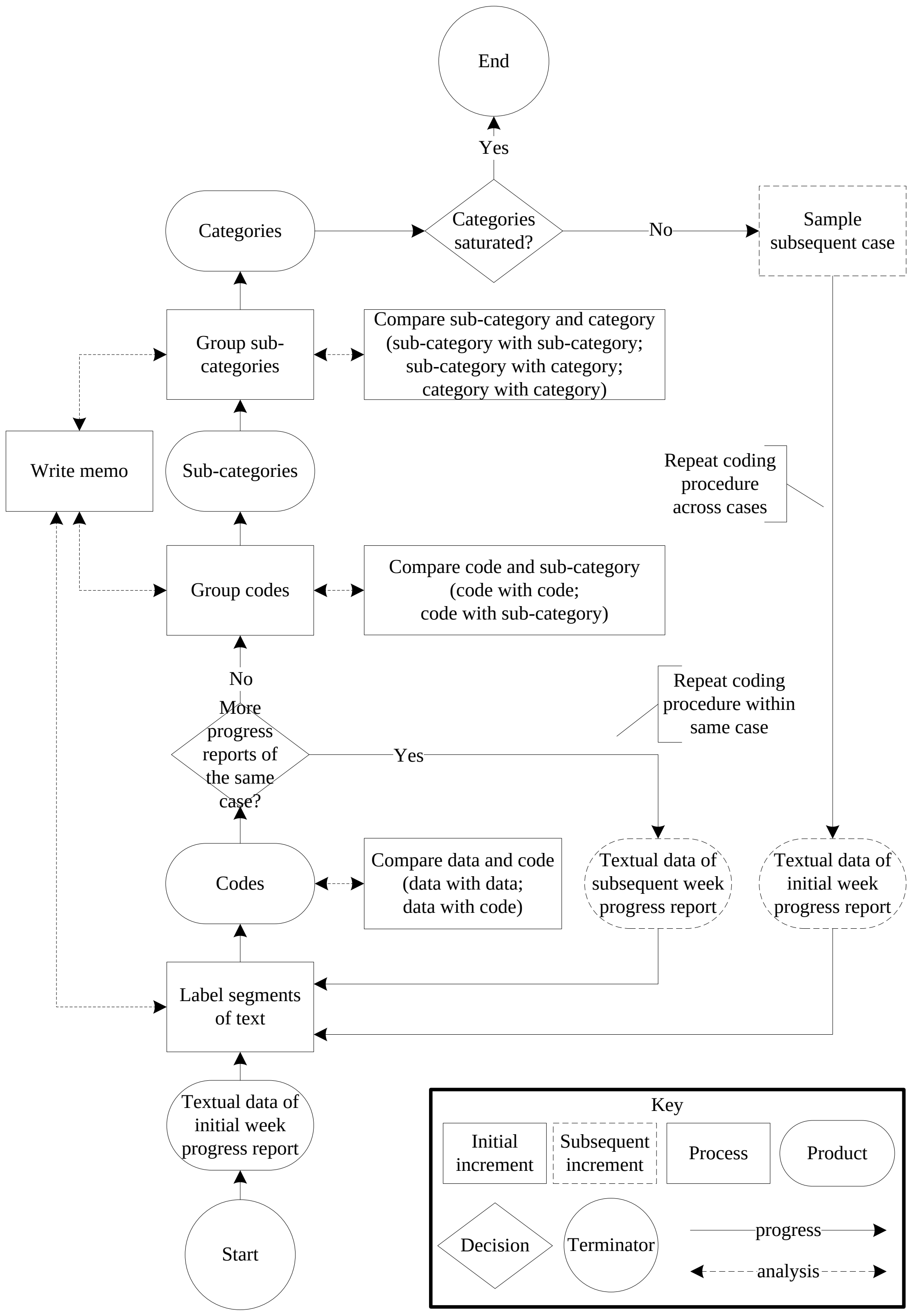

End
Yes
Categories
Categories saturated?
No
Sample subsequent case
Group sub-categories
Compare sub-category and category (sub-category with sub-category; sub-category with category; category with category)
Write memo
Sub-categories
Repeat coding procedure across cases
Group codes
Compare code and sub-category (code with code; code with sub-category)
No
More progress reports of the same case?
Yes
Repeat coding procedure within same case
Codes
Compare data and code (data with data; data with code)
Textual data of subsequent week progress report
Textual data of initial week progress report
Label segments of text
Textual data of initial week progress report
Start
Key
Initial increment
Subsequent increment
Process
Product
Decision
Terminator
progress
analysis

**Figure 7.1 Coding procedure**

As Figure 7.1 shows that, GT analysis in this research involved: developing codes, sub-categories and categories; constant comparison; writing memos; and theoretical sampling (all explained below). These practices were applied in an iterative and incremental approach where the work moved through these stages a number of times in order to arrive to the emergent categories (Locke, 2001; Orlikowski, 1993).

### 7.3.2.1 Codes

Coding the textual data of P1-IT (Inc4) progress reports involved assigning meaning to actions, structural units, and the relationships between them; in order to account for what was perceived to be happening within segments of text (Urquhart, 2013; Locke, 2001; Corbin & Strauss, 2008; Charmaz, 2006; Holton, 2007). For example, the textual data in the first week report (see Table 7.2, N_01) was coded as shown below:

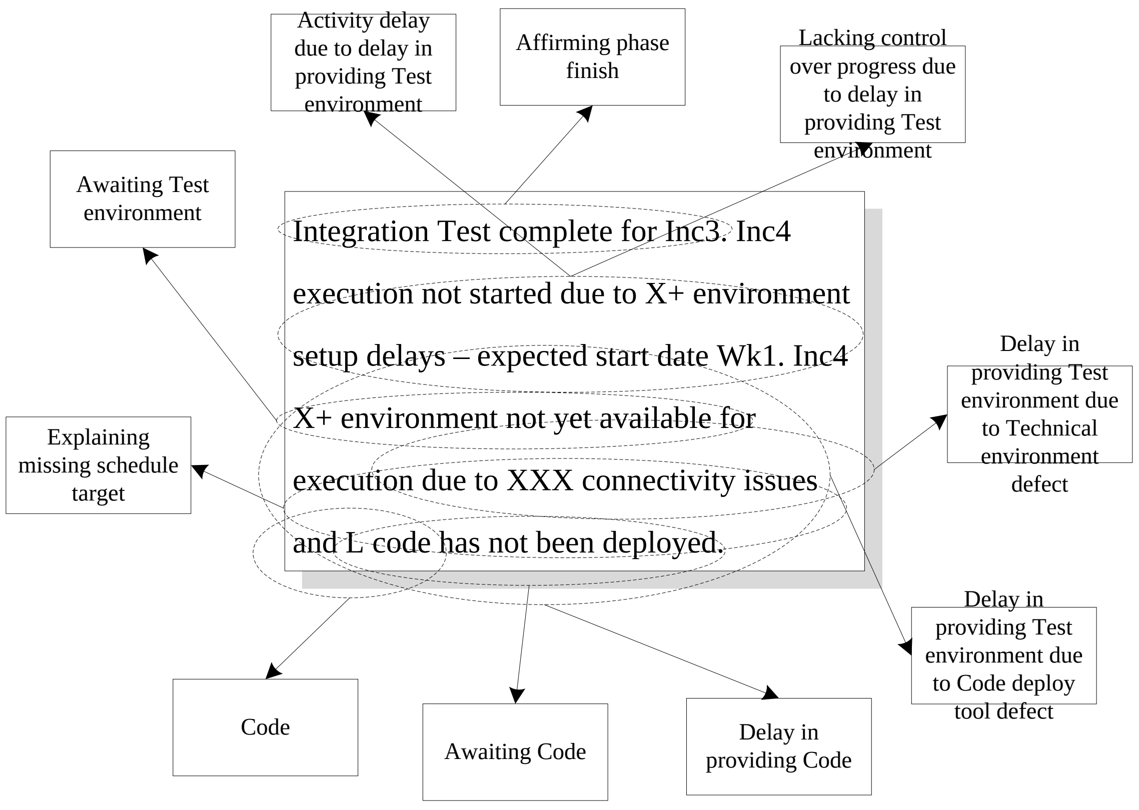

Activity delay due to delay in providing Test environment
Affirming phase finish
Lacking control over progress due to delay in providing Test environment
Awaiting Test environment
Integration Test complete for Inc3. Inc4 execution not started due to X+ environment setup delays – expected start date Wk1. Inc4 X+ environment not yet available for execution due to XXX connectivity issues and L code has not been deployed.
Delay in providing Test environment due to Technical environment defect
Explaining missing schedule target
Delay in providing Test environment due to Code deploy tool defect
Code
Awaiting Code
Delay in providing Code

**Figure 7.2 Coding P1-IT (Inc4) textual data for N_01**

Figure 7.2 shows that during coding (also called Open coding) a statement can be coded more than once - coding the data every way possible (Holton, 2007). For example, the statement ‘Inc4 execution not started due to X+ environment setup delays – expected start date Wk1’ was coded as ‘Activity delay due to delay in providing Test environment’ as well as ‘Lacking control over progress due to delay in providing Test environment' because it represents both situations. The difference between these two codes may appear to be rather subtle; however, they differ in meaning; one conveys a delaying obstruction, the other the condition of being delayed. That is, they relate to the same event (delay in providing Test environment), but to different event outcomes (activity delay, and lacking control over progress).

As Figure 7.1 showed that the subsequent week's textual data was then coded, see Figure 7.3:

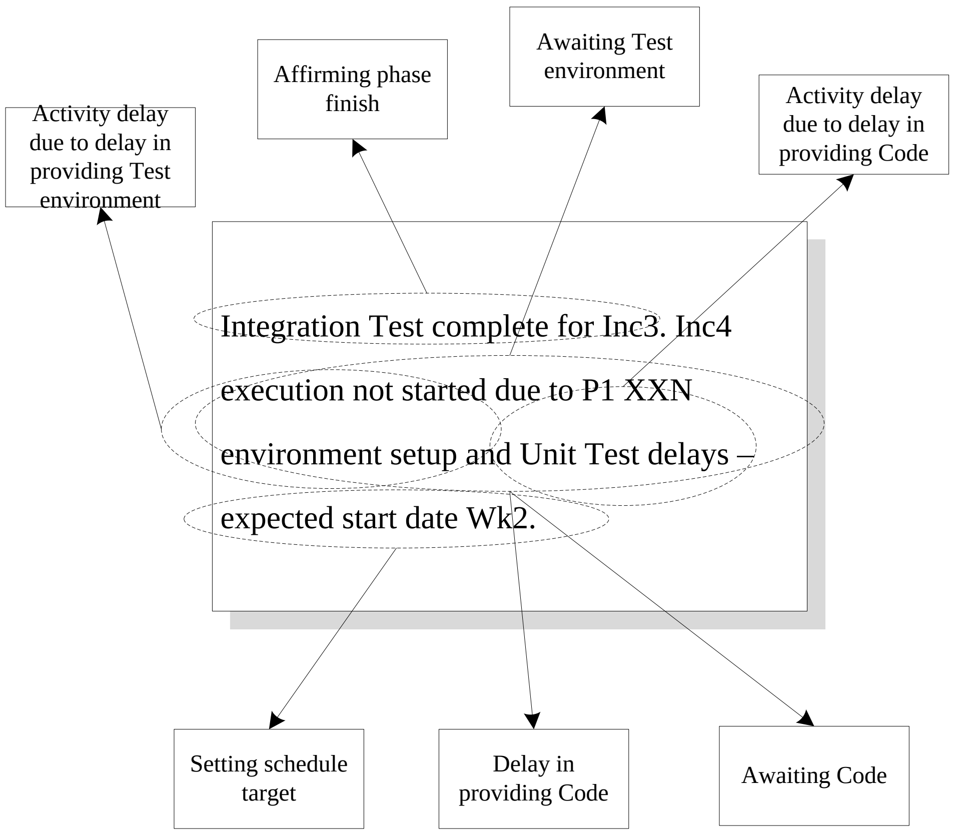

Affirming phase finish
Awaiting Test environment
Activity delay due to delay in providing Test environment
Activity delay due to delay in providing Code
Integration Test complete for Inc3. Inc4 execution not started due to P1 XXN environment setup and Unit Test delays – expected start date Wk2.
Setting schedule target
Delay in providing Code
Awaiting Code

**Figure 7.3 Coding P1-IT (Inc4) textual data for N_02**

Figure 7.3 shows that coding the subsequent progress report for the same case is done with the existing codes in mind (see process 'Compare data and code'). The text segments of the subsequent week are compared with those of the previous week and with the codes that were analysed in that week. A check is made to see if the new week's text can be labelled under existing codes or whether new codes are needed. This is called 'constant comparative method' in GT.

In this research, constant comparison was used to compare data segments belonging to a code, codes within a sub-category, and sub-categories within a category to check that similar items are grouped together (within code comparison). Constant comparison was also used to compare new instances of data with existing codes, sub-categories, and categories (Kelle, 2007; Locke, 2001; Urquhart, 2013; Charmaz, 2006). For example, the following text from N_02 was compared with the text in N_01:

> N_02: Inc4 execution not started due to P1 XXN environment setup and Unit Test delays
>
> N_01: Inc4 execution not started due to X+ environment setup delays

Both pieces of text indicate delays in starting the phase. However, N_02 states the delay was caused by the lack of code (a unit test is done as the last coding activity before integration testing) as well as a test environment, whilst N_01 only mentions the test environment. N_02 text was coded under an existing code for N_01, ‘Activity delay due to delay in providing Test environment’; and also under a new code ‘Activity delay due to delay in providing code’ - see Figure 7.3. The code ‘Activity delay due to delay in providing Test environment’ is now generalised across both narratives.

The above is an example of ‘memo writing’ in GT analysis, where the researcher writes their thoughts about and interpretations of the objects, events, and actions taking place during constant comparison of data items; and the identification of relationships between sub-categories and how the overall framework might fit together (Locke, 2001; Corbin & Strauss, 2008). Figure 7.1 shows that analytic memos being written to make sense of the text, and helping in refining the meaning of the emerged categories - more example memos are provided in Appendix A4.

Applying the above procedures on the remaining textual reports of P1-IT (Inc4) resulted in the Codes listed in Table 7.3. For description of these codes see Appendix A1. Once again, similarity may be noted between some of the codes in Table 7.3, yet each code conveys a different meaning. For example, the codes #5, 10, and 21 may appear similar, but each denotes a different step in the event sequence taking place with regards to fixing code defect by Peer supplier. Code #5 captures the *waiting* condition of the Test manager on the Peer supplier to fix the Code. Code #10 encapsulates the *delay* condition created by the Peer supplier to fix the Code. Code #21 describes the status of progress of the Test phase, which is now delayed to meet its activity targets because of the delay by Peer supplier; i.e. the same event can have different impacts on different entities.

| # | **Code** |
|---|---|
| 1 | Awaiting Client decision on Scope change |
| 2 | Awaiting Consumer decision on Scope change |
| 3 | Awaiting decision on Test completion report |
| 4 | Awaiting Code defect fix |
| 5 | Awaiting Code defect fix from Peer supplier |
| 6 | Awaiting Technical environment defect fix from Technical environment manager |
| 7 | Awaiting Code |
| 8 | Awaiting Test environment |
| 9 | Delay in deciding on Test completion report |
| 10 | Delay in fixing Peer supplier Code defect |
| 11 | Delay in providing Code |
| 12 | Delay in providing Test environment due to Code deploy tool defect |
| 13 | Delay in providing Test environment due to Technical environment defect |
| 14 | Client |
| 15 | Consumer |
| 16 | Peer Supplier |
| 17 | Code |
| 18 | Code defect |
| 19 | Technical environment defect |
| 20 | Test completion report |
| 21 | Activity delay due to delay in fixing Peer supplier Code defect |
| 22 | Activity delay due to delay in providing Code |
| 23 | Activity delay due to delay in providing Test environment |
| 24 | Activity delay due to Technical environment defect |
| 25 | Duration extension due to Technical environment defect |
| 26 | Lacking control over progress due to delay in approving Test completion report |
| 27 | Lacking control over progress due to delay in providing Test environment |
| 28 | Phase delay due to delay in deciding on Test completion report |

| # | **Code** |
|---|---|
| 29 | Phase delay due to Technical environment defect |
| 30 | Affirming phase finish |
| 31 | Explaining missing schedule target |
| 32 | Quantifying progress |
| 33 | Setting schedule target |
| 34 | Started late |
| 35 | Started late due to Delay in providing Code |
| 36 | Started late due to delay in providing Test environment |
| 37 | Started late due to Technical environment defect |
| 38 | Scope cut due to Client Change request |
| 39 | Scope move due to Peer supplier Code defect |

**Table 7.3 P1-IT (Inc4) Codes**

#### 7.3.2.2 Sub-categories

The next step was to group Codes that represent similar themes in Table 7.3 into higher level of abstraction to form sub-categories. The act of grouping lower level codes that represent similar themes into to higher level codes is called ‘developing categories’ in GT, which is achieved through the constant comparative method.

A sub-category in GT can be one of two types: selective or theoretical (Urquhart, 2013). Selective sub-categories represent the elements (constituent parts) of the project. Theoretical sub-categories express a relationship between the Selective sub-categories (Holton, 2007; Urquhart, 2013). In the example given in the previous section about codes #5, 10, and 21 (see paragraph preceding Table 7.3); the Test manager, Peer supplier, and Activity delay can be seen as selective sub-categories, the Test manager is associated with the Peer supplier through waiting on defect fix (theoretical sub-category) of the code, and the Peer supplier is associated with Activity delay through delaying fix of the defect (theoretical sub-category).

‘Compare code and sub-category’ in Figure 7.1 is exemplified by the process of comparing the various codes in Table 7.3 against one another for similarities and differences to ascertain whether they should be different codes or whether they are better combined. Similarly, when sub-categories are formed, the codes are compared to determine whether they should be grouped under a particular sub-category. Codes under a sub-category share similar characteristics; at the same time exhibit some variations (Urquhart 2013, page: 9). For example, although the first eight codes in Table 7.3 represent ‘waiting’, they represent different aspects of waiting. Whilst codes #1, 2, and 3 represent waiting on decision, codes #4, 5, and 6 represent waiting on defect fix, and codes #7 and 8 represent waiting on a product.

Applying the same approach on the remaining codes in Table 7.3 resulted in the sub-categories shown in Table 7.4. For description of these sub-categories see Appendix A1. For a list of the selective sub-categories that are linked by the theoretical sub-categories see Appendix A2.

| # | Code | Sub-category | Sub-category Type |
|---|---|---|---|
| 1 | Awaiting Client decision on Scope change | Awaiting decision | Theoretical code |
| 2 | Awaiting Consumer decision on Scope change | | |
| 3 | Awaiting decision on Test completion report | | |
| 4 | Awaiting Code defect fix | Awaiting defect fix | Theoretical code |
| 5 | Awaiting Code defect fix from Peer supplier | | |
| 6 | Awaiting Technical environment defect fix from Technical environment manager | | |
| 7 | Awaiting Code | Awaiting product | Theoretical code |
| 8 | Awaiting Test environment | | |
| 9 | Delay in deciding on Test completion report | Delay in deciding | Theoretical code |
| 10 | Delay in fixing Peer supplier Code defect | Delay in fixing defect | Theoretical code |
| 11 | Delay in providing Code | Delay in providing product | Theoretical code |
| 12 | Delay in providing Test environment due to Code deploy tool defect | | |
| 13 | Delay in providing Test environment due to Technical environment defect | | |
| 14 | Client | Client | Selective code |
| 15 | Consumer | Consumer | Selective code |
| 16 | Peer Supplier | Peer Supplier | Selective code |
| 17 | Code | Code | Theoretical code |
| 18 | Code defect | Defect | Theoretical code |

| # | Code | Sub-category | Sub-category Type |
|---|---|---|---|
| 19 | Technical environment defect | | |
| 20 | Test completion report | Test completion report | Theoretical code |
| 21 | Activity delay due to delay in fixing Peer supplier Code defect | Activity delay | Selective code |
| 22 | Activity delay due to delay in providing Code | | |
| 23 | Activity delay due to delay in providing Test environment | | |
| 24 | Activity delay due to Technical environment defect | | |
| 25 | Duration extension due to Technical environment defect | Duration extension | Selective code |
| 26 | Lacking control over progress due to delay in approving Test completion report | Lacking control over progress | Theoretical code |
| 27 | Lacking control over progress due to delay in providing Test environment | | |
| 28 | Phase delay due to delay in deciding on Test completion report | Phase delay | Selective code |
| 29 | Phase delay due to Technical environment defect | | |
| 30 | Affirming phase finish | Reporting progress status | Theoretical code |
| 31 | Explaining missing schedule target | | |
| 32 | Quantifying progress | | |
| 33 | Setting schedule target | | |
| 34 | Started late | Start variance | Selective code |
| 35 | Started late due to Delay in providing Code | | |
| 36 | Started late due to delay in providing Test environment | | |

| # | Code | Sub-category | Sub-category Type |
|---|---|---|---|
| 37 | Started late due to Technical environment defect | | |
| 38 | Scope cut due to Client Change request | Scope change | Theoretical code |
| 39 | Scope move due to Peer supplier Code defect | | |

**Table 7.4 P1-IT (Inc4) Sub-categories**

#### 7.3.2.3 Categories

Developing categories consists of grouping the sub-categories in Table 7.4 that represent similar themes into categories. The way codes are grouped into sub-categories and sub-categories into higher level categories can be driven by how best to answer the research questions in the programme of study (Urquhart, 2013; Locke, 2001).

The process ‘Compare sub-category and category’ in Figure 7.1 shows that the grouping of sub-categories into categories involves comparing sub-categories with each other to ascertain whether they are unique or are so similar that they should be merged. Sub-categories are compared with their categories to determine whether the category represents the sub-category; and categories are compared with one another to refine their meaning and what they represent. For example, the first three sub-categories in Table 7.4 relate to the Test manager waiting for products or services in order to progress the Test execution phase and the following three sub-categories represent delay in providing the products or services by the provider. These six sub-categories seemed to represent dependency when carrying out project activities, hence were grouped under a category named ‘Dependency’. It is possible that waiting may not lead to schedule delay, but waiting and delay are two sides of the same phenomenon; i.e. dependency.

Thus, the sub-categories in Table 7.4 were categorised as shown in Table 7.5. For description of these codes see Appendix A1.

| # | **Sub-category** | **Category** |
|---|---|---|
| 1 | Awaiting decision | Dependency |
| 2 | Awaiting defect fix | |
| 3 | Awaiting product | |
| 4 | Delay in deciding | |
| 5 | Delay in fixing defect | |
| 6 | Delay in providing product | |
| 7 | Client | People |
| 8 | Consumer | |
| 9 | Peer Supplier | |
| 10 | Code | Product |
| 11 | Defect | |

| # | Sub-category | Category |
|---|---|---|
| 12 | Test completion report | |
| 13 | Activity delay | Schedule |
| 14 | Duration extension | |
| 15 | Lacking control over progress | |
| 16 | Phase delay | |
| 17 | Reporting progress status | |
| 18 | Start variance | |
| 19 | Scope change | Scope |

**Table 7.5 P1-IT (Inc4) Categories**

### 7.3.2.4 Theoretical sampling and saturation

The emergent categories in Table 7.5 represent categories of phenomena present in the P1-IT (Inc4)'s textual data. It was reasonable to examine whether the categories were supported by sufficient amount of data (i.e. are well-developed) and how applicable the categories were to other cases (i.e. generalised) - see section 7.2.1. These can be achieved by checking that coding the data of additional cases uses existing categories (i.e. enlarge existing categories) and does not need new categories (i.e. generalise existing categories); this is called Theoretical saturation in GT (Corbin & Strauss 2008, pages: 143 and 148; Urquhart 2013, page: 9).

Reaching the saturation point in GT is considered at category level rather than sub-category, because a grounded theory (i.e. the product of a grounded theory study) is typically based on categories. Urquhart (2013) noted that 'It is usually quite obvious, in a grounded theory study, when to stop data collection. It is when the researcher finds no new concepts are emerging from the data - all that is happening is there are more instances of existing categories. In this way, *theoretical saturation* is reached - the particular category is seen to be 'saturated', that is, full' (page: 9) (emphasis in original). It may be noted that 'category' and 'concept' are used interchangeably in the GT literature - see also (Corbin & Strauss 2008, page: 159). Table 7.6 shows a count of references to the data items under each of the emergent categories.

| Category | P1-IT (Inc4) |
|---|---|
| Dependency | 26 |

| | |
|---|---|
| People | 4 |
| Product | 10 |
| Schedule | 38 |
| Scope | 2 |

**Table 7.6 Category saturation in P1-IT (Inc4)**

In order to determine whether the phenomena in Table 7.6 are only applicable to case P1-IT (Inc4), or whether other cases exhibit similar or different phenomena; another Integration Test phase was analysed in Project 1 to develop the above categories further. The theoretical sampling strategy in section 7.2.1 is designed for the development of categories and their generalisation across more than one case. The codes, sub-categories and categories from the first case were used as the starting point for the constant comparison in the later cases, rather than each case starting with a blank sheet. Further cases were analysed as shown in Figure 7.1 and the emergent codes, sub-categories, and categories are all presented in Appendix A1.

Selecting further cases to P1-IT (Inc4) for analysis followed the order presented in Table 7.1 (section 7.2.1). Since P1-IT (Inc4) was an Integration Test execution phase in Project 1, another Integration Test execution phase (P1-IT-Ex-Au (Inc1)) in Project 1 was analysed to further develop the categories in Table 7.6. This analysis resulted in the emergence of the same categories as P1-IT (Inc4) with exception of Scope, which indicates lack of saturation of this particular category; because the new case did not use the category (i.e. remained small), and the category remained representing the data of only one case thus far (i.e. not generalised). Thus another Integration Test execution phase (P1-IT-Ex-no-Au (Inc1)) in Project 1 was analysed, again without resulting in the saturation of category Scope.

In order to generalise the developed categories from P1-IT (Inc4), P1-IT-Ex-Au (Inc1), and P1-IT-Ex-no-Au (Inc1) in Project 1, across the three projects, the analysis then moved to the Test phase most contributing to delay in Project 2: P2-IT-PP (Integration Test - Plan & Preparation). This resulted in emergence of the same categories as the preceding ones, in addition the category ‘Scope’ re-emerged. Then in order to develop the categories further, P2-IT-Ex (Integration Test - Execution phase) was selected in Project 2 which confirmed the categories emerged thus far. In order to generalise the emerged categories further and ascertain saturation, the research analysed P3-DBT

(Inc6) the most delayed phase in Project 3, which supported the emergence of all the categories thus far whereby achieving theoretical saturation.

Table 7.7 shows the count of references to the textual information where instances of the particular category emerged in all six cases. Table 7.7 shows the cases on the horizontal axis, the categories on the vertical axis, and the count of the references to the categories for each case in the intersection of the horizontal to the vertical entry. A count of zero indicates that no category appeared in the particular case - Table 7.7 shows that all the categories became saturated as the analysis progressed to the last three cases.

| **Name** | **P1-IT (Inc4)** | **P1-IT-Ex-Au (Inc1)** | **P1-IT-Ex-no-Au (Inc1)** | **P2-IT-PP** | **P2-IT-Ex** | **P3-DBT (Inc6)** | **Total** |
|---|---|---|---|---|---|---|---|
| Dependency | 26 | 16 | 10 | 31 | 19 | 70 | 172 |
| People | 4 | 6 | 2 | 11 | 5 | 20 | 48 |
| Product | 10 | 13 | 11 | 10 | 14 | 57 | 115 |
| Schedule | 38 | 31 | 11 | 36 | 46 | 83 | 245 |
| Scope | 2 | 0 | 0 | 4 | 5 | 13 | 24 |

**Table 7.7 Theoretical saturation across the six cases**

Examining Table 7.7 closely, the following observations can be made:

- The category ‘Scope’ emerged in P1-IT (Inc4), which is the phase most contributing to the delay of Project 1. However, Scope did not appear in the other two integration test phases in Project 1: P1-IT-Ex-Au (Inc1) and P1-IT-Ex-no-Au (Inc1). This does not mean, however, that the delay in P1-IT (Inc4) was only due to scope related phenomena.

- The category ‘Schedule’ emerged as the most prevalent across the six cases (referenced 245 times - see the ‘Total’ column). The case with highest schedule related phenomena is P3-DBT (Inc6), followed by P2-IT-Ex. Considering that P3-DBT (Inc6) was a combined increment (Design, Build, and Test); P2-IT-Ex would be the highest among the Test-only phases with schedule related phenomena.

- The category ‘Dependency’ emerged as the second most prevalent phenomena, with P3-DBT (Inc6) exhibiting emergence of highest number of this phenomenon among the cases. Among the Test-only phases, however, P2-IT-PP exhibits highest of dependency related phenomena.

- The fact that, P3-DBT (Inc6) exhibits highest number of phenomena in all categories reflects the composition of the phase in reporting on a combined Design, Build, and Test phases rather than the Test-only phase as with the other five cases. Given the difference in the nature of this last case, lack of emergence of new categories supports the achievement of theoretical saturation.

The emergent sub-categories that are present in all six cases are:

- Awaiting defect fix, Awaiting product, Delay in providing product under the category Dependency.

- Defect under the category Product.

- Activity delay, Lacking control over progress, Reporting progress status under the category Schedule.

## 7.4 Narrative schema

The emergent sub-categories (consisting of both selective and theoretical codes) from the six cases (see Appendix A1) have been integrated into diagram to enable the visualisation of the relationships among the sub-categories and the making sense of what might be happening in the text (Urquhart 2013, page: 114; Charmaz, 2006, page: 117; Corbin & Strauss 2008, page: 124). Such a narrative schema, however, is limited to what was reported in the performance reports rather than the underlying physical system, and therefore it cannot be assumed to encapsulate all the contextual information surrounding the project.

Although the following subsections divide the narrative schema into three sections for easier reading, the narrative schema should be seen as one representation of what was reported in the six Test phases' performance reports. Some repetition may therefore be seen across the diagrams.

### 7.4.1 Notation of the narrative schema

The narrative schema was developed using the existing diagramming functionality of NVivo, the qualitative data management software application. This was used to graphically represent the contents of the textual data in the Test performance reports. Thus, the notation used in the narrative schema is that available in NVivo which provides a generic set of symbols that can be used by the researcher to draw diagrams of the underlying text. Recall that two types of sub-categories emerged (section 7.3.2.2): Selective and Theoretical, with the former representing elements (constituent parts) of the Test phase, and the latter representing the links between the elements. The selective sub-categories (nodes) were assigned different symbols; the theoretical sub-categories (links) were assigned a uniform symbol (diamond shape); the direction of the relationship is expressed as arrow going out from the source phenomenon/issue (arrow tail) towards the destination phenomenon (arrowhead) - see Table 7.8. The narrative schema depicts the sub-categories and the links among them rather than their higher level categories; because this offers an appropriate view of the various phenomena present during the Test phase execution and their influence on one another and on schedule delay.

| **Category/sub-category** | **Classification** | **Notation** |
|---|---|---|
| Dependency | Category | |
| Awaiting decision | Theoretical code | ◇ |
| Awaiting defect fix | Theoretical code | |
| Awaiting product | Theoretical code | |
| Awaiting resource | Theoretical code | |
| Delay in deciding | Theoretical code | |
| Delay in fixing defect | Theoretical code | |
| Delay in providing product | Theoretical code | |
| Delay in providing resource | Theoretical code | |
| People | Category | |
| Build manager | Selective code | ○ |
| Client | Selective code | |
| Consumer | Selective code | |
| Cross project delivery managers | Selective code | |
| Peer supplier | Selective code | |
| Solution architecture manager | Selective code | |
| Technical environment manager | Selective code | |
| Test data manager | Selective code | |
| Test manager | Selective code | |
| Tester | Selective code | |
| Product | Category | |
| Application server | Theoretical code | |
| Authentication tool | Theoretical code | |
| Build approach | Theoretical code | |
| Code | Theoretical code | |
| Code deploy tool | Theoretical code | |
| Component catalogue tool | Theoretical code | |
| Defect | Theoretical code | |

| **Category/sub-category** | **Classification** | **Notation** |
|---|---|---|
| Enterprise service bus | Theoretical code | ◇ |
| Introducing new product | Theoretical code | |
| Performance report | Theoretical code | |
| Regression Test tool | Theoretical code | |
| Scope of deliverables | Theoretical code | |
| Technical environment | Theoretical code | |
| Test completion report | Theoretical code | |
| Test data | Theoretical code | |
| Test data build | Theoretical code | |
| Test data requirement | Theoretical code | |
| Test environment | Theoretical code | |
| Use Case | Theoretical code | |
| Wireframe | Theoretical code | |
| Schedule | Category | |
| Activity delay | Selective code | △ |
| Duration compression | Selective code | |
| Duration extension | Selective code | |
| No schedule baseline | Selective code | |
| Phase delay | Selective code | |
| Start variance | Selective code | |
| Lacking control over progress | Theoretical code | ◇ |
| Nonworking time | Theoretical code | |
| Reporting progress status | Theoretical code | |
| Scope | Category | |

| **Category/sub-category** | **Classification** | **Notation** |
|---|---|---|
| Requirements gap | Theoretical code | ◇ |
| Scope change | Theoretical code | |

**Table 7.8 Notation of the narrative schema**

The aim of the research at this stage was to learn about phenomena most contributing to project delay present during the Test phases, therefore, the category ‘Schedule’ in table 7.8 has a special role since it captures the changes to the schedule reflected in the metrics in Chapter 5. For example, the sub-categories ‘Activity delay’ and ‘Phase delay’ relate to the behaviour of schedule delay as reported in section 5.3.3.2 (Chapter 5); the sub-categories ‘Duration extension’ and ‘Duration compression’ relate to the behaviour of schedule accuracy in section 5.3.3.6 (Chapter 5); the sub-category ‘Start variance’ relate to the behaviour of schedule change in section 5.3.3.4 (Chapter 5). This special role of ‘Schedule’, however, was not given superior explanatory power over the other categories; in fact all the categories were treated equally to ensure emergence of the phenomena in the data rather than enforcing preconceived ideas.

The relationships linking the nodes in the narrative schema may be seen as sequence of events taking place during the Test phase execution. It was noted in section 3.4.1 (Chapter 3) that, the textual reports, which the event sequence is based on, assumed considerable background knowledge of the projects and omitted generally known contextual information within ABC projects. Consequently, a gap developed in the narrative schema in some positions in the event sequence relating to the missing contextual information. These positions are assigned a grey hexagon (labelled ‘No reference to recipient’) in the narrative schema to indicate the location in the event sequence where we sought the support of the contextual information from Chapter 3 to fill the gaps during analysis. In the following, textual narrative information that was derived from the context presented in Chapter 3 is indicated by ‘(context)’, and information provided by project participants is indicated by ‘(participant)’.

The following sections present the narrative schema of the Test manager’s interaction with each of the three spheres (project participant groups) presented in Figure 3.6 (section 3.3 - Chapter 3): product development, product line development, and others - Table 7.9 lists the participants belonging to each of the groups. The particular perspective in this narrative schema is that of the Test manager as reported in the Test performance reports (solely written by the Test manager). However, it will be the case that the other phase managers will have different (valid) interpretations of the circumstances of the particular situation; and so other analyses could be done from the view point of various different roles. Nevertheless, the Test manager’s narratives were scrutinised by other project participants attending the project performance meetings, and therefore had to carry a degree of accuracy if the Test manager to maintain their professional integrity.

| **Sphere** | **Representative** | **Acronym** |
|---|---|---|
| Product development | Test manager | TM |
| | Build manager | BM |
| | Design manager | DM |
| | Project manager | PM |
| | Tester | TS |
| Product line development | Technical environment manager | TEM |
| | Solution architecture manager | SAM |
| | Enterprise service bus manager | ESBM |
| | Technical architecture manager | TAM |
| | Test data manager | TDM |
| Others | Client | CLT |
| | Peer supplier | PS |
| | Consumer | CONS |
| | Cross project delivery managers | CPDM |

**Table 7.9 Groups of project participants**

### 7.4.2 Product development narrative schema

The interaction of the TM within the product development sphere is depicted in Figure 7.4. In order to progress the Test phase the TM was awaiting a product (Code) and a defect fix from the BM. Although the BM provided code to the TM, the delay in providing the code and the defects in the code both caused activity delay and phase delay in the Test phase progress. The delay in providing product (Code) by the BM (context) caused start variance (starting Test phase execution late).

The TM was awaiting a resource (Design information) from the DM (context), and delay in providing the resource caused activity delay. The TM was also awaiting a resource (the TS) to perform test activities, however, the TS was on vacation (nonworking time) which led to activity delay. The Tester’s unavailability also caused delay in providing product (Code) to CONS. The TM reported progress status, delivered performance reports, and conveyed lacking control over progress to the PM (context).

The preceding text illustrates complexity of interdependent problems interacting with each other, and that the factors influencing schedule duration can be implicit; exerting their influence through means other than direct interaction - such as the Code causing schedule delay through the prevalence of defects.

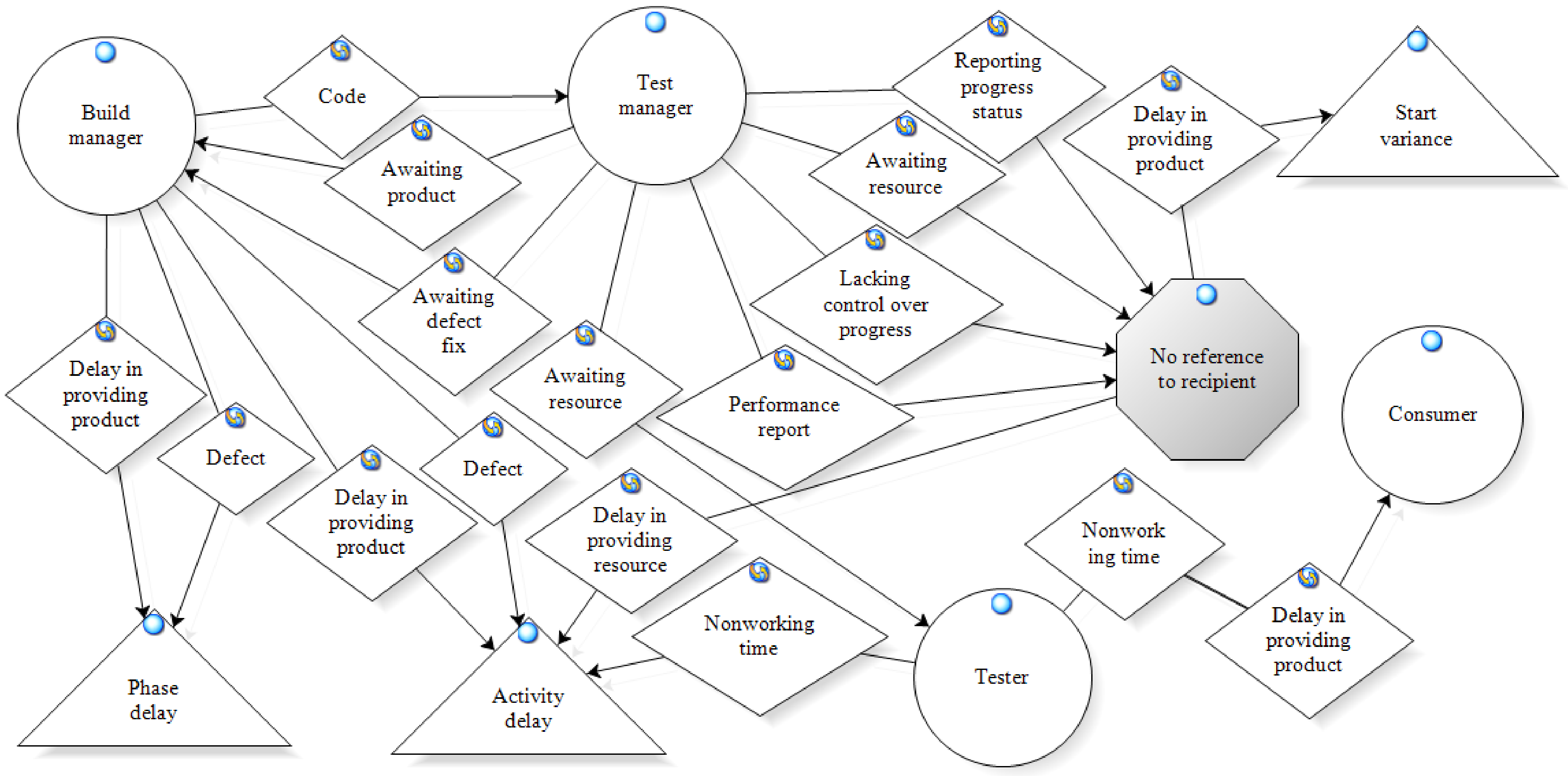


**Figure 7.4 Product Development narrative schema**

### 7.4.3 Product line development narrative schema

The interaction of the TM with the product line development sphere is depicted in Figure 7.5. The TM was awaiting a defect fix from the TEM which caused activity and phase delays. The TM was also awaiting defect fix from the SAM, which caused activity delay. The SAM provided a definition of the scope of deliverables to TM, and also scope changes during Test execution, resulting in the Test phase to operate without baseline schedule (see section 3.4 - Chapter 3), i.e. fluctuation of scope during phase execution. The SAM also provided Wireframes (instead of conventional functional design documents), which caused gaps in the requirements during Test phase execution. The TM received Use cases from SAM (participant) instead of conventional functional design documents, which also caused a requirements gap.

The TM was awaiting defect fix (of Authentication and Code deploy tools) from the TEM (context), but the defect caused activity delay, phase delay, and delay in starting the phase (start variance). The TM was also awaiting product (Test environment) from the TEM (context); however, delay in providing product caused activity delay, start variance, and duration compression. The TM was also awaiting resource (Technical architecture information) from TAM (participant), but delay in providing resource caused activity delay. The TM conveyed lack of control over progress to the PM (context).

Figure 7.5 shows a number of products affecting one another and causing schedule delay. For example, a defect in the Technical environment - managed by the TEM (context) - caused activity delay, phase delay, start variance, and duration extension. Furthermore, the defect in the Technical environment created defects in other products such as the Enterprise service bus (ESB) - managed by the ESBM (participant) - which in turn caused activity delay and delay in providing product; the Technical environment defect also created defect in the Regression Test tool - managed by the TEM (participant) - which in turn caused delay in providing product. In addition, defects in Technical environment caused delay in providing products (Code and Test environment) needed for testing.

Figure 7.5 shows other products causing problems. A defect in the Authentication tool - managed by the TEM (context) - caused activity delay, phase delay, and duration extension. Defect in the Code deploy tool - managed by the TEM (context) - caused activity delay and delay in providing product. Defect in the Component catalogue tool - managed by the TEM (context) - caused activity delay and delay in providing product. Introducing new products caused activity delays, such as: Authentication tool and Code branch (specific area on the configuration management system to store and maintain versions of the developed code) - managed by the TEM (participant), and new ESB Code - managed by the ESBM (participant). Application server upgrade - managed by the TEM (participant) caused defect in the ESB. Defect in the Test environment. A defect in the Code caused delay in providing product (Code) and scope change (scope increase due to deferred code defects from previous Test phase).

Finally, Figure 7.5 shows that the TM submitted their requirements of Test data to the TDM.

The interdependency and influence of project products on one another, during project execution, can be noted from the preceding text. The complexity of the sequence of events led to products to cause problems in other products, influencing schedule duration, and eventually leading to schedule delay.

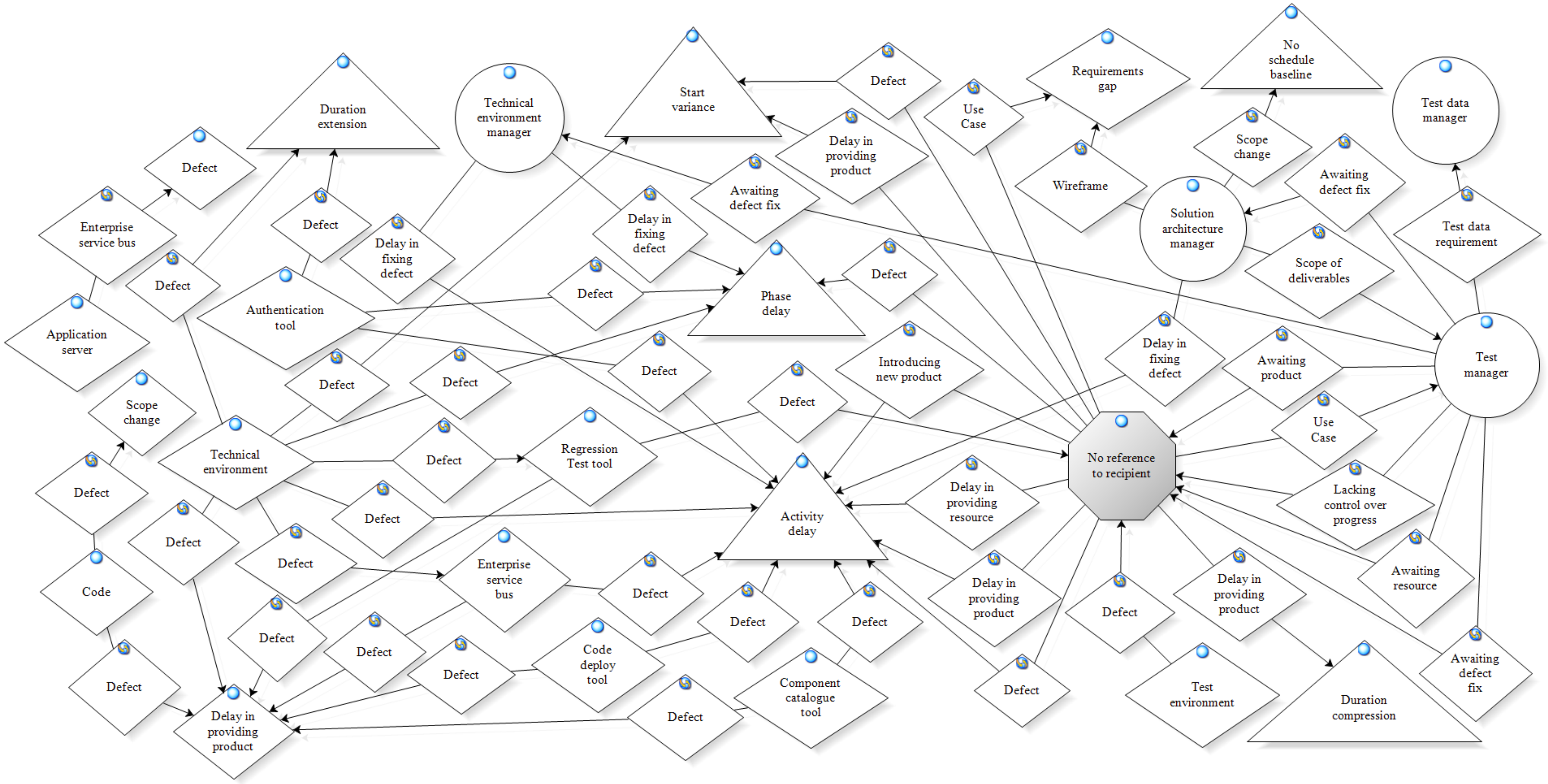


**Figure 7.5 Product Line Development narrative schema**

### 7.4.4 Others narrative schema

The interaction of the TM with the ‘other’ project participants is depicted in Figure 7.6. The TM was awaiting decision (on scope change) and resource (clarification on requirements gap) from CONS. However, a delay in providing resource caused activity and phase delays. Furthermore, requirements gap (i.e. the need for a clarification of the requirements) from CONS caused defect, which in turn caused activity delay; and that there was scope change from CONS. The TM delayed providing product (tested Code) to CONS because the TS was away.

The TM was awaiting decision (on scope change) from CLT; however, delay in deciding caused phase delay. The CLT requested scope change from TM (context). The TM submitted Test completion report to CLT (context), and was awaiting decision (sign-off); however, delay in deciding caused phase delay.

The TM was awaiting decision on Build approach; because of a defect discovered during the Test phase execution, which was blocking progress, and the fix of which required agreement with the CPDM as it was impacting other dependent projects; however, delay in deciding caused phase delay.

The TM was awaiting product (Test data and Test data build) from PS; whilst the PS provided Test data and Test data build to TM, delay in providing the products to TM (context) caused phase delay and duration extension. A defect from PS caused activity delay and scope change; the TM was awaiting defect fix from PS, but delay in fixing defect caused activity delay.

The TM conveyed lacking control over progress to the PM (context).

The influence of the external parties to ABC on controlling schedule duration becomes visible in the emergent picture, and that this influence is sometime indirect; for example, through delay in signing-off project deliverables.

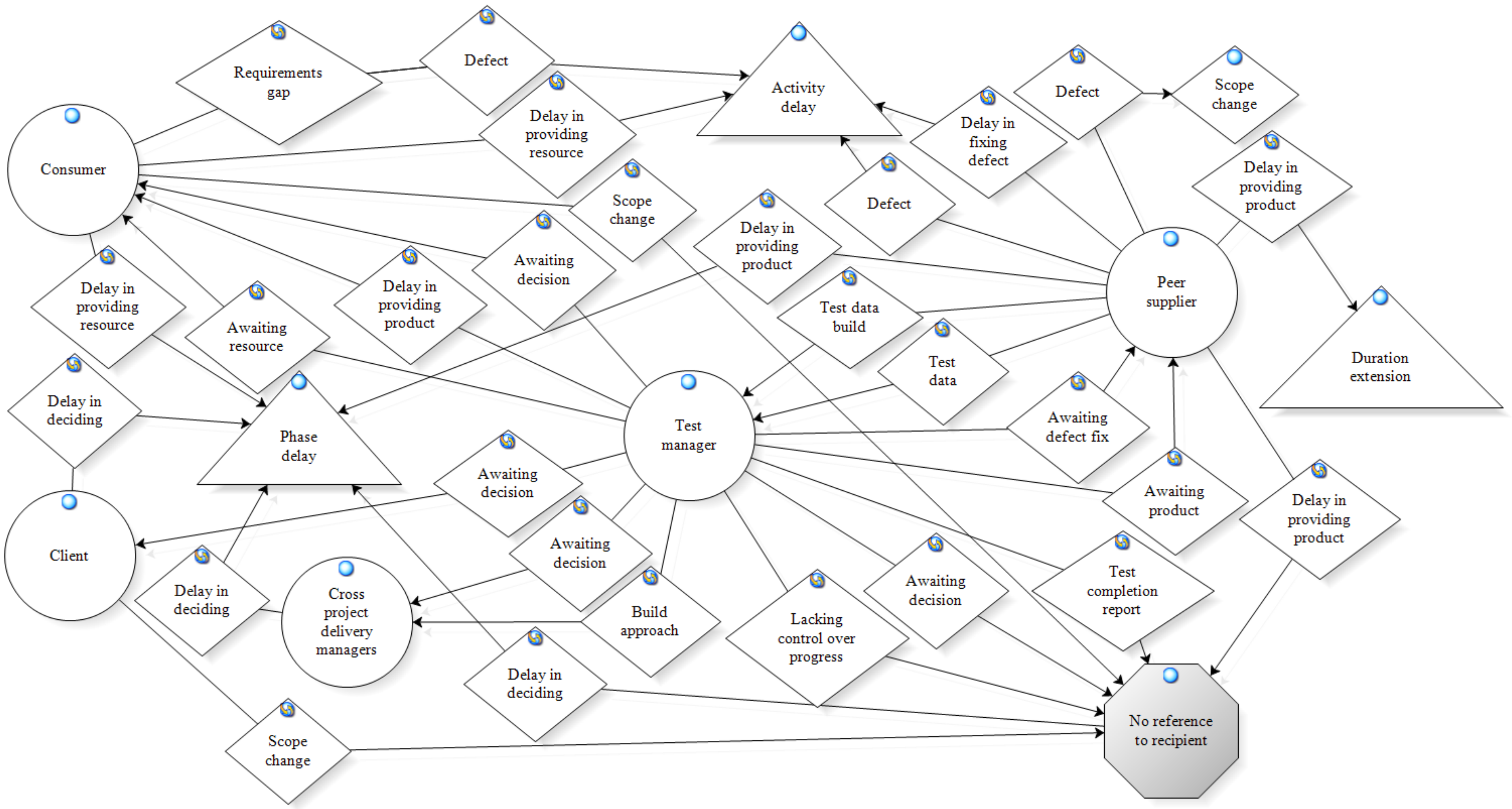


**Figure 7.6 Others narrative schema**

### 7.4.5 A conclusion on the narrative schema

The preceding sections illustrate that the narrative schema is a representation of the textual content of the Test progress reports. This is not a model in the sense that in its current state it can be generalised. For instance when it shows 'defect' it is referring to a specific mention of defects in the progress reports. Some of the situations could have different outcomes to the ones identified in the narrative; e.g. defects in project products may lead to a duration extension in the situation described in the text, but will not be the case with all defects (e.g. they could catch up). Furthermore, the challenges inherent in developing software in a globally distributed environment (see sections 3.2.2 and 3.3 - Chapter 3) seems to have not been mentioned in the progress reports; this may be due to the assumption, by the development teams, that they ought to operate within this environment and therefore no point in complaining about obstacles relating to the distributed nature of the work.

In addition, it was seen that the narrative schema was incomplete and, in some situations, needed the support of the contextual information and/or input from the participants to make sense of what was happening - Tables 7.10 and 7.11 summarise when this support was needed (indicated by an 'x' against the source). In the tables' header, *n* refers to the narrative schema, *c* to the contextual information from Chapter 3, and *p* to the project participants input.

| **People** | ***n*** | ***c*** | ***p*** |
|---|---|---|---|
| Build manager | x | | |
| Client | x | x | |
| Consumer | x | | |
| Cross project delivery managers | x | | |
| Design manager | | x | |
| ESB manager | | | x |
| Peer supplier | x | | |
| Project manager | | x | |
| Solution architecture manager | x | | x |
| Technical architecture manager | | | x |
| Technical environment manager | x | x | x |

| **People** | ***n*** | ***c*** | ***p*** |
|---|---|---|---|
| Test data manager | x | | |
| Test manager | x | | |
| Tester | x | | |

**Table 7.10 People information to support the narrative schema**

| **Product** | ***n*** | ***c*** | ***p*** |
|---|---|---|---|
| Application server | x | | x |
| Authentication tool | x | x | |
| Build approach | x | | |
| Code | x | | |
| Code deploy tool | x | x | |
| Component catalogue tool | x | x | |
| Defect | x | | |
| Enterprise service bus | x | | x |
| Introducing new product | x | | x |
| Performance report | x | | |
| Regression Test tool | x | | x |
| Scope of deliverables | x | | |
| Technical environment | x | | |
| Test completion report | x | x | |
| Test data | x | x | |
| Test data build | x | x | |
| Test data requirement | x | | |
| Test environment | x | x | |
| Use Case | x | | x |
| Wireframe | x | | |

**Table 7.11 Product information to support the narrative schema**

## 7.5 Summary

This chapter set out to answer the second research question:

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

The QUAL approach categorised the phenomena present during the execution of six Test phases across three projects. A total of 213 phenomena (159 codes, 49 sub-categories, and 5 categories) emerged. The thesis then developed a narrative schema depicting the relationships among the sub-categories as present in the content of the textual data in the Test phase performance reports, which showed, to a considerable extent, the factors that influenced schedule duration across the six cases leading to schedule delay from the perspective of the Test manager. This, however, needed the support of the contextual information and input from the project participants to obtain a fuller picture of the sequence of events influencing schedule delay in ABC. As was seen that a modern software development project can be very complex; it involves interaction among various actors (human and nonhuman) who to varying degrees influence schedule duration.

Although the findings from this chapter answered the RQ2; they are very local to ABC, and are interpretations by the participants of what was happening. In order to take the study to the next level of understanding and be able to extend the findings to a broader range of experiences/contexts of other researches, an explanatory model was developed (using Actor-network theory) to understand the causes of schedule delay - the subject of Chapter 8.

# 8 Explanation development

## 8.1 Introduction

This chapter describes the process of developing explanation of schedule delay and presents its findings; the final step in the research process outlined in Chapter 4: Figure 4.1 and section 4.4.4. The chapter attempted to answer the third research question:

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

The investigation of software development projects at ABC in the previous chapters has revealed that projects contain different actors - such as design, build and test specialists and their managers - and technical artefacts - such as code and development and test environments - which support and constrain each other in a network of interactions. It will be seen in this chapter that this picture coincides with conceptualisation of organisational behaviour known as actor-network theory (ANT). The fit between the current research and ANT is explored and where possible ANT concepts are used to explain the behaviours found in the projects investigated. Finally, the research journey, as a whole, is reflected upon.

## 8.2 The fit between the current research and ANT

Examining the narrative schema in sections 7.4.2 - 7.4.4 (Chapter 7) indicates that, varied phenomena were in a network of interaction during the execution of the Test phases, where some were nodes - e.g. people - and others as links that connect the nodes - e.g. products. This suggests that the project can be seen as a network of actors and intermediaries, both human and nonhuman, each of whom can, to varying degrees, empower or constrain others. For example, the ‘People’ category are human actors, the ‘Product’ cases can usually be seen as intermediaries. The narrative schema that came out of the Grounded theory (GT) analysis seemed aligned to that expected by Actor-Network Theory (ANT), and therefore the ability of ANT to explain/illuminate the picture that had emerged seemed to demand some attention. Thus, Table 8.1 applied

ANT concepts to the GT categories, followed by description of the ANT concepts (*in italics*) and how they might be applied within the software engineering management field. This will be illustrated by examples drawn from, but not limited to, ABC's software project management practices.

| GT category | Description | Relates to | ANT concept | Description | Indicator of problematic project |
|---|---|---|---|---|---|
| Dependency | Test manager was dependent on other project participants, for provision of products and/or services, in order to maintain progress of the Test phase on schedule | Reliance on someone or something for resources in order to carry out tasks | Coordination | The extent to which a network is governed by principles, such as rules (explicit agreement) or conventions (a way in which things are usually done), governing the interaction among the actors dependent on one another, aiming to stabilise the actor-network | Weak coordination: project conventions (rather than rules) govern the interaction among the actors; conventions are not widely accepted by project actors, which exert constraining influences on achieving project objectives |
| People | Human project participants who to varying degree contribute to the Test phase progress | The ability to do or influence something; and  the needs for resources in order to carry out tasks | Actor | An element within the network of associations that has the ability to exert influence on the other elements in the network; that is, it can act. Most actors can be seen as in fact actor-networks. They are effectively a representative of a group of actors | A project with a large number of actors is likely to be more complex (and therefore more problematic) than one that has very few actors |

| GT category | Description | Relates to | ANT concept | Description | Indicator of problematic project |
|---|---|---|---|---|---|
| Product | Artefacts produced by project participants for use by other project participants which enable further producing project deliverables | Things that circulate/flow between people; i.e. which link people together | Intermediary and Inscription | Intermediary: an element within the network of associations that facilitates interaction between actors; they are the relationships/associations the actors forge to enable interaction. A product can become a nonhuman actor (see 1st paragraph after this table)<br><br>Inscription: embedding programs of action in technical artefacts to influence the artefact user to operate in a certain way | Intermediaries in the project become ‘*mediators*’. Whilst *intermediary* transports meaning or force without transformation; a *mediator* may change the input in some way before they pass it on (Latour 2005, page: 39), thus adding uncertainty to the progress of the project<br><br>Programs of action can be weakly inscribed, leading to weakening irreversibility (see final item in this table). A ‘strong’ program of action is not just one that is detailed and enforceable. It needs to be |

| GT category | Description | Relates to | ANT concept | Description | Indicator of problematic project |
|---|---|---|---|---|---|
| | | | | | widely accepted - i.e. contribute to alignment |
| Schedule | The means by which project/phase activities were ordered on a timeline showing what activity was planned to be carried out, by when, and in what order; for example, Gantt chart | Organisation of activities according to certain rules | Coordination | The extent to which a network is governed by principles, such as rules (explicit agreement) or conventions (a way in which things are usually done), governing the interaction among the actors dependent on one another, aiming to stabilise the actor-network | Weak coordination: project conventions (rather than rules) govern the interaction among the actors; conventions are not widely accepted by project actors, which exert constraining influences on achieving project objectives |
| Scope | The total number of products and services agreed to be delivered as part of the project or within a phase in a particular point in time; scope can change during | Increasing stability in project execution through defining the boundary of the work and reducing uncontrolled change | Inscription and Irreversibility | Inscription: embedding programs of action in technical artefacts to influence the artefact user to operate in a certain way | Programs of action can be weakly inscribed, leading to reducing irreversibility (see next). A ‘strong’ program of action is not just one that is detailed and enforceable. It needs to be widely accepted - |

| GT category | Description | Relates to | ANT concept | Description | Indicator of problematic project |
|---|---|---|---|---|---|
| | project execution due to change requests | | | Irreversibility: the degree of stability of an actor-network and its resistance to going back and changing things that have already been done | i.e. contribute to alignment<br><br>Unstable project; prevalence of disorder in project activities; disruption in producing deliverables |

**Table 8.1 Application of ANT concepts to GT categories**

In ANT terminology, an interaction between *actors* is facilitated by some form of *intermediary*. It could be, but is not limited to, text inscribed and circulated on paper or an electronic medium (Callon 1991, page: 135) as with a test performance report. In Figure 8.1, below, a group of actors (including Design, Build and Test managers) work to perform a software development task using intermediaries (such as a functional design and design defect reports) to coordinate their activities. Actors and intermediaries can be human or nonhuman, and can be called ‘actant’ as a generic term to avoid separating the human from nonhuman (Akrich & Latour, 1992; Bijker & Pinch, 2012). An example of nonhuman actor might be where legacy software is involved: the complexity of its structure and the dependence of existing users on the system will influence the behaviour of other, human, actors. An *intermediary* (the relationship/association among actors) can itself become an *actor* (i.e. it can act) by putting other intermediaries into circulation (Callon 1991, page: 141); for example a software component under construction can have errors (code defects), the correction of which absorbs effort and causes delays; this is very similar to what Latour (2005) calls *mediator*; whilst *intermediary* transports meaning or force without transformation; a *mediator* may change the input in some way before they pass it on (page: 39); for example, a product behaves contrary to (e.g. generates defect) how it was intended to operate (e.g. to provide a service), thus adding uncertainty to the progress of the project. Thus, *actors* can be seen as elements of a project that interact through *intermediaries*.

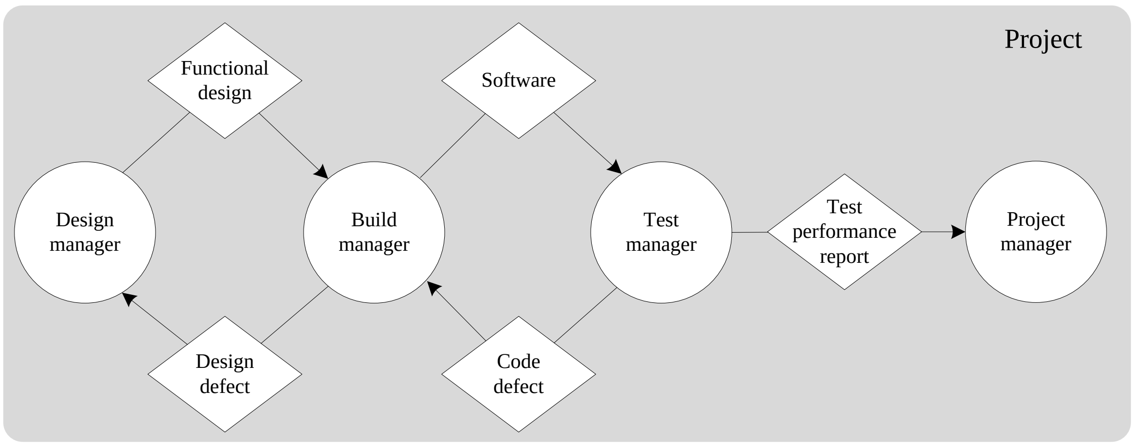


**Figure 8.1 Software development process**

A typical software development process (Figure 8.1) includes the Design, Build, and Test phases. The Design manager (*actor*) delivers functional design (*intermediary*) to the Build manager (*actor*). The Build manager registers design defects (*intermediary*) for the Design manager to resolve when seeking clarity on the functional design. The Build manager delivers software (*intermediary*) to the Test manager (*actor*). The Test manager registers code defects (*intermediary*) for the Build manager to fix when defects are discovered during testing of the Software. The Test manager produces regular test performance reports (*intermediary*) to inform the Project manager (*actor*) of progress of the Test phase execution. Thus, the category people in Table 8.1 can be seen as actors because they can create and circulate intermediaries; and the category products as intermediaries because they are created and circulated to others.

The actors in Figure 8.1 can be seen as *actor-networks*; they are effectively representatives of a group of actors, intermediaries, and their interactions (Callon, 1991). These *actor-networks* can be thought of as *black-boxes*. In ANT terminology, a *black-box* is an artefact with a number of elements (which itself would be a *network*) whose internal interaction is concealed. An outsider interacts only with the artefact’s external features, which may be a few well defined parameters (Callon, 2012; Monteiro, 2001). For example, software testers may be interested in the external behaviour of a software component and not in its internal workings. They treat the internal structure as a ‘black-box’ and simply check that the inputs and outputs conform to the functional design. Actors make their relationships with large and complicated actor-networks easier by treating them as a *black-box* (Law 2012, page: 125). This way, the black-box can join other actor-networks as a *punctualised* entity within these networks. *Punctualisation*, in ANT terms, converts an entire actor-network into a single point or node in another actor-network (Callon, 1991); this node can play the role of actor or intermediary in the new network. Alternatively an individual can be treated as a representative of a broader actor-network. Thus, the Design, Build, and Test phases each can be seen as individual actor-networks comprising a team that carry out daily tasks needed for that phase and a phase manager who represents the team to the outside world (the project).

The mechanism for embedding programs of action in technical artefacts (e.g. the functional design in Figure 8.1), with the aim of guiding the artefact user to operate in a certain way, is called *inscription* in ANT terms (Akrich & Latour, 1992; Latour, 1991). For example, requirements *inscribed* into functional designs which in turn *inscribed* into computer code, later influences the software user to operate in certain ways to carry out routine operations. A *weakly inscribed* program of action weakens the *irreversibility* of an *actor-network*. A *strong inscription* resists reversibility attempts (Monteiro, 2001).

*Irreversibility* in ANT refers to the degree of stability of an established *actor-network* and its resistance to going back and changing things that have already been done. A reversible actor-network is unstable; it is prone to influences exerted by internal and/or external forces attempting to reconstruct the network according to these influences (Law 2012, page: 115). An *irreversible* network brings stability; it resists these influences, which can be achieved through maintaining the *alignment* and *coordination* (see further down) between the actors and the overall control of the project manager to achieve project objectives (Law, 2012). The degree of *irreversibility*, therefore, is related to the extent of resistance to changing the *inscriptions* that have already been circulated (Monteiro 2001, page: 79), which make it difficult to deconstruct the network/association and establish a different one (Callon 1991, page: 150).

For example, requirements informally described by the client may be *weakly inscribed* during the Design and lead to reversibility at the Build and Test phases if the client then modifies their requirements. '*Weak inscription*' here refers to 'room for interpretation' as well as poor definition of system requirements; for example, a requirements document could be accurate but there may be lots of different ways that it can be implemented. The functional design phase selects a design which will meet those requirements, but the software developers will have some scope in deciding how that design will be converted into code. A stable actor-network enables steady progress in producing project deliverables and can be said to be black-boxed. Although, *irreversibility* may sound contrary to the desirable quality of agility in software projects, there is a need even for software produced using agile approaches to become eventually a stable project deliverable. Thus, the category scope in Table 8.1 can be paralleled with inscription and irreversibility; since when scope changes during project execution it indicates weak inscription of the requirements; this also indicates reversibility because it results in going back and changing things that have already done. The category product in Table 8.1 can also be seen as inscription because it embeds programs of action for users to use the product in certain ways.

ANT can be applied to non-projects as well as projects; whether during project setup or project execution (McLean & Hassard, 2004; Bloomfield & Vurdubakis, 1997). An

ANT study can examine the process of constructing a network, called *translation* in ANT (Callon, 1986; Monteiro, 2001; Callon & Law, 1989; Walsham & Sahay, 1999), through focussing on the attempts of the *focal actor* - an actor of interest to the area under study whose viewpoint of the network is being examined such as Test manager - to assemble relations among heterogeneous elements and align their interest to achieve particular purpose (Law, 1999). The Test manager position themselves at the centre of the network - an *obligatory passage point* (OPP) in ANT terms, in order to exercise control at a distance drawing from the strength and credibility of others (Law, 1986). One way of achieving this is through *inscribing* these interests into all sorts of *intermediaries* and circulating them to the target actors (Callon, 1991).

An ANT study can also investigate the operation of an already established actor-network (e.g. project); examining the interactions among the *actors* and *intermediaries* which are well understood and accepted, Callon (1991) called this the *dynamics of networks* and refer to it as 'the complex process in which actors and their talkative (sometimes indiscreet) intermediaries weave themselves together' (page: 144). For example, Figure 8.2 shows ABC's version of the iterative and incremental development (IID) model described in detail in section 3.2.3 (Chapter 3). This approach and the roles needed for its implementation are well understood, even before a new project is planned. Some elements of the project, for example, relationships with new client may need new working relationships to be formed that will involve translations.

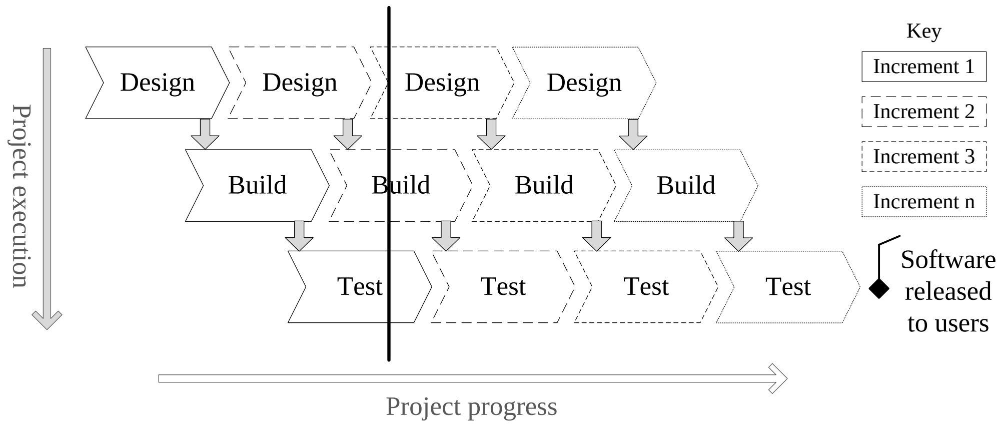


**Figure 8.2 ABC's software engineering method**

As noted in section 3.2.3 a problem controlling this version of IID (i.e. semi-parallel execution of increments) is that at any one time, the functional project teams will be working on different increments of the same project. This may be a problem when one of the specialist teams needs to call upon the services of another. For example, the Test phase in increment 1 may require fixes of the code developed by the Build phase in Increment 1. However, at the time of executing the Test phase in Increment 1 (see the solid vertical line cutting through the phases) the Build resources are working on building Increment 2 of the functionality which leads to an issue about how the Build team should prioritise the competing demands on their services. ANT offers the concepts of *alignment* and *coordination* to make sense of such interactions among project actors (Callon 1991, page: 152). The *dynamics of networks* ought to be supportive of achieving the network objectives if the *actor-network* is to succeed.

*Alignment* indicates the degree of agreement between the actors on, and their commitment to, their role in the *network* (Callon 1991, pages: 144-146). In project terms, during the execution of a software project, the Project manager attempts to maintain the Design, Build, and Test managers' alignment to the set targets agreed prior to the execution of their corresponding phases in order to meet project end date. Accomplishing *alignment* is, therefore, attempted during project set up; i.e. through the *translation* process described earlier.

*Coordination* in ANT refers to the extent to which a network is governed by principles, such as rules (explicit agreement) or conventions (a way in which things are usually done), governing the interaction among the actors dependent on one another, aiming to stabilise the actor-network (Callon 1991, page: 146-147). For example, the Build manager (in Figure 8.2), whilst focused on developing the code in Increment 2, is also fixing defects of the code developed in Increment 1; because the Fix team (who is part of the Build team) are selected members of the Build team dedicated to perform fix activities for the code they developed in Increment 1. However, no prior agreed timeline (rule) existed to provide such fixes to the Test manager of Increment 1, it was done by convention. A network governed by convention exhibits '*weak co-ordination*', which exert constraining influences on achieving network objectives. A *network* governed by rules, exhibit '*strong co-ordination*', which exerts empowering influences on achieving network objectives. Thus, absence/lack of acceptance of rules among the three phases/*actor-networks* above results in *weak coordination*. As will be seen that managing dependencies and defining and enforcing rules will become particularly challenging during project execution among the Design, Build, and Test *networks* - see

Figure 8.2. The weekly progress meetings held by ABC projects were part of the coordination process. The creation of the original schedule was also part of the coordination process. Thus, the category schedule (Table 8.1) can be seen as a coordination mechanism. There was a rule ‘phase teams must do everything to conform to the plan’, because delivering on schedule was the key success measure in ABC. The category dependency in Table 8.1 can be seen as a way of coordinating the relationships among project activities/phases, since a dependent activity in later phases is affected by an activity in the earlier phases.

The ‘weak’ and ‘strong’ modifiers used with ANT concepts above indicate points on a continuum rather than a yes/no state, and have been replaced here by ‘constraining’ and ‘empowering’ respectively to indicate the influence exerted.

## 8.3 An explanatory model of schedule delay

In order to make sense of the complexity emerged in the preceding analyses; a model (i.e. a simplified representation) is needed which illustrates the constraining and empowering influences at play during testing which caused schedule delay. One way to achieve this is to model the interactions among the project participants (Abdel-Hamid & Madnick 1991, page: 7) which converts the underlying complex reality into a comprehendible view to enable reasonable analysis (DeMarco1982, page: 41). As noted earlier that, developing an ANT model involves modelling the interactions among the actors in the network, which could be a graphical representation of a network comprised sequences of points and lines (Callon 2012 page: 90).

The analysis in this research selected the Test manager as its focal actor because the research was concerned primarily with the problems of testing. The ‘focal actor’ is the actor from which the analysis departs in following its interactions in the actor-network (Callon & Law 1989, page: 77-78). This is to allow the investigator to set practical limits on the analysis rather than to privilege one actor (Law, 2012). Not everyone at ABC was focused on maintaining testing on schedule at the expense of such things as ensuring new functionality did not degrade existing functions. While the Test manager’s perspective on the actor-network is reflected in the analysis, it is discussed in the context of the other actors concerned with project progress who have different perspectives, and which their interaction and influence appear in the analysis (Law, 2012).

The Test manager (TM), among other things, speaks on behalf of a team of testers, and thus can be seen as a representative of the Test actor-network. Similarly, the other project participants (see Table 7.9 - section 7.4.1 in Chapter 7), who contribute to the completion of the Test phase, can be seen as representatives of individual actor-networks (i.e. their respective teams). Thus, the developed ANT model (Figure 8.3) from the interaction of the TM with all project participants should be seen as interactions among various actor-networks (black-boxed in ANT terms - section 8.2), represented by their respective manager. In this model a solid-line diamond between the actor-networks indicate an association that exerts empowering influence on maintaining progress of the Test phase on schedule; in contrast a dotted-line diamond indicates a relationship that exerts constraining influence on maintaining progress of the Test phase on schedule; i.e. it causes schedule delay.

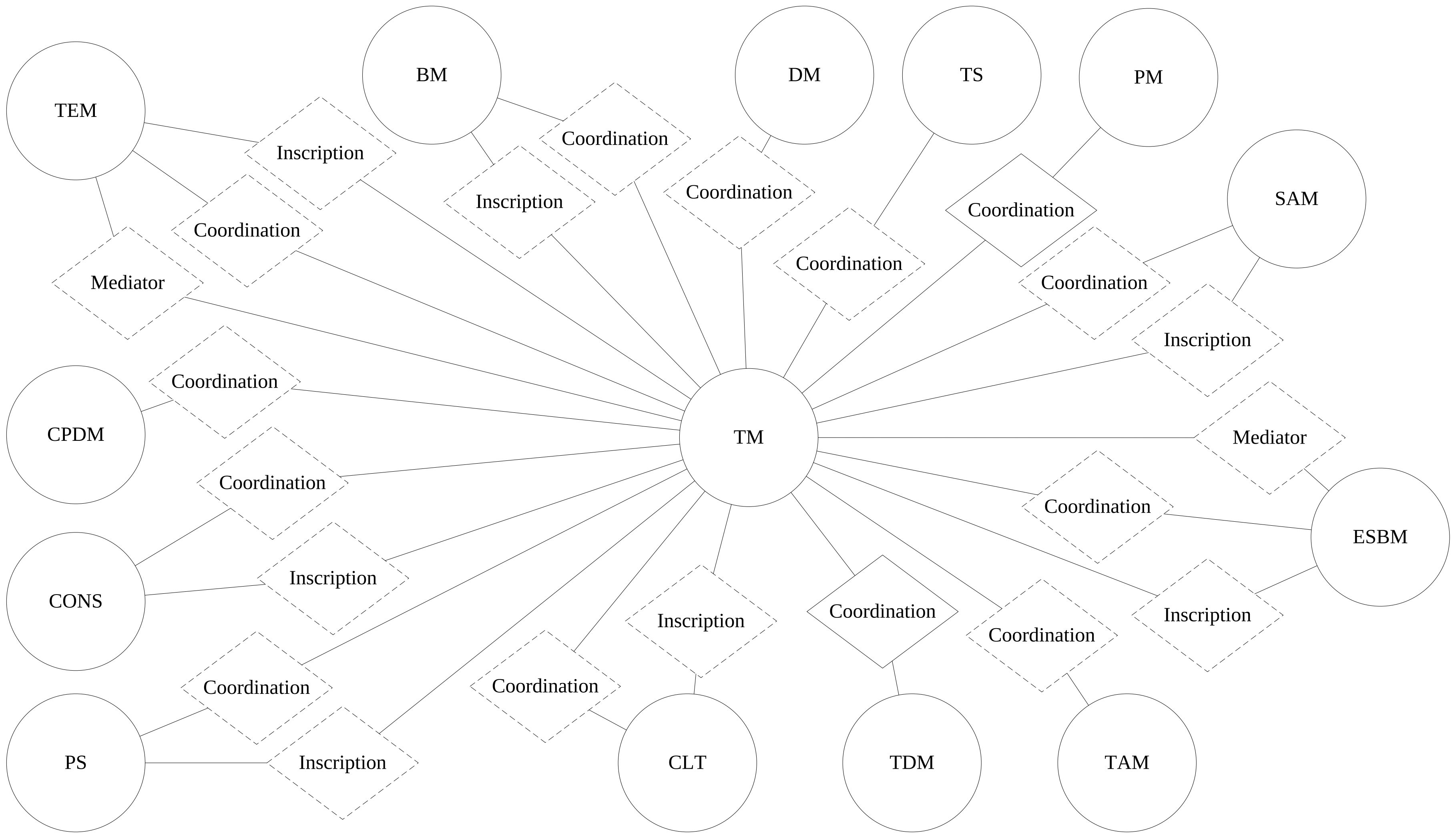


**Figure 8.3 ANT model of schedule delay**

The emergent picture shows prevalence of constraining influences on maintaining the progress of the Test phase on schedule; through constraining coordination, large number of actor-networks, intermediaries transforming to mediators, and constraining inscription between the Test actor-network and most of the actor-networks. The limited empowering influences to maintain progress can be seen in the interaction between TM with PM, and TM with TDM showing empowering coordination.

The constraining coordination, shown in the model, indicates that the various actor-networks focused on the rule 'phase teams *must* do everything to conform to the plan' in relation to the phase for which they were responsible because delivering on schedule was the key success measure in ABC (see Figure 8.2 - section 8.2). For testing there is another rule 'all defects must be detected and removed' which has priority over the deadline rule. Therefore, any supporting activities to other phases (such as the Test phase) may have been viewed as convention rather than rule. A convention 'phase teams *should* support other phase teams, especially in testing' would be interpreted in light of the priorities of the support provider. Thus, this convention is not widely accepted by all project actors; leading to delay in providing products and services to the TM, and eventually to schedule delay. If the BM slowed down on their work for the next increment in order to deal with fixes to last increment this too would delay final project completion. Although, the empowering coordination with the PM may have eventually brought progress back on schedule through the PM negotiating priorities with other managers in the weekly performance meetings, this was after event targets were missed.

The large number of actors (or actor-networks) in the model shows the complexity of the development process that the TM is attempting to grapple with; added to this, the complexity inherent in coordinating project activities between UK and India (see section 3.2.2 - Chapter 3; section 7.4.5 - Chapter 7), which is abstracted away by the model, but which makes the interaction among the actor-networks even more complex. Section 2.4.1 (Chapter 2) described the complexity of such projects as the interrelationship between the actor-networks in terms of: the number and variety of actors and intermediaries, the number and variety of interactions among actors (that exert constraining or empowering influences), the number and variety of interdependencies (that exert direct or indirect influences), and the rate of change of the project situation/context. Such projects, as the ones in Figure 8.3, are more problematic, than one that has very few actors; and therefore more difficult to control.

The model shows intermediaries that became mediators; behaving contrary to how they were intended to operate; that is, products (that were intended to provide services) created other products (defects), which affected many other products, which then led to schedule delay (see section 7.4.3 - Chapter 7). Attempting to anticipate this kind of chain of events and factoring them into the project plan prior to execution is a challenging task. Intermediaries transforming to mediators, during project execution, create uncertainty in managing such projects, because they demand additional time and effort.

The constraining (i.e. incorrect/incomplete) inscriptions emerged in the model indicate that the programs of action put in place among the actor-networks did not produce the intended outcome. That is, the products that were created to support the TM in executing the activities of the Test phase did not operate properly to enable TM to maintain the progress of the Test phase on schedule (see section 7.4.3 - Chapter 7). These products kept breaking down during phase execution leading to schedule delay. In addition, some of the user requirements were not appropriately inscribed into functional designs; some of the designs were not appropriately inscribed into code; and some of the code was not appropriately inscribed to reflect the designs. Thus led to various defects (design defects and code defects) emerging during the Test phase execution. Furthermore, the various change requests put forth by various actor-networks (section 7.4.4 - Chapter 7) during the development process indicates constraining inscriptions of the intended functionality needed to be developed at the end, which led to reversibility to the previous stages of development.

The reduced irreversibility indicates incorrect/incomplete inscription of the various products as noted in the preceding paragraph. Reversibility to the previous stages created disorder in project activities. For example, going back and changing user requirements, design and code statements, as well as to clarify what was needed created disruption. The resulting changes had to be retested by the TM within constraints of the original Test phase schedule. Reversibility destabilised the Test phase progress for the TM and led to schedule delay. ‘Traditional’ project planning is based on the assumption that the nature of future activities can be forecast with certainty and project managers can have perfect control over them. This research shows that this is not usually the case.

ANT provided a layer of insight into the causes of schedule delay that was not possible from the interpretive findings in Chapter 7 - which is much localised to ABC. ANT surfaced the problems inherent in the development process, the complexity of the delivery model, and the influence of the various parts of the overall system that hindered progress causing schedule delay.

The dominance of the constraining influences in the emerged picture (Figure 8.3) indicates that the TM had limited/no control over progress in order to maintain phase execution on schedule. While the TM is formally responsible for the progress of testing, actual progress is governed by factors outside their control. The more defects that are found, the more effort is needed to correct them. The effort needed to correct them requires services from Build, Design, and Technical environment, but these have competing demands made on them. One reason for the competing demands is the parallel incremental development approach which means work is going on for the next increment while there is a demand for remedial work for the previous increment leading to resource clashes. This contradicts the assumption that the project manager (the Test manager in focus) ‘owns’ and controls all the resources allocated to the project. In fact, the system forces the project to interact with other actor-networks, outside the control of the Test manager, that constrain the TM from achieving its objective in maintaining progress on schedule.

Hughes (2012) noted that a project has a context/environment not under the control of the project manager, yet to reduce sources of uncertainty, the project manager attempts to bring the context under the control of the project. Further, Law & Callon (1992, page: 46) noted that the ‘shape and fate of technological projects’ is a function of the project manager’s building of and maintaining the alignment and coordination of (i) the external actor-networks which the project is dependent on for products and services (ii) the internal actors to produce project deliverables (iii) maintaining control over events (obligatory passage point - OPP) which take place between the internal and external networks during project execution (Callon & Law, 1989; Law & Callon 1992). The absence of OPP may lead to lack of control on the project and eventually project failure (Law & Callon 1992). In practice, the OPP may move/shift from initially being the focal actor to become some other actor/s; what emerged from the ANT model is that, the TM is not the OPP, rather it is TEM who controls most of the products used by TM to carry out Test phase activities; and hence, controlling the events during project execution and influencing project success. Elbanna (2008)'s empirical study on Enterprise Resource Planning implementation, suggests that the examined project tended to drift from the plan because the actors, that were dependent on one another, slowed down or speeded up progress of work activities within the network based on their own priorities; in this way, the various project actors influenced the execution path of the project by being a positive force towards achieving project objectives or a negative force that impeded project progress.

It can be argued that product development projects contain a degree of uncertainty and instability, and that their outcome cannot be accurately predicted (Akrich et al. 2002). However, the stability of a project may be increased through the network of associations built by the project manager and the influences the project manager exerts, through the network, to maintain that stability (Latour 1986, page: 267). Hanseth et al. (2006)'s empirical study indicate that the complexity of the information technology project studied, which was operating in very open environment and used a number of very different stakeholders, led to unexpected results through emergence of side effects that generated new actions, which had their own side effects and so on, which then led to disorder and eventually to the failure of the project despite the application of project management best practices. The constraining coordination and inscriptions, the complexity of the actor-networks and emergent mediators, and the reversibility to previous phases emerged under the ANT lens, in this research, have constrained the Test manager's control on schedule duration and led to schedule delay.

In conclusion, the nature of an actor-network of being constructed from heterogeneous entities makes it prone to divergence in some way or another during its lifetime. Hence, the project manager ought to maintain the relations, with other actor-networks, in place (stabilise) during execution to avoid getting out of control. Law (2012) noted that 'large-scale heterogeneous engineering is difficult. Elements of the network prove difficult to tame or difficult to hold in place. Vigilance and surveillance have to be maintained, or else the elements will fall out of line and the network will start to crumble' (page: 108). The ANT model (Figure 8.3) illustrates the complexity of the projects undertaken; the numerous links/interfaces between project elements demands additional time from the project schedule that were not accounted for during planning. The more parts a project needs to interact with in order to carry out its activities, the more complex the project becomes, this complexity adds to the project schedule, and if was not incorporated in the original plan, it may lead to schedule delays. On partitioning the model of a project system, DeMarco noted long ago that ‘each time you partition, there are interfaces to be considered, interfaces between the various pieces of the partitioning. There may be a choice of ways to divide a whole into pieces. When this is true, the complexity of the resultant interfaces is a clue to which way is better. A partitioning with few and simple interfaces is preferable to one with many and complex interfaces’ (1982, page: 43). Thus, the development and delivery model in ABC ought to be simplified if it were to hand control back to the project manager and deliver projects on schedule.

This chapter attempted to answer the final research question:

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

The approach to developing explanation applied Actor-network theory concepts to the Grounded theory categories of phenomena present during Test phase execution, to examine the interaction developed among the project actors, culminating in an ANT model which identified the influences on schedule delay. As can be noted that, controlling schedule duration in software projects can be influenced by the interaction of heterogeneous elements (human and nonhuman), and that these influences emerge at the time of project execution, which can exert constraining influences on the progress of project activities and cause schedule delay. Furthermore, the effect of a project’s context on the progress of the project activities cannot be overlooked; it contradicts the assumption that the project manager ‘owns’ and controls all the resources allocated to the project; in fact, the system forces the project to interact with other actors, outside the control of the project manager, that constrain the project from achieving its objectives.

The reminder of this chapter reflects on the overall journey of this programme of study.

## 8.4 A reflection on the research journey

The works in this dissertation adopted the mixed method research (MMR) approach to enquiry, integrating quantitative (QUAN) and qualitative (QUAL) analyses to develop answers to the questions of the research. This approach views the understandings obtained from the QUAN and QUAL approaches as being complementary rather than alternative methods of scientific enquiry (Mingers, 2001; Mingers, 2003). The combination of QUAN and QUAL approaches to enquiry attempts to understand different sides of the same (Creswell & Plano Clark 2011; Biesta, 2010; Guba & Lincoln, 1994; Greene & Hall, 2010; Venkatesh et al. 2013). A research programme can seek different kinds of understanding; descriptive, interpretive, and/or theoretical understanding (Maxwell, 1992). The nature of the data available to the research which comprised project performance reports containing both numeric data (project metrics) and text (narratives of project participants' perspective) made is relevant to seek obtaining these types of understanding.

Descriptive understanding is concerned with accurately describing the activities observed in the studied domain. Activities are seen as *physical* and *behavioural events* rather than in terms of their meaning. These could be observed directly - through the human senses (seeing, hearing…etc.), or indirectly - inferred from data but which could in principle be observed. 'Accuracy' refers to not distorting the activities observed in the reporting of the account (Maxwell 1992, page: 285). This meant accepting that, while there was an attempt to distance the researcher from the researched phenomena, the knowledge claims made could not be guaranteed to be free from the influence of the participant's particular standpoint (Crotty, 2013; Chia, 2013) when analysing the schedule metrics in Chapter 5. For example, a key metric in the QUAN analysis was schedule delay where the research measured the number of days by which a project and its phases were completed late; whilst the researcher's task was solely to describe and apply the measurement on ABC data (i.e. without intervention), he is aware that the project schedule was constructed and influenced by the project manager.

Interpretive understanding is concerned with comprehending the studied phenomena from the perspective of the participants in the studied situation. Therefore, this understanding is based on the *mental construction* (concepts) of the people whose meaning is in question (Maxwell, 1992). As it is not possible to directly access the participant’s mind to obtain accounts (a description at a very low level of inference and abstraction) of their meaning, the researcher constructs the meaning based on the participant’s accounts and other evidence (Maxwell, 1992; Guba & Lincoln, 1994). This meant making meaning whilst the researcher was interacting with the researched phenomena (Crotty, 2013; Chia, 2013) when analysing the project participants’ statements in Chapter 7. For example, the Test phase’ performance report is the Test manager’s view/construction of the events during the Test phase execution, which other phase managers may disagree with; different people make different constructions of the same thing. Thus, the author of this thesis is aware that other perspectives are possible; for example, examining the performance reports of the Technical environment manager would have resulted in the emergence of a different set of categories of phenomena, since the Technical environment manager had responsibilities not for just one project, but several other ones simultaneously. Thus, the lateness of the Test phase from the Technical environment manager’s perspective may be just one of many, and considered less priority to, other challenges they were facing.

Theoretical understanding is concerned with obtaining a degree of abstraction from the physical and mental phenomena studied; it goes beyond the immediate description and interpretation of the phenomena and develops explanation of the studied domain; that is, a theory/explanation of some phenomenon (Maxwell, 1992). This meant while attempting to develop an explanation of schedule delay in Chapter 8, the researcher is aware that this understanding was based on the his application of explanatory constructs, and the connection among them, to the descriptive and interpretive understanding obtained beforehand, but which increases the degree of abstraction and make statements more generally applicable to a broader range of experiences/contexts of other researches.

The understandings obtained by a research programme can be examined from various aspects (Maxwell, 1992; Runeson & Höst, 2008; Yin, 2009; Onwuegbuzie & Johnson, 2006; Tashakkori & Teddlie, 1998; Guba & Lincoln, 1989, 2011). The following reflects on the findings of the research according to the types of understanding obtained: descriptive, interpretive, theoretical, and the generalizability of the findings to other contexts (Maxwell, 1992).

### 8.4.1 The descriptive results

This relates to obtaining ‘descriptive understanding’, and is concerned with accurately (i.e. without distortion) describing the activities and events (seen as physical phenomena) in the account, through using and applying appropriate terms; it can be examined through intersubjective agreement (Maxwell, 1992). Chapter 5 used schedule metrics on the numeric data of ABC project performance reports to describe schedule behaviour. The direct measurements (Kitchenham et al. 1995): scheduled start day, actual start day, scheduled finish day, actual finish day, and precedence relationships were collected retrospectively from ABC management reports complied at the time and which may not be complete or accurate. This may pose a threat to the validity of the direct measures. However, the project phase’s key dates, dependencies, and activity progress were monitored and scrutinised weekly for accuracy in the performance meetings in ABC. Theretofore, it is reasonable to attach a degree of confidence to the data.

The indirect measurements, derived from the direct measurements via equations (Kitchenham et al. 1995), were project schedule delay (PSD), phase schedule change (PSC), and phase schedule accuracy (PSA). It can be noted from Table 5.8 (section 5.3.3 - Chapter 5) that each metric measured an attribute of schedule (Procaccino & Verner, 2006); the terms used to define the metrics are relevant to what they were intended to measure, and the unit of measurement used for each metric was consistent with the direct measures it was derived from (Kitchenham et al. 1995). For example the PSD measured the ‘delay’ attribute of schedule, which was defined as the degree of mismatch between the scheduled duration of a project from its actual duration. The PSD metric was derived from the direct measures obtained from ABC progress reports. Thus the terms used to describe PSD is appropriate to how it was applied (i.e. measuring delay in schedule); and the indirect and direct measures used a consistent unit of measurement (number of days), which increases the validity of the PSD measure (Kitchenham et al. 1995). The PSD metric showed that the phases most contributing to schedule delay were the three Test phases: IT (Inc4) in Project 1; Integration Test - Plan & Preparation in Project 2; and DBT (Inc6) in Project 3. Although, it would be expected that the phases at the end of a project to be most subject to delay because they are affected by all the accumulated delays earlier in the project; it was seen that they introduced their own delay as well. These indicate points in the projects when something may be obstructing progress.

### 8.4.2 The interpretive findings

This relates to obtaining ‘interpretive understanding’, and is concerned with grounding the terms/meanings ascribed to participants’ statements (seen as mental phenomena) in the language and concepts of the people studied; it can be examined through member checking (Maxwell, 1992). Chapter 7 interpreted the textual data, the narrative of the project participants of ABC project performance reports, to categorise types of phenomena present during the execution of the Test phases and the way they influenced schedule duration. These accounts are the participants’ views at the time, not reconstructions by researcher; the research itself cannot have influenced outcomes. However, bias of participants at the time in structuring their narrative may pose threat to the interpretive validity. Similar to the numeric data, the participants’ statements were scrutinised during the performance meetings for its accurate representation of the events taking place at the time; and so should carry a degree of confidence.

Another threat to interpretive validity was the narrative in the textual reports being written in ABC terminology. In order to reduce this threat, triangulation of the contextual information about ABC projects (Chapter 3) and consultation with the project participants to interpret the terms in the textual reports and clarify points in the narrative schema that were not referenced, helped in further grounding the interpretations in the participants’ perception.

The use of Grounded theory techniques to analyse the textual information in this research was effective. Through its rigour, the GT enabled grounding the interpretive categories of phenomena into the perceptions of the project participants and revealed the interact interrelationships among the various project elements and how they influence one another and schedule duration. Nonetheless, a challenge was making sense of the various styles, advocated in the literature, of doing a GT study. In particular, the broad explanation offered in the literature on the practical applications of the theoretical sampling and theoretical saturation approaches. This challenge was overcome by consulting several authorities, and closely examining and re-examining the two approaches to clarify how they might be applied in practice.

### 8.4.3 The explanation development

This relates to obtaining ‘theoretical understanding’, and is concerned with developing theoretical constructions/explanation through increasing the degree of abstraction of the account from the physical or mental phenomena studied. Therefore, the concepts that the explanatory model applies to the phenomena and the relationships suggested among the concepts are examined for their appropriate characterisation of the phenomena (Maxwell, 1992). Chapter 8 increased the degree of abstraction of the GT categories of phenomena framing them within ANT concepts to develop explanatory model of the behaviour of schedule delay. A threat to validity may be in characterising a particular GT category in Table 8.1 (section 8.2) into an inappropriate ANT concept; for example, characterising ‘scope’ as ‘coordination’ since scope may not directly relate to reliance on someone or something for resources in order to carry out tasks. However, this threat was reduced through careful examination of how ANT concepts may be applied to software project management field as detailed in section 8.2.

Another threat to validity may be to use ANT concepts in incorrect ways; for example, using irreversibility as a concept of interaction (incorrect) instead of a feature of inscription (correct); because irreversibility indicates the degree of stability of an actor-network and its resistance to going back and changing the inscriptions that have already been circulated, rather than interaction among actors. Again, this threat was reduced through careful examination of the appropriate use of the ANT concepts in the literature (section 8.2).

The contribution of ANT to this study can be seen in its illumination of the constraining and empowering influences exerted by various actors (human and nonhuman) on achieving project objectives in maintaining the Test phase progress on schedule. Without the application of ANT concepts, it would have been difficult to identify the causes of schedule delay in such a complex project environment. ANT enabled investigating the interdependent areas of software project management through its demand on the researcher to attend to the context of the research object more carefully.

A similarity that can be observed between ANT and GT is that they both encourage the researcher to learn from the investigated domain and identify behaviours not immediately apparent, rather than impose preconceived ideas or existing frameworks on the domain. A difference that can be drawn is that, whilst GT categories emerge from the subject domain, ANT concepts (e.g. actors and intermediaries) are used to explain the subject domain. Thus, ANT and GT can be seen to complement one another; the GT analysis generating phenomena, while the ANT concepts illuminate the influence of interactions among them. Nonetheless, applying the ANT concepts to the GT categories was not straightforward; in particular, the limited explanation available in the literature on how concepts such as 'coordination' or 'irreversibility' ought to operate in practice. This challenge was overcome by careful examination and re-examination of the ANT concepts in light of various authorities who developed and added to the concepts over time. Although, applying ANT concepts to GT categories is not new - see for example Lopes (2010), further research producing innovative ways to bridge the two methods would benefit the field.

Finally, the overall methodological approach of this programme of study; where project metrics, grounded theory, and actor-network theory were integrated within an explanatory sequential mixed method design to make sense of the complexity of software project execution was effective. The implication is that, the research and practice bodies of knowledge need to match the complexity of a domain if they are to produce practical solutions to the challenges facing the area being investigated. Connecting the quantitative method to the qualitative method enabled project schedule delays (the quantitative results) to be explained through the more detailed analysis of selected cases using qualitative analysis; which could not have been possible using either of the methods alone; the mixed method approach offered a more complete understanding of the domain investigated. Furthermore, this research improved the clarity of the implementation and reporting from the explanatory sequential design by clearly distinguishing between the QUAN and QUAL phases of the research. The clarity was also improved by emergent nature of the approaches; since the design of the second phase (QUAL) was based on what was learned from the first phase (QUAN). Nonetheless, a challenge was the lengthy amount of time to implement to two phases.

### 8.4.4 Generalizability

Generalizability refers to the extent to which one can extend the account of a particular situation or population to other persons, times, or settings than those directly studied (Maxwell 1992, page: 293). Generalizability takes place through the development of a theory that makes sense of the particular context studied which may be useful in making sense of similar contexts (Maxwell 1992; Yin 2009, pages: 15 and 38; Verner et al. 2009, page: 319; Walsham 1995, page: 79; Gomm et al. 2000, page: 103; Mitchel 2000, page: 176; Stake, 2000; Donmoyer, 2000; Lincoln & Guba, 2000; Guba & Lincoln 1989, page: 243). That is, the 'claim is that those things not directly observed are similar to those described in the account; that the account can be generalized to some wider context' (Maxwell 1992, page: 294). Two aspects of generalizability - internal and external - are considered below.

#### 8.4.4.1 Internal generalizability

This is concerned with the applicability of the findings to cases inside ABC that were not directly observed by the study (Maxwell 1992). It is worth noting that the explanations of schedule delay developed in this thesis represent six cases across three projects; and thus, the findings can be seen as already generalised across the three projects, and can be extended to other projects within the programme which the three projects belong to because the context of the programme is the same.

However, ABC is a global company providing system development and integration services to varied public and private organisations, which their existing information technology (IT) environments and needs differ. Therefore, three aspects of system development and delivery need to be considered when examining the applicability of the findings of the research to projects, inside ABC but which fall, outside the studied programme.

*Software development method* - ABC employs customised development methods applicable to the client's IT environment and needs. For example, a particular client relationship may require using ABC's customised version of the Iterative and Incremental Development (IID) method; another client may require using ABC's customised version of the Agile practices. However, all client relationships that require using ABC's IID, throughout the world, use the same ABC's customised version of the method. Same applies to the Agile projects, and so on. A threat to internal generalizability is that the findings of this research may only be applicable to projects using ABC's IID. However, as the next section illustrates that the findings can be extended to projects using other development methods (e.g. Agile) in other organisations, which reduce the threat of extending the findings to ABC's non-IID projects.

*Onshore/offshore model* - ABC projects make extensive use of onshore/offshore distributed development to lower development cost and win contracts. However, some of ABC clients (e.g. the banking industry) are reluctant to share sensitive data outside the country of which the client is based. A threat to internal generalizability is that some of the research findings may not be applicable to the projects undertaken within a country. Nonetheless, the next section notes that the findings can be extended to projects executed within a country in other organisations, which reduces the threat of applying the same on ABC projects.

*Size of the system being developed* - a lot of ABC projects develop and integrate large software systems; with some projects undertaking medium size systems development. A threat to internal generalizability is the applicability of the findings to projects developing medium size systems in ABC. Yet, the next section shows that projects developing medium or small software from other organisations exhibit similar challenges to the ones identified in this research; therefore, the threat above can be reduced for similar projects within ABC.

#### 8.4.4.2 External generalizability

This is concerned with the applicability of the findings to cases outside ABC (Maxwell 1992). There appears to be few studies investigating schedule behaviour at phase level; existing work mostly examine behaviour at project level. This makes it difficult to compare the findings at phase level, but does mean this research fills a gap in the research literature. Furthermore, this study is one of only a few that have investigated the management of software projects developing enterprise architecture systems using globally distributed teams. Still, the following relates the empirical findings of this research to those of other researchers and practitioners.

*Applicability to the same industry sector*- the findings of this research can be useful to organisations operating within the same industry sector as ABC; i.e. global organisations that develop software through onshore/offshore model, and target public and private clients. For example, Rainer (1999)'s study on project schedule behaviour at IBM (see section 2.3.2 - Chapter 2) identified factors relating to dependency such as 'waiting on resources' and 'waiting on code defects' as contributing to schedule delay (which are compared to the 'constraining coordination' concept in this research); factors related to product such as ‘code’, ‘defect/fix’, and ‘system reliability problems’ (which are compared to the ‘intermediaries becoming mediators’ in this research). However, the findings of this thesis extends Rainer (1999)’s work as they are based on projects developing large systems with globally distributed teams, compared to Rainer’s investigation of projects developing small systems with collocated teams.

*Applicability to projects developing small or medium size systems* - the findings of this research was based on projects developing large systems through Iterative and Incremental Development and distributed globally. The findings can be useful to projects developing small and medium size software systems through Agile and in collocated teams. For example, Lehtinen et al. (2014)’s study (section 4.2 - Chapter 4) point to the dependency on other project participants for decisions, products, or resources to progress project work (i.e. ‘constraining coordination’ in this research); and project products causing problems (i.e. ‘intermediaries becoming mediators’). A key point about this research, however, is that it is examining projects in a very integrated enterprise architecture environment compared to Lehtinen and colleagues research.

*Applicability to projects developing large software systems* - the findings of this research can be applied to projects developing large systems where more specific/subsystems are embedded within more generic/larger system structures. Petersen et al. (2014)’s study aimed at identifying bottlenecks in developing very large-scale system of systems (SoS), at Ericsson AB in Sweden, is a case in point (see section 2.3.3 - Chapter 2). The findings are similar to the findings of this research. For example, factors relating to clarity and consistency of requirements between the SoS and system view in Petersen et al. (2014)’s study is compared with (‘constraining inscription’ and ‘reversibility’); factors relating to architecture dependencies and lack of available architecture knowledge is analogous to (‘constraining coordination’ and ‘intermediaries becoming mediators’); coordination related factors such as coordinating work activities and communication being impacted by multiple teams working on different systems in Petersen et al. (2014)’s study can be contrasted with the (‘large number of actors/actor-networks’); factors relating to the SoS and the system view having different/conflicting views of priority can be paralleled with (‘constraining coordination’) in this research. The works in this thesis extends Petersen et al. (2014)’s study in its examination of projects developing software in a globally distributed environment.

*Applicability to projects developing enterprise systems* - the findings of this research can be applied to projects developing systems within enterprise architecture environment. Hustad & Lange (2014)'s study (see section 2.4.4 - Chapter 2) investigating five SOA projects in Norway suggests comparable insights to the findings of this research in the factors influencing schedule delay. For example, a number of the findings of Hustad & Lange (2014)'s study can be contrasted with the ('constraining coordination') concept in this research, such as challenges with coordinating dependent activities among the projects, delays caused by these dependencies, insufficient communication, and dependency on the competency of the internal and external parties. Other findings in Hustad & Lange (2014)'s study such as the approach to running the four projects in parallel and constant change of shared services causing delays are contrasted with the ('constraining inscription'); the complexity of the project management effort was significantly underestimated with the large number of parties involved in the development effort can be compared with the ('large number of actor-networks' concept in this research).

*Applicability to projects developing software globally* - the findings of this research can be extended to projects developing software through globally distributed teams. For example, Herbsleb et al. (2001)'s findings (see section 2.3.4 - Chapter 2) that work distributed across sites take longer to complete ('constraining coordination') compared to same-site work, and that the factors influencing the delay being: the size of the change ('constraining inscription' and 'irreversibility'), the number of software components affected by the change ('intermediaries becoming mediators'), and the number of people involved in carrying out the change ('large number of actors'). This thesis extends Herbsleb et al. (2001)'s study in its focus on developing large software systems in an enterprise architecture environment.

*Applicability to non-project situations* - the findings of this research may be useful to non-project situations. For example, Lopes (2010)'s work (section 6.5.1 - Chapter 6) combining Grounded theory with Actor-network theory to explain the learning process during decision-making under uncertain and complex situations, generated comparable findings to the works in this thesis. The categories of phenomena and their interactions in Lopes (2010)'s PhD thesis such as 'decision-maker' is compared with ('actor') in this research; the 'support', 'systems', and 'uncertainty' compared with ('intermediary'); the 'context', 'uncertainty', and 'learning' with ('constraining coordination'). The two studies differ as our research investigated software engineering projects to understand controlling schedule duration, compared to Lopes (2010) which investigated human's to understand the learning process during decision-making; however, the two studies are similar as the situation examined in both studies occur within an uncertain and complex environment/context. Furthermore, it can be argued that the learning process during decision-making under uncertainty and complexity' could be applied to project control.

# 9 Conclusion

## 9.1 Introduction

This thesis has argued that there is a need for a better understanding of the project behaviours that influence software project progress. In particular, the interactions that emerge among project actors during project execution and the way they influence schedule duration, leading to possible schedule delay, should be studied. The researcher's experience has been that projects developing and integrating large software systems within an enterprise architecture environment and with globally distributed teams are particularly vulnerable to delays. The thesis has also argued that for this understanding to be useful research needs to draw upon empirical data, both the quantitative and qualitative, reflecting the many interdependent facets of such projects.

Having concluded the research effort, this final chapter formulates the answers to the research questions, conclusions from the research, and directions for further work.

## 9.2 Answering the research questions

The first question to which this research sought answers was:

RQ1 - To what extent do the mechanisms used to control schedule duration in projects developing and integrating large software systems within enterprise architecture environment through globally distributed teams identify the causes of schedule delay?

The mechanism used to control schedule duration in ABC included; the reporting of weekly performance of work activities by phase managers and meeting with project management to monitor progress, overlapping phases within an increment during project execution to maintain project end date, and the use of the parallel incremental development approach to deliver software in shorter time.

The weekly performance reports combined numeric data (locally-tailored SPI) with textual data (narratives) to inform management of what was going on. While the SPI could only tell that there was a delay, the textual reports explained why. The SPI provides unreliable information to schedule delay, because whilst project phases were completed later than scheduled, the SPI data at the end of the phase suggested that phases were finished on schedule (i.e. SPI value is 1).

The tendency for phases planned as (Finish-to-start) to be started when the phase it depended upon had not been completed - a tactic aimed at maintaining the project end date, created more dependences and rework for the dependent phase. We can see, particularly with Project 1, that the PM was able to recover lost time, only for it to be lost at the testing phase.

The research also suggests that the parallel nature of incremental development though improves staff utilisation, it also can have a slowing effect; for example, building code for the second increment can create resource clashes with the demands of remedial work as a result of the testing of the first increment being carried out at the same time. Thus, the parallel incremental development contributes to schedule delay rather than enabling identification of the causes of delay. We can see that the testing phase is the most difficult phase to control and the methods used for control appeared to be particularly weak.

Thus, the control mechanisms enabled identifying the project phases most contributing to project delay, rather than the specific causes of schedule delay. The latter, needed the support of the textual data.

RQ2 - In such an environment, what phenomena emerge during the execution of the project that influence schedule duration?

The research coded the textual data, in the project progress reports, into categories of phenomena present during the execution of the Test phases, the emergent categories were: dependency, people, product, schedule, and scope. For example, the category ‘schedule’ showed delay in completing activities and phases, a phase’s duration being extended or compressed, or phases starting later than planned. The category 'dependency' showed waiting on and delays to providing various products and services to the Test phase by project participants internal and external to ABC.

The developed narrative schema depicted the intricate relationships among the various phenomena revealing the factors influencing schedule duration; for example, defect in the Technical environment created defects in other products; such as the Enterprise service bus, which in turn influenced schedule duration. The large number of parties (14) that could be called upon to intervene at testing phase, and the large number of products (20) supporting the Test phase progress and their influence on one another illustrate the complexity inherent in executing the Test phase.

Thus, the phenomena that influenced schedule duration during the execution of the project were identified. However, this was limited to the perspective of the Test manager, rather than the underlying physical system, and it was a partial view; it needed the support of the contextual information and input from project participants. Although, the three sources of information enabled obtaining a more complete view of what was happening, the emergent interpretation was very local to ABC; hence, and a broader understanding was needed.

RQ3 - How do the interactions that develop among project actors during project execution influence schedule delay?

Studying the interaction among the project actors under an ANT lens enabled identifying the causes of schedule delay since it revealed the constraining influences exerted during such interactions on maintaining progress on schedule: constraining coordination, large number of actor-networks, intermediaries becoming mediators, constraining inscription, and reduced irreversibility.

The constraining coordination indicated that project rules were not widely accepted by the actor-networks due to competing priorities in maintaining their increment on schedule instead of supporting the one being tested, leading to delays in providing dependent products and services and leading to schedule delay. The large number of actor-networks revealed that the project was complex (due to the integrated enterprise architecture environment) and difficult to manage (because of globally distributed work activities). The intermediaries turning into mediators led to uncertainty in managing the project because they behaved differently to what was expected: various products and services used to support Test phase activities kept breaking down. Flawed inscriptions showed the programmes of action put in place among the actor-networks not producing the intended outcomes; various design and code defects emerging and the scope changing frequently during testing. The reversibility to previous phases created disruption in producing deliverables and destabilised progress; as project deliverables had to be reworked.

## 9.3 Conclusions from the research

The conclusions of the research can be grouped under three headings:

### 9.3.1 Underlying assumptions

The main managerial concern at project level in ABC was the delivery date, and project control was focused on this. The main metric used in project control was a locally-tailored SPI. Conventional SPI is calculated as Earned Value/Planned Value (EV/PV) where PV is the sum of the agreed estimates for the work which are scheduled to complete on the selected date, and EV is the sum of these values for the work that has actually been completed. ABC used a version of this based on counts of the key milestone events that have actually been achieved as opposed to those scheduled to be completed. The adoption of this ‘event-SPI’ was designed to quickly identify where obstacles to planned progress had appeared.

Another key practice in the management of projects at ABC was the adoption of a parallel incremental approach. The project was divided into increments. Activities were planned so that work on increments could be executed in parallel. For example, the design team finished the design for increment 1 and then started immediately on the design for increment 2, while the build team worked on coding increment 1.

The basic planning of activities within the parallel incremental approach was based on the assumption that, where possible, links between the incremental phases would be finish-to-start. A dependent activity (such as build) would only start when the necessary precursor (in this case, design) has been completed. From a quality control viewpoint this is the optimal approach. Another assumption was that, the only constraint on the execution of an activity was the completion of the activities upon which it was dependent.

### 9.3.2 Consequences of assumptions

The finish-to-start assumption, where one task is seen as requiring the completion of another previous one before it can start, in practice was a false one (see for example Figure 5.3 in section 5.3.3.3). Work was regularly started on the subsequent activity before the completion of the preceding dependent activity; for example, build work started before design had been completed. This would almost certainly mean rework when the final designs came through, it may also have been responsible for the number of defects coming through to testing.

For the parallel incremental approach to work properly, the resource requirements for all specialist activities have to be self-contained, that is that they should not be affected by demands for work outside that needed for the current task. In practice, software development staff could, for example, be working on building code for increment 2 of a project, but then be required to do remedial work as the consequence of the testing of increment 1. This caused a resource clash requiring the prioritising of build tasks.

This situation illustrates the complexity inherent in the development process. Schneberger & McLean (2003) noted that the complexity of a system is a function of the number and variety of components, the number and variety of their interactions, the number and variety of interdependencies, and the rate of change in the system; however, they also found that the ‘rate of change’ factor increases the complexity of the system greater than either the variety of components or their interaction factors. The rate of change (i.e. incrementing and iterating through development) in ABC projects was considerable (increments occurred in parallel), carried out by the same project teams working on multiple changes simultaneously. In addition, other change-related activities took place during the Test phase execution on all running increments, such as introducing a new product, scope changes, and requirements gaps. Thus, the parallel incremental approach increased the complexity of managing the project.

While the assumption that activities could be constrained by other activities upon which they were dependent turned out to be flexible, other constraints became apparent that had not been made explicit in project plans. There were several occasions where activities were constrained by the need to wait for services to be delivered which were supplied by other project actors. These appeared to be when either (i) specialist technical resources outside the project team were needed or (ii) assurance was needed that new functionality being developed by the team would not have a detrimental effect on existing implemented systems. The external parties involved could need to service requests from a range of different clients within the organisation, and once again there could be resource clashes, and a need for prioritisation at a higher level than the current project.

The number and variety of interdependencies, described above, increases the complexity of the project. Schneberger & McLean (2003) noted that interdependency among system components makes understanding and managing the system more difficult than a system with independent components. Furthermore, these interdependencies are more complex to manage when project actors are geographically distributed, compared to collocated projects, due to challenges in coordinating project tasks and communication among project teams at distance (Herbsleb & Moitra, 2001). Schneberger & McLean (2003) noted that distributed environments and enterprise systems are event driven, and when events occur in some pattern, complexity increases exponentially. The distributed nature of the project actors in two sites within the UK and between UK and India increased the complexity of interdependencies to manage project activities. In complex programme of work there are many organisational constraints on progress that may not be identified in a conventional plan.

It is interesting to see this in the light of the current promotion of Agile development practices. These advocate that development teams be self-contained as far as possible, so that external brakes on speed of code delivery are minimised (Stapleton, 1997). However, Petersen et al. (2014) point out that many contemporary systems can be seen as ‘systems of systems’ (SoS) where an individual software application is incorporated into a larger system to deliver business benefits. They noted that applications that are built as part of a SoS are particularly subject to development bottlenecks because of the needs to conform to externally imposed requirements. The ABC development environment fits into this characterisation.

### 9.3.3 Testing

The analysis of which activities seemed most prone to delay singled out testing as being the biggest contributor. As testing comes near the end of the development lifecycle it may of course be inheriting problems from earlier activities: a key factor in testing time is the number of errors found, which cannot be forecast, that need correction and re-testing. The management system in place assigned the test manager a key role in attempting to control the testing progress, but they did not have direct control of many actors whose contribution was essential to the successful completion of the project, such as the build and design teams; specialists responsible for the test environment, enterprise service bus, solution and technical architectures, suppliers of test data, the clients, peer suppliers, and other project teams.

The large number and variety of project actors, described above, increases the complexity of the project. A total of 49 different project elements (sub-categories of phenomena - Table A.1 in Appendix A) were in interaction during the Test phase execution, which illustrate the complexity of the execution effort. Schneberger & McLean (2003) found that that the variety of components in a computer system creates a more complex system for managers to understand and deal with; they also found that the variety has a greater effect than the number of components and their interactions. This research showed that, a total of 20 different products and 14 different people were in interaction during the Test phase execution.

The number and variety of interactions, among the project actors, increases the complexity of the project (Schneberger & McLean, 2003). Chapter 2 (section 2.4.1), extended Schneberger & McLean (2003)'s definition of system complexity to project complexity, in that: complexity of software project management can refer to, the interrelationship between the project elements in terms of: the number and variety of project elements (human and nonhuman), the number and variety of interactions among project elements (that exert constraining or empowering influences), the number and variety of interdependencies (that exert direct or indirect influences), and the rate of change of the project situation/context. This research showed that, there were 22 interactions at play among the project actors; and these interactions were of 3 different types: coordination, mediators, and inscription; added to this, the context of the project in having large number of actors and reversibility to previous phases (Figure 8.3 in Chapter 8), which illustrate the complexity being grappled with.

## 9.4 Implications for research and practice

### 9.4.1 Implications for practice

1 In order for project managers to maintain a project's progress on schedule, they ought to have control on all project resources; since the influences exerted by the project's context on the project's schedule duration cannot be underestimated (see section 8.3). In ABC's case, for example, the Technical environment manager (currently under the Technical environment management function) needs to be brought under the control of the Test manager, because almost all Test activities are dependent on the former to make progress; the Test manager can set priorities but without due impact on the technical aspects of the enterprise environment since more than one project may be going on at the same time.

2 In order to reduce the challenges of managing the development of large software systems, the use of globally distributed teams need to be minimised; since coordination of project activities becomes more difficult with distributed teams, which influences schedule delay and can increase the overall project cost (see section 2.3.4).

3 When employing the Iterative and incremental development model on a project, scheduling overlapping increments need to be avoided; since having the development team working on more than one increment simultaneously contributes to schedule delay due to competing priorities and resource clashes, which increases task's lead time (see section 2.3.2).

4 Project managers may need to assess the complexity of a project before execution commences, and where possible factor in this complexity in the schedule estimates. An indicator of project complexity can be a view of the interrelationship between the project elements in terms of, the number and variety of: project elements, expected interactions, anticipated interdependencies, and the expected rate of change of the project situation (see section 2.4.1).

5 When using the SPI (Schedule performance index) metric to track schedule performance, awareness of its limitations need to be exercised, since the SPI measure is misleading (see section 2.2.3). In general, regardless of the type of the measurement used to assess progress, project managers must remain vigilant to the progress of the critical tasks as these may be indistinguishable by the measurement used from the non-critical ones (see section 2.3.2).

6 Clear and specific rules of interaction between the project and other dependent projects or supporting functions need to be articulated prior to project execution; in order to commit these dependent teams to their role in providing timely support to enable maintaining the project's progress on schedule (see section 8.3).

7 Equal attention to the nonhuman elements of the project network need to be paid as is done to the human elements, since they may exert constraining influences on achieving project objectives that are no less hindering of project progress compared to human elements (see section 7.4).

### 9.4.2 Implications for research

1 The considerable influences exerted by external factors on the progress of a project suggests that the traditional perception of project boundary might need to be revisited to include elements surrounding the project because they are beyond the control of the project but still influence schedule delay; i.e. research to focus more on programmes rather than projects in isolation.

2 This research suggests that more attention needs to be paid to the interdependent areas of projects developing enterprise systems within globally distributed environments, rather than one or two discrete areas, because the interactions among these areas exert constraining influences on achieving project objectives.

3 Project execution need to be viewed as occurring in an interactive environment, involving complex networks of project elements (human and nonhuman) influencing one another and producing intermediary activities and products not accounted for during planning, rather than just the project team producing standard products on a specified timeframe.

## 9.5 Future work

A number of directions for further research can depart from the findings of this research. For example, future studies can examine the progress reports of phases other than testing of the ABC projects to understand how the interaction among the project actors influence schedule delay in these phases. In addition, further research into project rules that govern the interaction across teams/phases dependent on one another to progress work activities would be useful to identify ways in which coordination can be strengthened - both candidates for future possible work by the author of this thesis.

Other lines of enquiry may further investigate projects with similar characteristics to the ones examined here; i.e. projects that develop large software systems *and* within enterprise architecture environment *and* with globally distributed teams, but in a different organisation; in order to further support/extend the findings of the influencing factors on schedule delay emerged in this research.

Future research can look at the creation of simulation models of the interactions found in this research, and by rearranging the interdependencies and relationships of the actor-networks, the research can examine through different scenarios how the various interaction arrangements influence project outcome, and perhaps suggest ways to increase project success. This may help in addressing challenges facing management to resolve competing organisational and project priorities, physical resource clashes, and the escalation processes to resolve such problems.

Furthermore, the causes of schedule delay identified in this research can be used as knowledge base to intelligent systems like Case-based reasoning (CBR). A future study can implement a CBR system based on the six cases (Test phases) investigated in this research: then add new cases of other projects from other organisations over time to expand the knowledge base. Armed with more cases, the CBR system can further be used to inform novice project managers of potential causes of schedule delay before they start their project, by comparing the context of their projects with the knowledge base.

To conclude then, it would seem that our understanding of controlling schedule duration and the way it influences schedule delay in software projects is still developing, and further work is required in order to unpack such a complex undertaking; Checkland (1981, page: xii) put it succinctly:

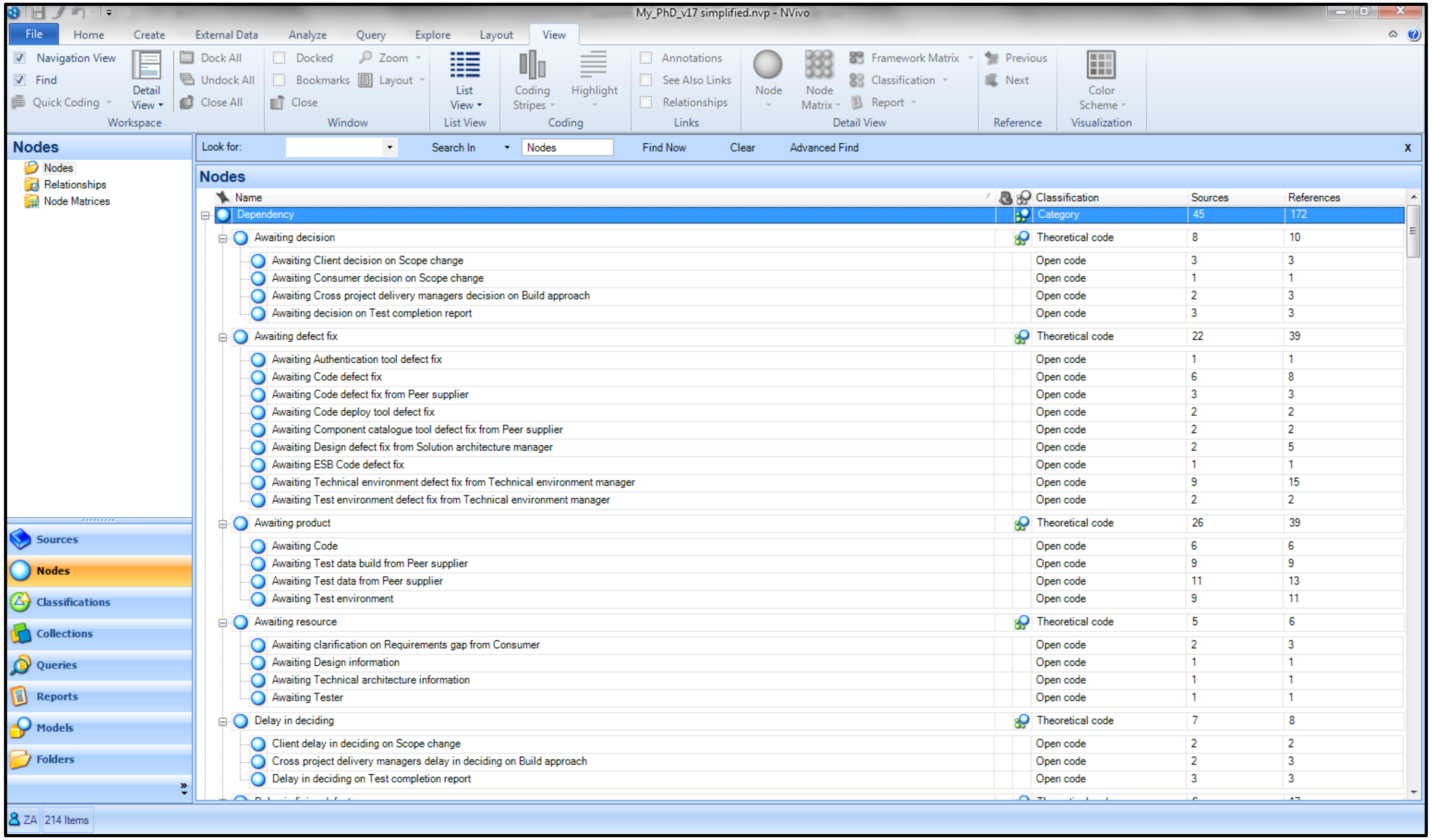
> '*Obviously the work is not finished, and can never be finished. There are no absolute positions to be reached in the attempt by men to understand the world in which they find themselves: new experience may in the future refute present conjectures. So the work itself must be regarded as an on-going system of a particular kind: A learning system which will continue to develop ideas, to test them out in practice, and to learn from the experience gained*'.

# Appendix A: Qualitative approach data

This appendix is associated with Chapter 7 (Qualitative approach) and provides supporting information on the Grounded theory analysis carried out in the QUAL approach, it includes: code hierarchy structure, relationship among the sub-categories, the textual data of five cases (source data of this research), and sample analytic memos.

## A1. Code hierarchy structure

Figure A.1 shows screenshot example of the code hierarchy structure in the qualitative data management software NVivo, followed by Table A.1 showing the full list of emergent codes from analysing the textual data of the performance reports of the six cases. The code classification and hierarchy follows the description presented in section 7.3.2 - Chapter 7: i.e. Category\Selective or Theoretical code\Open code; the hierarchy is indicated through indentation of the various codes under the column ‘Name’ in Table A.1. Where the name of the code is indicative of its meaning, no description is provided.



Nodes

| Name | Classification | Sources | References |
|---|---|---|---|
| Dependency | Category | 45 | 172 |
| Awaiting decision | Theoretical code | 8 | 10 |
| Awaiting Client decision on Scope change | Open code | 3 | 3 |
| Awaiting Consumer decision on Scope change | Open code | 1 | 1 |
| Awaiting Cross project delivery managers decision on Build approach | Open code | 2 | 3 |
| Awaiting decision on Test completion report | Open code | 3 | 3 |
| Awaiting defect fix | Theoretical code | 22 | 39 |
| Awaiting Authentication tool defect fix | Open code | 1 | 1 |
| Awaiting Code defect fix | Open code | 6 | 8 |
| Awaiting Code defect fix from Peer supplier | Open code | 3 | 3 |
| Awaiting Code deploy tool defect fix | Open code | 2 | 2 |
| Awaiting Component catalogue tool defect fix from Peer supplier | Open code | 2 | 2 |
| Awaiting Design defect fix from Solution architecture manager | Open code | 2 | 5 |
| Awaiting ESB Code defect fix | Open code | 1 | 1 |
| Awaiting Technical environment defect fix from Technical environment manager | Open code | 9 | 15 |
| Awaiting Test environment defect fix from Technical environment manager | Open code | 2 | 2 |
| Awaiting product | Theoretical code | 26 | 39 |
| Awaiting Code | Open code | 6 | 6 |
| Awaiting Test data build from Peer supplier | Open code | 9 | 9 |
| Awaiting Test data from Peer supplier | Open code | 11 | 13 |
| Awaiting Test environment | Open code | 9 | 11 |
| Awaiting resource | Theoretical code | 5 | 6 |
| Awaiting clarification on Requirements gap from Consumer | Open code | 2 | 3 |
| Awaiting Design information | Open code | 1 | 1 |
| Awaiting Technical architecture information | Open code | 1 | 1 |
| Awaiting Tester | Open code | 1 | 1 |
| Delay in deciding | Theoretical code | 7 | 8 |
| Client delay in deciding on Scope change | Open code | 2 | 2 |
| Cross project delivery managers delay in deciding on Build approach | Open code | 2 | 3 |
| Delay in deciding on Test completion report | Open code | 3 | 3 |

**Figure A.1 Code hierarchy structure in NVivo**

| **Name** | **Classification** | **Description** |
|---|---|---|
| Dependency | Category | Test manager was dependent on other project participants, for provision of products and/or services, in order to maintain progress of the Test phase on schedule |
| Awaiting decision | Theoretical code | Test manager awaited decision from other project participants. Decision refers to sign-off, agreement, or approval. Where ‘Name’ does not specify the project participant, it means that the textual data did not identify the project participant |
| Awaiting Client decision on Scope change | Open code | |
| Awaiting Consumer decision on Scope change | Open code | |
| Awaiting Cross project delivery managers decision on Build approach | Open code | |
| Awaiting decision on Test completion report | Open code | |
| Awaiting defect fix | Theoretical code | Test manager awaited defect fix from other project participants |
| Awaiting Authentication tool defect fix | Open code | |

| Name | Classification | Description |
|---|---|---|
| Awaiting Code defect fix | Open code | |
| Awaiting Code defect fix from Peer supplier | Open code | |
| Awaiting Code deploy tool defect fix | Open code | |
| Awaiting Component catalogue tool defect fix from Peer supplier | Open code | |
| Awaiting Design defect fix from Solution architecture manager | Open code | |
| Awaiting ESB Code defect fix | Open code | |
| Awaiting Technical environment defect fix from Technical environment manager | Open code | |
| Awaiting Test environment defect fix from Technical environment manager | Open code | |
| Awaiting product | Theoretical code | Test manager awaited product from other project participants. Product refers to the items under the category ‘Product’ - see further down |
| Awaiting Code | Open code | |
| Awaiting Test data build from Peer supplier | Open code | |
| Awaiting Test data from Peer supplier | Open code | |
| Awaiting Test environment | Open code | |
| Awaiting resource | Theoretical code | Test manager awaited resource from their team or other project participants. |

| **Name** | **Classification** | **Description** |
|---|---|---|
| | | Resource refers to information, clarification, or people (see the items under category 'People' further down) |
| Awaiting clarification on Requirements gap from Consumer | Open code | |
| Awaiting Design information | Open code | |
| Awaiting Technical architecture information | Open code | |
| Awaiting Tester | Open code | |
| Delay in deciding | Theoretical code | Project participant delayed making decision needed by Test manager |
| Client delay in deciding on Scope change | Open code | |
| Cross project delivery managers delay in deciding on Build approach | Open code | |
| Delay in deciding on Test completion report | Open code | |
| Delay in fixing defect | Theoretical code | Project participant delayed fixing defect needed by Test manager |
| Delay in fixing Design defect from Solution architecture manager | Open code | |
| Delay in fixing Peer supplier Code defect | Open code | |
| Delay in fixing Technical environment defect from Technical environment manager | Open code | |
| Delay in providing product | Theoretical code | Project participant delayed providing |

| Name | Classification | Description |
|---|---|---|
| | | product needed by Test manager |
| Delay in providing Code | Open code | |
| Delay in providing Code due to Component Catalogue Tool defect | Open code | |
| Delay in providing Code due to Technical environment defect | Open code | |
| Delay in providing Code to Consumer due to Build approach | Open code | |
| Delay in providing Code to Consumer due to Code defect | Open code | |
| Delay in providing Code to Consumer due to Regression Test tool defect | Open code | |
| Delay in providing Code to Consumer due to Technical environment defect | Open code | |
| Delay in providing Code to Consumer due to Tester on vacation | Open code | |
| Delay in providing Peer supplier Test data | Open code | |
| Delay in providing Peer supplier Test data build | Open code | |
| Delay in providing Test data build due to Component Catalogue Tool defect | Open code | |
| Delay in providing Test environment | Open code | |
| Delay in providing Test environment due to Code deploy tool defect | Open code | |
| Delay in providing Test environment due to ESB Code defect | Open code | |
| Delay in providing Test environment due to Technical environment defect | Open code | |
| Delay in providing resource | Theoretical code | Project participant delayed providing resource needed by Test manager |

| Name | Classification | Description |
|---|---|---|
| Delay in providing clarification on Requirements gap from Consumer | Open code | |
| Delay in providing Design information | Open code | |
| Delay in providing Technical architecture information | Open code | |
| People | Category | Human project participants who to varying degree contribute to the Test phase progress |
| Build manager | Selective code | As defined in Table 3.3 - section 3.2 - Chapter 3 |
| Build manager | Open code | |
| Client | Selective code | As defined in Table 3.6 - section 3.3 - Chapter 3 |
| Client | Open code | |
| Consumer | Selective code | As defined in Table 3.6 - section 3.3 - Chapter 3 |
| Consumer | Open code | |
| Cross project delivery managers | Selective code | Project participants that developed/updated parts of the enterprise system, but which are not investigated by this research |

| Name | Classification | Description |
|---|---|---|
| Cross project delivery managers | Open code | |
| Peer supplier | Selective code | As defined in Table 3.6 - section 3.3 - Chapter 3 |
| Peer Supplier | Open code | |
| Solution architecture manager | Selective code | As defined in Table 3.6 - section 3.3 - Chapter 3 |
| Solution architecture manager | Open code | |
| Technical environment manager | Selective code | As defined in Table 3.6 - section 3.3 - Chapter 3 |
| Technical environment manager | Open code | |
| Test data manager | Selective code | As defined in Table 3.6 - section 3.3 - Chapter 3 |
| Test data manager | Open code | |
| Test manager | Selective code | As defined in Table 3.3 - section 3.2 - Chapter 3 |
| Test manager | Open code | |
| Tester | Selective code | Member of the Test team; works for the Test manager |

| Name | Classification | Description |
|---|---|---|
| Tester | Open code | |
| Product | Category | Artefacts produced by project participants for use by other project participants which enable further producing project deliverables |
| Application server | Theoretical code | Software/ hardware artefact that serves requests made from client computer machines. For example, a personal computer downloads an article from the internet (a server computer) |
| Application server | Open code | |
| Authentication tool | Theoretical code | As defined in Table 3.1 - section 3.2.1 - Chapter 3 |
| Authentication tool | Open code | |
| Build approach | Theoretical code | Document that outlines the approach to develop software application/component |
| Build approach | Open code | |
| Code | Theoretical code | As defined in Table 3.7 - section 3.3 - |

| **Name** | **Classification** | **Description** |
|---|---|---|
| | | Chapter 3 |
| Code | Open code | |
| Code deploy tool | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Code deploy tool | Open code | |
| Component catalogue tool | Theoretical code | As defined in Table 3.1 - section 3.2.1 - Chapter 3 |
| Component catalogue tool | Open code | |
| Defect | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Authentication tool defect | Open code | |
| Code defect | Open code | |
| Code deploy tool defect | Open code | |
| Component catalogue tool defect | Open code | |
| Design defect | Open code | |
| Design defect due to Requirements gap | Open code | |
| ESB Code defect | Open code | |
| ESB Code defect due to Application server upgrade | Open code | |

| **Name** | **Classification** | **Description** |
|---|---|---|
| ESB Code defect due to Technical environment defect | Open code | |
| Regression test tool defect | Open code | |
| Regression Test tool defect due to Technical environment defect | Open code | |
| Technical environment defect | Open code | |
| Test environment defect | Open code | |
| Enterprise service bus | Theoretical code | As defined in Table 3.1 - section 3.2.1 - Chapter 3 |
| Enterprise service bus | Open code | |
| Introducing new product | Theoretical code | Integrating or upgrading new product into the enterprise system during Test phase execution |
| New product | Open code | |
| Performance report | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Performance report | Open code | |
| Regression Test tool | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Regression Test tool | Open code | |

| Name | Classification | Description |
|---|---|---|
| Scope of deliverables | Theoretical code | List of the artefacts required to be designed, developed, and tested |
| Component Inventory | Open code | List of the software components required to be modified or newly developed |
| Technical environment | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Technical environment | Open code | |
| Test completion report | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Test completion report | Open code | |
| Test data | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Test data | Open code | |
| Test data build | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Test data build | Open code | |
| Test data requirement | Theoretical code | Specification of the type/format of the Test data required to be available on |

| Name | Classification | Description |
|---|---|---|
| | | the Test environment ready for performing Test execution |
| Test data requirement | Open code | |
| Test environment | Theoretical code | As defined in Table 3.7 - section 3.3 - Chapter 3 |
| Test environment | Open code | |
| Use Case | Theoretical code | A scenario of how the future software will be used by the user to carry out business functions |
| Use Case | Open code | |
| Wireframe | Theoretical code | A page schematic or screen blueprint/visual guide representing the skeletal framework of online application software; they are created for the purpose of arranging elements to best accomplish a particular purpose during the design stage of the development lifecycle |
| Wireframe | Open code | |

| **Name** | **Classification** | **Description** |
|---|---|---|
| Schedule | Category | The means by which project/phase activities were ordered on a timeline showing what activity was planned to be carried out, by when, and in what order; for example, Gantt chart |
| Activity delay | Selective code | As defined in section A4.2 |
| Activity delay due to Authentication tool defect | Open code | |
| Activity delay due to Code defect | Open code | |
| Activity delay due to Code deploy tool defect | Open code | |
| Activity delay due to Component catalogue tool defect | Open code | |
| Activity delay due to delay in fixing Design defect from Solution architecture manager | Open code | |
| Activity delay due to delay in fixing Peer supplier Code defect | Open code | |
| Activity delay due to delay in fixing Technical environment defect | Open code | |
| Activity delay due to delay in providing clarification on Requirements gap | Open code | |
| Activity delay due to delay in providing Code | Open code | |
| Activity delay due to delay in providing Technical architecture information | Open code | |
| Activity delay due to delay in providing Test environment | Open code | |
| Activity delay due to Design defect | Open code | |

| **Name** | **Classification** | **Description** |
|---|---|---|
| Activity delay due to ESB Code defect | Open code | |
| Activity delay due to introducing new product | Open code | |
| Activity delay due to lack of Design knowledge | Open code | |
| Activity delay due to Technical environment defect | Open code | |
| Activity delay due to Tester on vacation | Open code | |
| Duration compression | Selective code | As defined in section A4.4 |
| Duration compression due to delay in providing Test environment | Open code | |
| Duration extension | Selective code | As defined in section A4.5 |
| Duration extension due to Authentication tool defect | Open code | |
| Duration extension due to delay in providing Peer supplier Test data | Open code | |
| Duration extension due to Technical environment defect | Open code | |
| Lacking control over progress | Theoretical code | Test manager conveying lacking control over Test phase progress due to factors outside their sphere of influence and which affect maintaining Test phase progress on schedule |
| Lacking control over progress due to Authentication tool defect | Open code | |

| Name | Classification | Description |
|---|---|---|
| Lacking control over progress due to Code defect | Open code | |
| Lacking control over progress due to delay in approving Test completion report | Open code | |
| Lacking control over progress due to delay in deciding on Build approach | Open code | |
| Lacking control over progress due to delay in fixing Peer supplier Code defect | Open code | |
| Lacking control over progress due to delay in providing Peer supplier Test data | Open code | |
| Lacking control over progress due to delay in providing Peer supplier Test data build | Open code | |
| Lacking control over progress due to delay in providing Technical architecture information | Open code | |
| Lacking control over progress due to delay in providing Test environment | Open code | |
| Lacking control over progress due to Design defect | Open code | |
| Lacking control over progress due to lack of Design knowledge | Open code | |
| Lacking control over progress due to Technical environment defect | Open code | |
| No schedule baseline | Selective code | Execution of Test phase schedule without being approved prior to start of execution |
| Unapproved schedule | Open code | |
| Unapproved schedule due to Scope change | Open code | |
| Nonworking time | Theoretical code | Vacation time for Tester |

| Name | Classification | Description |
|---|---|---|
| Nonworking time | Open code | |
| Phase delay | Selective code | As defined in section A4.3 |
| Phase delay due to Authentication tool defect | Open code | |
| Phase delay due to Code defect | Open code | |
| Phase delay due to delay in Client deciding Scope change | Open code | |
| Phase delay due to delay in deciding on Build approach | Open code | |
| Phase delay due to delay in deciding on Scope change | Open code | |
| Phase delay due to delay in deciding on Test completion report | Open code | |
| Phase delay due to delay in fixing Technical environment defect | Open code | |
| Phase delay due to delay in providing clarification on Requirements gap | Open code | |
| Phase delay due to delay in providing Code | Open code | |
| Phase delay due to delay in providing Peer supplier Test data | Open code | |
| Phase delay due to Design defect | Open code | |
| Phase delay due to Technical environment defect | Open code | |
| Reporting progress status | Theoretical code | Test manager reporting the status of the progress of the Test phase; e.g. affirming activity finish - see below Open codes |

| **Name** | **Classification** | **Description** |
|---|---|---|
| Affirming activity finish | Open code | |
| Affirming activity in-progress | Open code | |
| Affirming activity start | Open code | |
| Affirming phase finish | Open code | |
| Affirming phase start | Open code | |
| Explaining missing schedule target | Open code | |
| Indicating maintaining progress | Open code | |
| Quantifying progress | Open code | Test manager quantified the progress of activities, such as: 3 out of 10 components tested |
| Setting schedule target | Open code | |
| Voicing concerns about resource contention | Open code | |
| Start variance | Selective code | Test phase execution started later than scheduled |
| Started late | Open code | |
| Started late due to delay in providing Code | Open code | |
| Started late due to delay in providing Test environment | Open code | |
| Started late due to Technical environment defect | Open code | |

| **Name** | **Classification** | **Description** |
|---|---|---|
| Scope | Category | The total number of products and services agreed to be delivered as part of the project or within a phase in a particular point in time; scope can change during project execution due to change requests |
| Requirements gap | Theoretical code | Unclear functional requirements discovered during Test phase execution |
| Requirements gap due to provision of Use Cases instead of Functional Designs | Open code | |
| Requirements gap due to Solution architecture manager providing Wireframe | Open code | |
| Requirements gap from Consumer | Open code | |
| Scope change | Theoretical code | Modification of scope of work, during project execution, which was agreed prior to project execution |
| Consumer Change request | Open code | |
| Open scope due to Solution architecture manager providing unconfirmed Scope of deliverables | Open code | Unapproved scope of deliverables |
| Scope cut | Open code | |
| Scope cut due to Client Change request | Open code | |

| Name | Classification | Description |
|---|---|---|
| Scope increase due to deferred Code defects from previous Test phase | Open code | |
| Scope move | Open code | Specific functionality was taken out from the current increment and moved to a future increment during Test phase execution |
| Scope move due to Peer supplier Code defect | Open code | |

**Table A.1 Code hierarchy structure**

## A2. Relationships among the sub-categories

Table A.2 shows the relationships (i.e. theoretical codes in Table A.1) among the elements (i.e. the selective codes in Table A.1) across the six cases.

| From Name (Selective code) | Relationship type (Theoretical code) | To Name (Selective code) |
|---|---|---|
| People\Test manager | Awaiting decision | People\Client |
| People\Test manager | Awaiting decision | No reference to recipient |
| People\Test manager | Awaiting decision | People\Cross project delivery managers |
| People\Test manager | Awaiting decision | People\Consumer |
| People\Test manager | Awaiting defect fix | People\Build manager |
| People\Test manager | Awaiting defect fix | No reference to recipient |
| People\Test manager | Awaiting defect fix | People\Technical environment manager |

| **From Name (Selective code)** | **Relationship type (Theoretical code)** | **To Name (Selective code)** |
|---|---|---|
| People\Test manager | Awaiting defect fix | People\Solution architecture manager |
| People\Test manager | Awaiting defect fix | People\Peer supplier |
| People\Test manager | Awaiting product | People\Build manager |
| People\Test manager | Awaiting product | No reference to recipient |
| People\Test manager | Awaiting product | People\Peer supplier |
| People\Test manager | Awaiting resource | No reference to recipient |
| People\Test manager | Awaiting resource | People\Tester |
| People\Test manager | Awaiting resource | People\Consumer |
| People\Test manager | Build approach | People\Cross project delivery managers |
| People\Build manager | Code | People\Test manager |
| People\Build manager | Defect | Schedule\Activity delay |
| People\Build manager | Defect | Schedule\Phase delay |
| Product\Component catalogue tool | Defect | Dependency\Delay in providing product |
| Product\Authentication tool | Defect | Schedule\Activity delay |
| Product\Authentication tool | Defect | Schedule\Duration extension |
| Product\Authentication tool | Defect | Schedule\Phase delay |
| Product\Code deploy tool | Defect | Schedule\Activity delay |
| Product\Enterprise service bus | Defect | Schedule\Activity delay |
| Product\Technical environment | Defect | Schedule\Activity delay |
| Product\Technical environment | Defect | Schedule\Phase delay |
| Product\Technical environment | Defect | Schedule\Duration extension |
| Product\Test environment | Defect | No reference to recipient |
| No reference to recipient | Defect | Schedule\Activity delay |
| No reference to recipient | Defect | Schedule\Phase delay |

| From Name (Selective code) | Relationship type (Theoretical code) | To Name (Selective code) |
|---|---|---|
| People\Peer supplier | Defect | Schedule\Activity delay |
| People\Peer supplier | Defect | Scope\Scope change |
| Product\Code | Defect | Scope\Scope change |
| No reference to recipient | Defect | Schedule\Start variance |
| Product\Technical environment | Defect | Product\Enterprise service bus |
| Product\Component catalogue tool | Defect | Schedule\Activity delay |
| Product\Technical environment | Defect | Product\Regression Test tool |
| Product\Regression Test tool | Defect | No reference to recipient |
| Product\Technical environment | Defect | Dependency\Delay in providing product |
| Product\Code | Defect | Dependency\Delay in providing product |
| Product\Regression Test tool | Defect | Dependency\Delay in providing product |
| Product\Code deploy tool | Defect | Dependency\Delay in providing product |
| Product\Enterprise service bus | Defect | Dependency\Delay in providing product |
| Scope\Requirements gap | Defect | Schedule\Activity delay |
| Product\Technical environment | Defect | Schedule\Start variance |
| People\Client | Delay in deciding | Schedule\Phase delay |
| No reference to recipient | Delay in deciding | Schedule\Phase delay |
| People\Cross project delivery managers | Delay in deciding | Schedule\Phase delay |
| People\Technical environment manager | Delay in fixing defect | Schedule\Activity delay |
| People\Technical environment manager | Delay in fixing defect | Schedule\Phase delay |
| People\Solution architecture manager | Delay in fixing defect | Schedule\Activity delay |
| People\Peer supplier | Delay in fixing defect | Schedule\Activity delay |
| People\Build manager | Delay in providing product | Schedule\Activity delay |
| People\Build manager | Delay in providing product | Schedule\Phase delay |

| From Name (Selective code) | Relationship type (Theoretical code) | To Name (Selective code) |
|---|---|---|
| No reference to recipient | Delay in providing product | Schedule\Activity delay |
| No reference to recipient | Delay in providing product | Schedule\Duration compression |
| People\Test manager | Delay in providing product | People\Consumer |
| People\Peer supplier | Delay in providing product | Schedule\Phase delay |
| People\Peer supplier | Delay in providing product | Schedule\Duration extension |
| People\Peer supplier | Delay in providing product | No reference to recipient |
| No reference to recipient | Delay in providing product | Schedule\Start variance |
| Schedule\Nonworking time | Delay in providing product | People\Consumer |
| No reference to recipient | Delay in providing resource | Schedule\Activity delay |
| People\Consumer | Delay in providing resource | Schedule\Activity delay |
| People\Consumer | Delay in providing resource | Schedule\Phase delay |
| Product\Application server | Enterprise service bus | Product\Defect |
| No reference to recipient | Introducing new product | Schedule\Activity delay |
| People\Test manager | Lacking control over progress | No reference to recipient |
| People\Tester | Nonworking time | Schedule\Activity delay |
| People\Tester | Nonworking time | Dependency\Delay in providing product |
| People\Test manager | Performance report | No reference to recipient |
| People\Test manager | Reporting progress status | No reference to recipient |
| People\Consumer | Requirements gap | Product\Defect |
| People\Client | Scope change | No reference to recipient |
| People\Consumer | Scope change | No reference to recipient |
| People\Solution architecture manager | Scope change | Schedule\No schedule baseline |
| People\Solution architecture manager | Scope of deliverables | People\Test manager |
| People\Test manager | Test completion report | No reference to recipient |

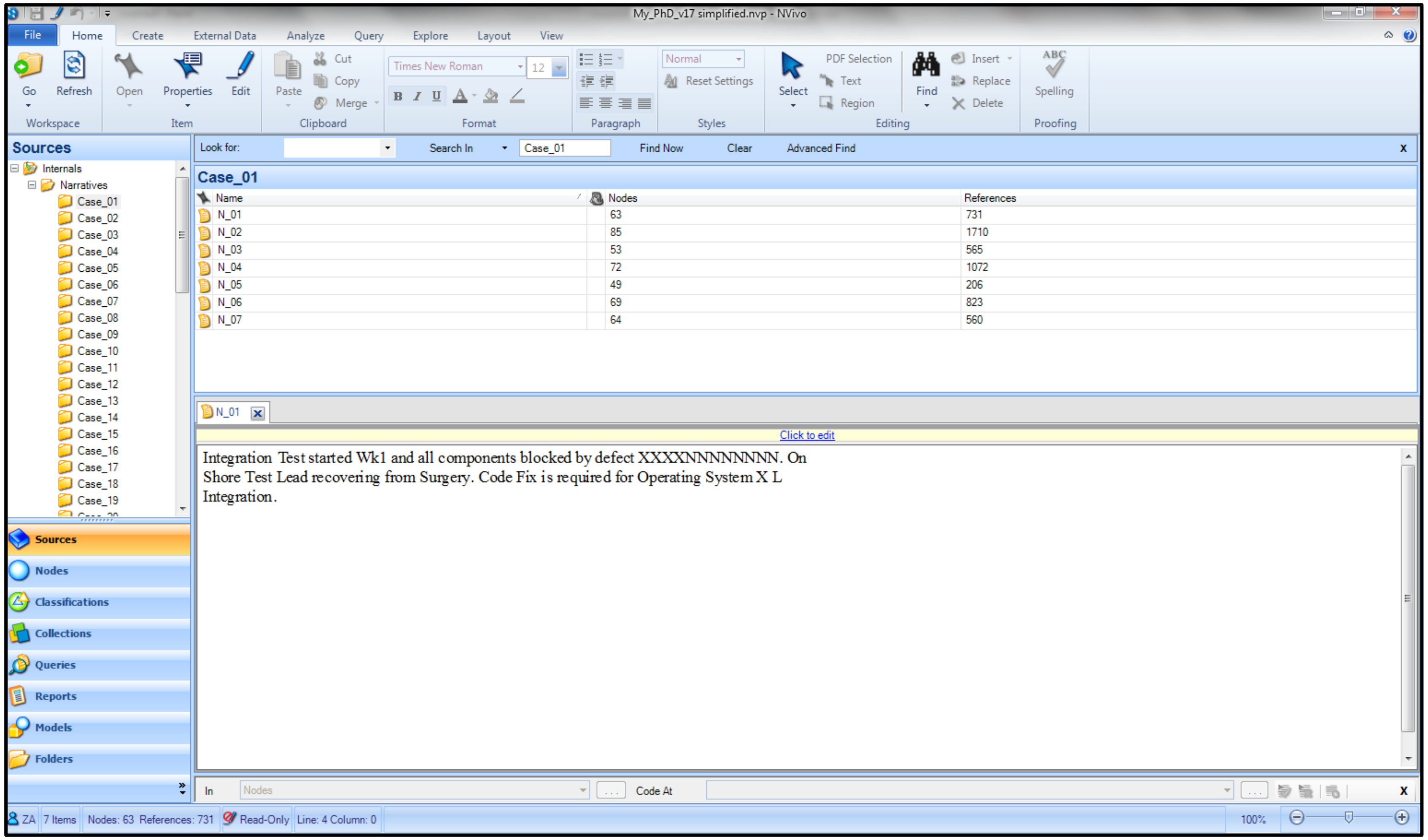
| From Name (Selective code) | Relationship type (Theoretical code) | To Name (Selective code) |
|---|---|---|
| People\Peer supplier | Test data | People\Test manager |
| People\Peer supplier | Test data build | People\Test manager |
| People\Test manager | Test data requirement | People\Test data manager |
| No reference to recipient | Use Case | People\Test manager |
| No reference to recipient | Use Case | Scope\Requirements gap |
| People\Solution architecture manager | Wireframe | Scope\Requirements gap |

**Table A.2 Relationships among the sub-categories**

## A3. Textual information of the cases

This section presents the textual information (source data) of the Test phase performance reports, an extract from NVivo (the qualitative data management software). Entries between square brackets [researcher entry] indicate an entry made by the researcher; for example, [No performance report] indicates that progress report does not exist for the particular week.

The textual information for P1-IT (Inc4) was presented in Table 7.2 (section 7.3.1 - Chapter 7); the textual information of the remaining five cases is presented in Tables A.3 through to A.7 following a screenshot example of the data in NVivo (Figure A.2).



Case_01

| Name | Nodes | References |
|---|---|---|
| N_01 | 63 | 731 |
| N_02 | 85 | 1710 |
| N_03 | 53 | 565 |
| N_04 | 72 | 1072 |
| N_05 | 49 | 206 |
| N_06 | 69 | 823 |
| N_07 | 64 | 560 |

Integration Test started Wk1 and all components blocked by defect XXXXNNNNNNNN. On Shore Test Lead recovering from Surgery. Code Fix is required for Operating System X L Integration.

**Figure A.2 Textual information in NVivo**

| Narrative # | Textual information |
|---|---|
| N_01 | Integration Test started Wk1 and all components blocked by defect XXXXNNNNNNNN. On Shore Test Lead recovering from Surgery. Code Fix is required for Operating System X L Integration. |
| N_02 | Integration Test started Wk1 and making good progress, but faces a very aggressive schedule to complete by the planned completion date (Wk4). Onshore Test Lead recovering from surgery; mitigation strategy in place. Late start to execution due to delays in Integration Test environment readiness. Aiming to absorb the 1 week lost, by delivering over a 3 week execution plan (rather than 4 weeks). No points claimed on Wk2 due to configuration issues in introducing the new Authentication Tool. Agreed with Client Test Management to track points for the Authentication Tool separately from next reporting period onwards to clarify the position, as many of the components are now running clean without the Authentication Tool. |
| N_03 | Confirmed at Consumer Supplier Cross Management Meeting (Wk3) that the Authentication Tool part can be delivered on Wk5 (1 week extension) without impact to Consumer. Therefore reporting separate EV/PV for with and without Authentication Tool. Require re-plan for extension to Wk5. |
| N_04 | 10 of 17 components clean. Pass through Authentication Tool components execution due this week; however XXX environment not available – completion at risk. Severe downtime incurred on most days due to environment and code/deploy issues. Test Lead replacement identified – joining Wk5 team. |
| N_05 | 17 of 17 components clean with the Authentication Tool. 7 of 10 Inc3 scenarios clean (incl. Authentication Tool). Downtime incurred due to environment and code/deploy issues has been reduced this week. |
| N_06 | 3 component tests and 2 Authentication Tool only scenarios outstanding. Outstanding defects are held on a variety of Third-party and architecture queries. |
| N_07 | Integration Test complete for R1 Inc3:1 outstanding defect with Third-party with no ETC, agreed by all stakeholders this can be carried |

| Narrative # | Textual information |
|---|---|
| | over to Inc4 (Defect NNNNN – components X and Y issue, technical error defect in analysis with Third-party as possibly L issue.) |

**Table A.3 Case P1-IT-Ex-Au (Inc1) Textual information**

| Narrative # | Textual information |
|---|---|
| N_01 | 10/17 components are running clean without Authentication Tool, but recent environment downtime has threatened the target to complete all by Wk2. Now expecting a small delay into next week. Integration Test Environment received 1 week late due to issues in connection with new ESB code to support XXXN upgrade. |
| N_02 | 14 of 17 components clean. Pass through Authentication Tool components execution due this week; however XXX environment not available – completion at risk. Severe downtime incurred on most days due to environment and code/deploy issues. Test Lead replacement identified – joining Wk3 team. |
| N_03 | Outstanding points are on Inc3 scenarios, functional knowledge needed to create input message. |

**Table A.4 Case P1-IT-Ex-no-Au (Inc1) Textual information**

| Narrative # | Textual information |
|---|---|
| N_01 | Integration Test data requirements have been submitted to Test Data team. E/J data requirements have been submitted to Third party. Request X creation has started 2 weeks ahead of plan and is 60% complete. |
| N_02 | E/J data requirements have been submitted to Third party. Request X creation and review is complete. No further activities can be |

| Narrative # | Textual information |
|---|---|
| | carried out until Environment L data is built. |
| N_03 | Awaiting Integration Test data from the Third-party. Without a data build environment, the data must be built directly on the test environment. During this period the test environment is unavailable for testing. Therefore further delays on commissioning of Environment L will mean that Integration Test data cannot be delivered without stopping Assembly Test execution. |
| N_04 | Some preparation completed, but require Environment L data build to proceed further. Awaiting all Integration Test data from the Third-party. 3 week delay expected on Functionality X data – now expecting on Wk6. No revised ETC has been provided for Functionality Y data but expecting further delays. If Integration Test data can be provided by Wk6 Functionality X and dd/mm Functionality Wk8 then we could complete Integration Test 1 week later than currently planned. Component Catalogue Tool defect NNNNNNNNNNN raised with Third-party. This is preventing completion of environment build. Late delivery of these environments hinders test execution due to negative impact on fix quality and execution environment downtime. |
| N_05 | Some preparation completed, but require Environment L data build to proceed further. Awaiting all Integration Test data from the Third-party. 3 week delay expected on Functionality X data – now expecting on Wk6. No revised ETC has been provided for Functionality Y data but expecting further delays. If Integration Test data can be provided by Wk6 Functionality X and dd/mm Functionality Wk8 then we could complete Integration Test 1 week later than currently planned. Component Catalogue Tool defect NNNNNNNNNNN raised with Third-party. This is preventing completion of environment build. Late delivery of these environments hinders test execution due to negative impact on fix quality and execution environment downtime. |
| N_06 | Some preparation completed, but require Environment L data build to proceed further. Awaiting Integration Test data from Third-party. Revised ETC of Wk7. Integration Test preparation should have completed on Wk6. |
| N_07 | Some preparation completed, but require Environment L data build to proceed further. Awaiting Integration Test data from Third- |

| Narrative # | Textual information |
|---|---|
| | party. Re-plan based on assumption data delivered Wk7. Integration Test preparation should have completed on Wk6. |
| N_08 | Some preparation completed, but require Environment L data build to proceed further. Awaiting Integration Test data from Third-party. Re-plan based on assumption data delivered Wk9. Integration Test preparation should have completed on Wk6. |
| N_09 | [No performance report] |
| N_10 | [No performance report] |
| N_11 | [No performance report] |
| N_12 | Some preparation completed, but require Environment L1 test data from the Third-party build to proceed further. Re-plan to complete 7 weeks behind original plan on Wk12 - based on assumption Environment L data delivered, successfully pipecleaned and cloned by Wk13. P1_R2_IT1 Integration Test now includes 3 XX scenarios and scenarios for deferred defects in Assembley Test. 7 P1_R2_IT1 Integration Test scenarios with Supplier component changes in P1_R3_IT2 moved to P1_R3_IT2 Assembly Test – this does not change the timelines for testing these scenarios. |
| N_13 | Some preparation completed, but require Environment L1 test data from the Third-party build to proceed further. Re-plan to complete 8 weeks behind original plan on Wk13 - based on assumption Environment L data delivered, successfully pipecleaned and cloned by Wk14. P1_R2_IT1 Integration Test now includes 3 XX scenarios and scenarios for deferred defects in Assembley Test. 7 P1_R2_IT1 Integration Test scenarios with Supplier component changes in P1_R3_IT2 moved to P1_R3_IT2 Assembly Test – this does not change the timelines for testing these scenarios. |
| N_14 | [No performance report] |
| N_15 | All data successfully pipecleaned. Third-party are now cloning all P1_R3 Integration Test data. |
| N_16 | All master data successfully pipecleaned. Environment L test data clones built but not visible in the test environment. |

| Narrative # | Textual information |
|---|---|
| N_17 | [No performance report] |
| N_18 | Environment test data clones built and visible in the test environment. Complete. |

**Table A.5 Case P2-IT-PP Textual information**

| Narrative # | Textual information |
|---|---|
| N_01 | Integration Test execution should have started on Wk1. Re-planned to start and complete 3 weeks later due to delays in Environement L test data – revised completion date Wk7. This is based on the assumptions that all integration test data is delivered to test on Wk2 and data pipecleaning activities and cloning takes 10 days. Consuming Team's Change Requests NN and NN are not dependant on delayed data so will only complete Integraion Test execution 1 week behind the original Integration Test completion date – Wk5. If Integration Test completes more than 1 week behind the original plan we will encounter an impact to P1_R3 due to resource contention. |
| N_02 | Integration Test execution should have started on Wk1. Re-planned to start and complete 3 weeks later due to delays in Environement L test data – revised completion date Wk7. This is based on the assumptions that all integration test data is delivered to test on Wk2 and data pipecleaning activities and cloning takes 10 days. Consuming Team's Change Requests NN and NN are not dependant on delayed data so will only complete Integraion Test execution 1 week behind the original Integration Test completion date – Wk5. If Integration Test completes more than 1 week behind the original plan we will encounter an impact to P1_R3 due to resource contention. Execution readiness activities in progress. |
| N_03 | Integration Test execution should have started on Wk1. Re-planned to start and complete 5 weeks later due to delays in Environment L |

| Narrative # | Textual information |
|---|---|
| | test data – revised completion date Wk9. This is based on the assumptions that all integration test data is delivered to test Wk4 and data pipecleaning activities and cloning takes 10 days. Execution readiness activities in progress. |
| N_04 | [No performance report] |
| N_05 | [No performance report] |
| N_06 | [No performance report] |
| N_07 | Re-planned to complete 7 weeks later due to delays in the Third-party Environment L test data – revised completion date Wk11. This is based on based on the assumption that Environment L data is delivered, successfully pipecleaned and cloned by Wk8. 7 P1_R2 Integration Test scenarios with Supplier component changes in P1_R3 moved to P1_R3 Assembly Test – this does not change the timelines for testing these scenarios. Execution started Wk7. |
| N_08 | Re-planned to complete 7 weeks later due to delays in the Third-party Environment L test data – revised completion date Wk12. This is based on the assumption that Environment L data is delivered, successfully pipecleaned and cloned by Wk8. 7 P1_R2 Integration Test scenarios with Supplier component changes in P1_R3 moved to P1_R3 Assembly Test – this does not change the timelines for testing these scenarios. Execution started Wk7. |
| N_09 | [No performance report] |
| N_10 | Integration Test due to complete on Wk12. 54 Defects have been raised in total: 11 Rejected; 29 tested and passed; 2 Implementation Completed / Pending Completion / Pending Rejection; 3 Awaiting Impact / Awaiting Implementation; 9 Reviewed. |
| N_11 | High number of Design Queries have now been processed and teams are now working through resulting Defects and Design Defects backlog. Integration Test due to complete on Wk12. 64 Defects have been raised in total: 16 Rejected; 34 tested and passed; 5 Implementation Completed / Pending Completion / Pending Rejection; 6 Awaiting Impact / Awaiting Implementation; 3 Reviewed. |

| Narrative # | Textual information |
|---|---|
| N_12 | [No performance report] |
| N_13 | P1_R2_IT1 ST now due to complete Wk14. 36 points are still to be claimed in total: 4 are pending de-scope (Currently with the Client); 32 are blocked. 94 Defects have been raised in total: 60 Tested and Passed, 26 Rejected, 5 Implementation Completed/Pending Completion/Pending Rejection, 3 Awaiting Impact/Awaiting Implementation. |
| N_14 | P1_R2 Integration Test now due to complete Wk14. 25 points are still to be claimed in total. 12 points (Consumer Change Request NN) XXNNN issue. Currently under investigation. 8 are pending transfer into P1_R3_IT5 (Currently with Client). 4 points on component XXXNNN. Authentication Tool issue. 1 point on component XXXNNN. XXX signature error. |
| N_15 | P1_R2 Integration Test now due to complete Wk15. 1 points are still to be claimed in total. 1 point blocked by defect XXXXNNNNNNNN. Expected to be resolved Wk15. |
| N_16 | Completed Wk15. |

**Table A.6 Case P2-IT-Ex Textual information**

| Narrative # | Textual information |
|---|---|
| N_01 | Component Inventory has been discussed and is in process of being finalised by Sol Arc. Technical Design phase for CRM and ESB scheduled to start, but no FDs have been received. Technical Design phase for CRM and ESB scheduled to start, but no FDs have been received. Component Inventory has been discussed and is in process of being finalised by Sol Arc. No FDs in the traditional format have been received. Use Cases have been provided but they do not provide the level of functional coverage typical of previous FDs. Component Inventory – List of CRM and ESB components has not yet been finalised, hence scope cannot be considered closed. Build |

| Narrative # | Textual information |
|---|---|
| | schedule is therefore open to change. |
| N_02 | TD and Build started early for stable components. Technical Design phase for CRM and ESB scheduled to start, but no FDs have been yet received. Wireframes to be sent through by Sol Arch by end of week. Technical Design phase for CRM and ESB scheduled to start, but no FDs have yet been received. Wireframes to be sent through by Sol Arch by end of week. TD and Build started early for stable components. No FDs in the traditional format have been received. Use Cases have been provided but they do not provide the level of functional coverage typical of previous FDs. |
| N_03 | Components R and C items are blocked by Design Query (NNNNN, NNNNN – prioritised with Sol Arch). Component R development blocked by Design Query (NNNNN). Creation of new XXX_XX_XXX branch in progress (Technical Environement IssueTicket NNNNNN) – holding up check in and claiming of points. Technical Environment team targeting completion COP Thursday. Behind plan – SPI 0.5. Wireframes sent through by Sol Arch – Build clarifications raised as Design Queries. Recoverable on plan once blocking Design Queries and Technical Environment Issue Tickets are resolved. No FDs in the traditional format have been received. Wireframes have been provided and are being evaluated for functional gaps; suggested improvements will be fed back to Sol Arch team. |
| N_04 | [No performance report] |
| N_05 | TD/Build activities in progress. Technical Environement Issue Ticket NNNNNN – CRM DB extract in progress and blocked by underlying environment issues. Issues escalated with Technical Environement team as high priority. Behind plan – SPI 0.64. Recoverable on plan once blocking Technical Environement Issue Ticket is resolved. No FDs in the traditional format have been received. Wireframes have been provided and are being evaluated for functional gaps; suggested improvements will be fed back to Sol Arch team. Technical Environment Issue Ticket NNNNNN – CRM DB extract in progress and blocked by underlying environment |

| **Narrative #** | **Textual information** |
|---|---|
| | issues. Issues escalated with Technical Environement team as high priority. |
| N_06 | [No performance report] |
| N_07 | Previously blocking build environment Technical Environment Issue Ticket NNNNNN now resolved, unlocking UT for Components T and C. Remaining Build and Assembly Test activities continue to be impacted by Technical Environment Issue Tickets and Design Queries below:<br>XXX/XXX Launch: Design Queiry NNNNNN – Clarify expected applet behaviour for Component E. Design Defect to be raised.<br><br>Technical Environment Issue Ticket NNNNNN and NNNNNN – Assembly Test environment deploy issues. Environment XXXNN being used temporarily.<br><br>Design Query NNNNNN – Component S. Recently answered, UT now unlocked.<br><br>Behind plan – SPI 0.78. Progress continues to be impacted by Technical Environment Issue Tickets and Design Queries. Likely to overrun into next week – Original completion date was end of this week. Absolute deadline for completion is Wk10 (Consumer It1 Assembly Test start) – we expect to complete before.<br><br>No FDs in the traditional format have been received. Wireframes have been provided and are being evaluated for functional gaps; suggested improvements will be fed back to Sol Arch team. Further progress blocked by Technical Environment Issue Ticket NNNNNN and NNNNNN; and Design Query NNNNNN. Prioritised with Technical Environment team and Sol Arch. |

| Narrative # | Textual information |
|---|---|
| N_08 | XXXXXN and XXXXXN envs are available for use but continue to exhibit instability. XXXNN being used for Assembly Test activities while investigation continues.<br><br>3/5 work packages complete. Remaining Build and Assembly Test activities are impacted by Technical Environment Issue Ticket and Design Query below:<br>Component C: Technical Environment Issue Ticket NNNNNN – Log in access to Component Catalogue Tool. Preventing Build from progressing.<br><br>XXX/XXX Launch: Design Query NNNNNN – Clarify expected applet behaviour for Component E. Sol Arc due to discuss this with Consumer business representatives.<br><br>Behind plan – SPI 0.91. Progress continues to be impacted by a Technical Environment Issue Ticket and a Design Queries. Likely to overrun into next week – Original completion date was end of last week Wk7. Absolute deadline for completion is Wk10 (Consumer It1 BE Assembly Test start) – expected to complete before.<br><br>Further progress blocked by Technical Environment Issue Ticket NNNNNN; and Design Query NNNNNN. Prioritised with Technical Environment team and Sol Arch. |
| N_09 | XXXXXN and XXXXXN envs are available for use and operational. 4/5 work packages complete. XXX/XXX Launch: Design Query NNNNNN – Design change required; Design Defect NNNNN raised. Build based on current design is closed. |

| Narrative # | Textual information |
|---|---|
| | Component C: Discussing appropriate build approach with Cross project delivery teams. Downstream impact highlighted to Consumer Assembly Test team.<br><br>Behind plan – SPI 0.93. Completion held up by clarification on appropriate build approach for Component C. Likely to overrun into next week – Original completion date was end of last week Wk 7. Downstream impact highlighted to Consumer It1 BE Assembly Test team – due to start Wk10. |
| N_10 | XXXXXN and XXXXXN envs are available for use and operational. 4/5 work packages complete. Targeting completion of final Component C package by end of week.<br><br>Component C: Key resource has been on vacation. Prioritisation of other DBT activities has led to this package being postponed until resource returns. Downstream impact highlighted to Consumer Assembly Test team.<br><br>Behind plan – SPI 0.93. Completion held up by clarification on appropriate build approach for Component C. Targeting completion by end of week. Downstream impact highlighted to Consumer It1 BE Assembly Test team – due to start Wk10. |
| N_11 | XXXXXN and XXXXXN envs are available for use and operational. 4/5 work packages complete. Targeting completion of final Component C package by end of week.<br><br>Component C: Technical Environment Issue Ticket NNNNN – issue with test harness preventing injection of messages into ESB. |

| Narrative # | Textual information |
|---|---|
| | Highlighted to downstream Consumer Assembly Test team.<br><br>Slightly Behind Plan – [SPI: 0.93, Comp: 93%]. Completion held up by Unit Test delay of Component C package due to env issue. Targeting completion by end of week. Downstream impact highlighted to Consumer It1 BE Assembly Test team. |
| N_12 | XXXXXN and XXXXXN envs are available for use and operational. 4/5 work packages complete. Targeting completion of final Component C package by end of week.<br><br>Component C: Technical Environment Issue Ticket NNNNN – token correction in env file; prevents test harness messages being picked up from ESB queue. Prioritised with Technical Environment team. Highlighted to downstream Consumer Assembly Test team.<br><br>Behind plan – SPI 0.93. Completion held up by Unit Test delay of Component C package due to env issue. Targeting completion by end of week. Downstream impact highlighted to Consumer It1 BE Assembly Test team. |
| N_13 | XXXXXN and XXXXXN envs are available for use and operational. 4/5 work packages complete. Targeting completion of final Component C package by end of week.<br><br>Component C: Multiple issues have been worked through. Technical Environment Issue Ticket NNNNN – messages not picked up from ESB queue. Under investigation with Build and Technical Environment team. Highlighted to downstream Consumer Assembly Test team. |

| Narrative # | Textual information |
|---|---|
| | Behind plan – SPI 0.93. Completion held up by Unit Test delay of Component C package due to env issue. Targeting completion by end of week. Downstream impact highlighted to Consumer It1 BE Assembly Test team. |
| N_14 | Complete – [SPI: 1.00, Comp: 100%]. 5/5 work packages complete. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |
| N_15 | Complete – [SPI: 1.00, Comp: 100%]. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |
| N_16 | Complete – [SPI: 1.00, Comp: 100%]. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |
| N_17 | Complete – [SPI: 1.00, Comp: 100%]. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |
| N_18 | Complete – [SPI: 1.00, Comp: 100%]. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |
| N_19 | Complete – [SPI: 1.00, Comp: 100%]. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |
| N_20 | Complete – [SPI: 1.00, Comp: 100%]. Environmental issues resolved. Completed build numbers publicised to Assembly Test and Integration Test teams for deployment. |

**Table A.7 Case P3-DBT (Inc6) Textual information**

## A4. Analytic memos

In the course of coding the textual data, a number of analytic memos were written to record the thoughts emerged during the analysis, and which helped in refining and comparing the emergent codes/sub-categories and their relationships. Section 7.4.1 in Chapter 7 outlined the special role of the category ‘Schedule’ in this research; thus, the following presents exerts of the memos for this category and example sub-categories.

### A4.1 Memo: Schedule

This memo attempts to define the category 'Schedule' and its properties and relationships. Schedule refers to the means by which project/phase activities are organised on a timeline showing what activity is planned to be started and completed by when - an example would be the Gantt chart used widely in the project management discipline. Schedule seems to manifest itself in various ways, such as: delay in achieving event targets (Activity delay), delay in completing the phase on schedule (Phase delay), shortening the schedule during Test phase execution (Duration compression), lengthening the schedule during Test phase execution (Duration extension), actual execution may start later than scheduled (Start variance), the Test manager may have no control over progress/actual schedule (Lacking control over progress), the schedule may have not been baselined during execution (No schedule baseline), the schedule may include vacation time (Nonworking time), and the Test manager may report progress status of the schedule (Reporting progress status).

### A4.2 Memo: Activity delay

Activity delay represents a situation where fewer activities were carried out (i.e. fewer event targets were achieved) compared to what were scheduled to be carried out at a particular point in time. The missed events can be recovered without affecting subsequent phases or the project finish day; i.e. without affecting the project’s critical path; for example, through working overtime to catch up. Activity delay could be due to: product defect such as Authentication tool defect or Code defect; or due to delay in providing or fixing product, such as delay in providing Code or delay in fixing Design defect; or delay in providing information such as delay in providing clarification on Requirements gap; or other factors such as introducing new product in the middle of Test phase execution.

### A4.3 Memo: Phase delay

Phase delay represents completing project phase later than was scheduled. Phase delay differs from Activity delay in that, missing the phase's scheduled completion day may impact subsequent phases and possibly the project finish day; i.e. the critical path for the project is affected. Phase delay can be paralleled with the Project schedule delay measure introduced in section 5.3.3.1 (Chapter 5) where the number of days in which a phase was late was calculated. Phase delay can occur due to several factors; for example: due to product defect such as Technical environment defect, or due to delay in making decision such as delay in deciding on (signing off) Test completion report; or due to delay in providing product such as delay in providing Peer supplier Test data; or due to delay in fixing defect such as delay in fixing Technical environment defect.

### A.4.4 Memo: Duration compression

Compressing schedule duration refers to shortening the schedule, during execution, in order to complete the phase earlier than was scheduled to complete; or to doing the same volume/amount of work, which was scheduled to do, in a shorter period of time in order to maintain the original completion day. The decision to compress duration may be made internally by the phase manager and project manager without involving the Client, in order to avoid extending the phase or project finish date. This may be achieved through getting the phase team to work overtime in order to do the same volume of work in a shorter period. Duration compression can be paralleled with the Phase schedule accuracy metric introduced in section 5.3.3.5 (Chapter 5) where the duration of a project phase appears to have been overestimated. Duration compression occurs due to delays in providing product such as Test environment.

### A4.5 Memo: Duration extension

Extending schedule duration refers to lengthening the schedule, during execution, in order to complete the phase later than was scheduled to complete; or doing the same volume/amount of work, that was scheduled originally to do, but in a longer period of time. The decision to extend schedule is often made after agreement with the Client, as schedule extensions typically have implications on subsequent phases and possibly project finish day. Duration extension can be paralleled with the Phase schedule accuracy metric introduced in section 5.3.3.5 (Chapter 5) where the duration of a project phase appears to have been underestimated. Duration extension occurs due to product defects such as Authentication tool defect; and due to delay in providing product such as delay in providing Peer supplier Test data.

# Bibliography

Abdel-Hamid, T., & Madnick, S. E. (1991). *Software Project Dynamics: An Integrated Approach*. USA: Prentice-Hall.

Ackoff, R. L. (1979). The Future of Operational Research is Past. *Journal of the Operational Research Society, 30*(2), 93-104.

Adolph, S., Kruchten, P., & Hall, W. (2012). Reconciling perspectives: A grounded theory of how people manage the process of software development. *The Journal of Systems and Software, 85*, 1269– 1286.

Akkermans, H., & Helden, K. v. (2002). Vicious and virtuous cycles in ERP implementation: a case study of interrelations between critical success factors. *European Journal of Information Systems, 11*(1), 35–46.

Akrich, M., Callon, M., & Latour, B. (2002). The Key to Success in Innovation Part I: The Art of Interessement. *International Journal of Innovation Management, 6*(2), 187–206.

Akrich, M., & Latour, B. (1992). A Summary of a Convenient Vocabulary for the Semiotics of Human and Nonhuman Assemblies. In W. E. Bijker & J. Law (Eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (pp. 259-264). London: MIT Press.

Alberts, C. J., & Dorofee, A. J. (2010). *Risk Management Framework*. Retrieved from Carnegie Mellon University:

Allison, I. (2005). Towards an agile approach to Software Process Improvement: addressing the changing needs of software products. *Communications of the IIMA, 5*(1), 67-76.

Allison, I. (2010). *Organizational factors shaping software process improvement in small-medium sized software teams: A multi-case analysis*. Paper presented at the 2010 Seventh International Conference on the Quality of Information and Communications Technology, Porto.

Allison, I., & Merali, Y. (2007). Software process improvement as emergent change: A structurational analysis. *Information and Software Technology, 49*(6), 668–681.

Anbari, F. T. (2003). Earned Value Project Management Method And Extensions. *Project Management Journal, 34*(4), 12-23.

Anderson, L. (2006). Analytic Autoethnography. *Journal of Contemporary Ethnography, 35*(4), 373-395.

Andrade, A. D., & Urquhart, C. (2010). The affordances of actor network theory in ICT for development research. *Information Technology & People, 23*(4), 352-374.

APM. (2006). *APM Body of Knowledge* (5th EDITION ed.). UK: Association for Project Management.

Bakker, K. d., Boonstra, A., & Wortmann, H. (2010). Does risk management contribute to IT project success? A meta-analysis of empirical evidence. *International Journal of Project Management, 28*(5), 493-503.

Balaji, S., Ahuja, M. K., & Ranganathan, C. (2006). *Offshore Software Projects: Assessing the Effect of Knowledge Transfer Requirements and ISD capability*. Paper presented at the Proceedings of the 39th Annual Hawaii International Conference on In System Sciences, 2006. HICSS'06, Hawaii.

Bannerman, P. L. (2008). Risk and risk management in software projects: A reassessment. *The Journal of Systems and Software, 81*(12), 2118-2133.

Basili, V. R., Rombach, D., & Schneider, K. (2006). Preface. In V. R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, & R. Selby (Eds.), *Empirical Software Engineering Issues: Critical Assessment and Future Directions* (pp. V-XI). International Workshop, Dagstuhl Castle, Germany: Springer-Verlag Berlin Heidelberg.

Basili, V. R., Selby, R. W., & Hutchens, D. H. (1986). Experimentation in Software Engineering. *IEEE Transactions on Software Engineering, SE-12*(7), 733-743.

Baskerville, R., & Pries-Heje, J. (1999). Grounded action research: a method for understanding IT in practice. *Accounting, Management & Information Technologies, 9*(1), 1–23.

Bass, J. M., Allison, I. K., & Banerjee, U. (2013). Agile Method Tailoring in a CMMI Level 5 Organization: Addressing the Paradox. *Journal of International Technology and Information Management, 22*(4), 77-98.

Battin, R. D., Crocker, R., Kreidler, J., & Subramanian, K. (2001). Leveraging Resources in Global Software Development. *IEEE Software, 18*, 70-77.

Bauer, H. A., & Birchall, R. H. (1978). *Managing large scale software development with an automated change control system*. Paper presented at the IEEE Computer Society's Second International Computer Software and Applications Conference, 1978. COMPSAC '78.

Belady, L. A., & Lehman, M. M. (1976). A model of large program development. *IBM Systems journal, 15*(3), 225-252.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly, 11*(3), 369-386.

Biesta, G. (2010). Pragmatism and the Philosophical Foundations of Mixed Methods Research. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 95-117). USA: SAGE Publications, Inc.

Bijker, W. E., & Pinch, T. (2012). Preface to the Anniversary Edition. In W. E. Bijker, T. P. Hughes, & T. Pinch (Eds.), *The Social Construction of Technological Systems* (Anniversary Edition ed., pp. xi-xxxiv): Massachusetts Institute of Technology.

Blackburn, J., Scudder, G., & Wassenhove, L. N. V. (1996). *Improving Speed and Productivity of Software Development*. INSEAD Working Paper Series. Technology Mnagement INSEAD. Fontainebleau, France.

Blackburn, S. (2002). The project manager and the project-network. *International Journal of Project Management, 20*(3), 199–204.

Bloomfield, B. P., Coombs, R., Cooper, D. J., & Rea, D. (1992). Machines and manoeuvres: Responsibility accounting and the construction of hospital information systems. *Accounting, Management and Information Technologies, 2*(4), 197–219.

Bloomfield, B. P., Coombs, R., Owen, J., & Taylor, P. (1997). Doctors as Managers: Constructing Systems and Users in the National Health Service. In B. P. Bloomfield, R. Coombs, D. Knights, & D. Litter (Eds.), *Information Technology and Organizations: Strategies, Networks, and Integration* (pp. 113-134). USA: Oxford University Press Inc.

Bloomfiled, B. P., & Vurdubakis, T. (1997). Paper Traces: Inscribing Organizations and Information Technology. In B. P. Bloomfield, R. Coombs, D. Knights, & D. Litter (Eds.), *Information Technology and Organizations: Strategies, Networks, and Integration* (pp. 85-111). USA: Oxford University Press Inc.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer, 35*(1), 64-69.

Boehm, B. W. (1981). *Software Engineering Economics*. USA: Prentice-Hall.

Boehm, B. W., & Ross, R. (1989). Theory-W Software Project Management: Principles and Examples. *IEEE Transactions on Software Engineering, 15*(7), 902-916.

Bower, D. C. (2007). *New Directions in Project Performance and Progress Evaluation.* (PhD), RMIT University, Melbourne, Australia.

Bower, D. C., & Finegan, A. D. (2009). New approaches in project performance evaluation techniques. *International Journal of Managing Projects in Business, 2*(3), 435-444.

Bryant, A. (2002). Re-Grounding Grounded Theory. *Journal of Information Technology*

*Theory and Application (JITTA), 4*(1), 25-42.

Budd, C. I., & Budd, C. S. (2010). *A Practical Guide to Earned Value Project Management* (Second Edition ed.). USA: Management Concepts, Inc.

Burke, R. (2013). *Project management: Planning and Control Techniques* (Fifth Edition ed.). UK: Wiley.

Butler, T., & Fitzgerald, B. (1999). Unpacking the systems development process: an empirical application of the CSF concept in a research context. *Journal of Strategic Information Systems*, *8*(4), 351–371.

Cadle, J., & Yeates, D. (2008). *Project Management for Information Systems* (F. edition Ed.). England: Pearson Education.

Callon, M. (1986). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. In J. Law (Ed.), *Power, action and belief: a new sociology of knowledge?* (pp. 196-233). London: Routledge.

Callon, M. (1991). Techno-economic Networks and Irreversibility. In J. Law (Ed.), *A Sociology of Monsters: Essays on Power, Technology and Domination* (pp. 132-161). London: Routledge.

Callon, M. (2012). Society in the Making: The Study of Technology as a Tool for Sociological Analysis. In W. E. Bijker, T. P. Hughes, & T. Pinch (Eds.), *The Social Construction of Technological Systems* (Anniversary Edition ed., pp. 77-97). London: Massachusetts Institute of Technology.

Callon, M., & Law, J. (1989). On the Construction of Sociotechnical networks: content and context revisited. *Knowledge and Society: Studies in the Sociology of Science Past and Present*, *8*(1), 57-83.

Cameron, R. (2012). *Applying the Newly Developed Extended Mixed Methods Research (MMR) Notation System*. Paper presented at the British Academy of Management 2012 Conference, 11-13th September, Cardiff, Wales.

Cameron, R. (2013). *A Methodological Scan of a National Industry Based Research Program for the Rail Industry 2007-2014.* Paper presented at the 12th European Conference on Research Methodology for Business and Management Studies, Portugal.

Cameron, R., & Sankaran, S. (2013). Mixed Methods Research Design: Well Beyond the Notion of Triangulation. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 383-401). Universitetsforlaget: Copenhagen Business School Press.

Carmel, E. (1995). Time-to-completion factors in packaged software development.

*Information and Software Technology, 37*(9), 515-520.

Carmel, E., & Agarwal, R. (2001). Tactical Approaches for Alleviating Distance in Global Software Development. *IEEE Software, 18,* 22-29.

Carr, M. J., Konda, S. L., Monarch, I., Ulrich, F. C., & Walker, C. F. (1993). *Taxonomy-Based Risk Identification*. Retrieved from Carnegie Mellon University:

Carver, J. C. (2003). *The Impact of Background and Experience on Software Inspections.* (Doctor of Philosophy), University of Maryland, College Park, USA.

Casey, V., & Richardson, I. (2006). *Project Management within Virtual Software Teams*. Paper presented at the International Conference on Global Software Engineering, 2006. (ICGSE'06), Florianopolis.

Cerpa, N., & Verner, J. M. (2009). Why Did Your Project Fail? *Communications of the ACM, 52,* 130-134.

Chai, K.-H., & Xin, Y. (2006). The application of new product development tools in industry: the case of Singapore. *IEEE Transactions on Engineering Management, 53*(4), 543-554.

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

Checkland, P. (1981). *Systems Thinking, Systems Practice*. Great Britain: John Wiley & Sons Ltd.

Chia, R. (2013). Paradigms and Perspectives in Organizational Project Management Research: Implications for Knowledge-Creation. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 33-55). Universitetsforlaget: Copenhagen Business School Press.

Cho, S., Mathiassen, L., & Nilsson, A. (2008). Contextual dynamics during health information systems implementation: an event-based actor-network approach. *European Journal of Information Systems, 17*(6), 614-630.

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for Development – Guidelines for Process Integration and Product Improvement* (Third Edition ed.). USA: Addison-Wisely.

Chua, B. B., & Verner, J. M. (2005). Risk management practices and tools: A pilot study of Australian software development projects.

Cicmil, S., Williams, T., Thomas, J., & Hodgson, D. (2006). Rethinking Project

Management: Researching the actuality of projects. *International Journal of Project Management*, *24*(8), 675–686.

Ciolkowski, M., & Briand, L. (2006). Roadmapping: Working Group 2 Results. In V. R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, & R. Selby (Eds.), *Empirical Software Engineering Issues: Critical Assessment and Future Directions* (pp. 175-177). International Workshop, Dagstuhl Castle, Germany: Springer-Verlag Berlin Heidelberg.

Clegg, C., Axtell, C., Damodaran, L., Farbey, B., Hull, R., Lloyd-Jones, R., . . . Tomlinson, C. (1997). Information technology: a study of performance and the role of human and organizational factors. *Ergonomics, 40*(9), 851-871.

Clegg, S. (2013). Foreword. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 17-18). Universitetsforlaget: Copenhagen Business School Press.

Cockburn, A. (2008). Using Both Incremental and Iterative Development. *CROSSTALK The Journal of Defense Software Engineering*, 27-30.

Coleman, G., & O’Connor, R. (2007). Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology, 49*(6), 654–667.

Collins, K. M. T. (2010). Advanced Sampling Designs in Mixed Research: Current Practices and Emerging Trends in the Social and Behavioral Sciences. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 353-377). USA: SAGE Publications, Inc.

Conchúir, E. Ó., Ågerfalk, P. J., Olsson, H. H., & Fitzgerald, B. (2009). Global Software Development: Where are the Benefits? *Communications of the ACM, 52,* 127-131.

Conte, S. D., Dunsmore, H. E., & Shen, V. Y. (1986). *Software engineering metrics and models*: Benjamin-Cummings Publishing Co., Inc..

Corbin, J., & Strauss, A. (2008). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3e ed.). USA: Sage Publications Inc.

Corporation, M. (2012). Finish Variance fields.

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (Third Edition ed.). USA: SAGE Publications Inc.

Creswell, J. W., Clark, V. L. P., & Garrett, A. L. (2008). Methodological Issues in

Conducting Mixed Methods Research Designs. In M. M. Bergman (Ed.), *Advances in Mixed Methods Research: Theories and Applications* (pp. 66-83). UK: SAGE Publications Ltd.

Creswell, J. W., & PlanoClark, V. L. (2011). *Designing and Conducting Mixed Methods Research* (2nd Edition ed.). USA: SAGE Publications Inc.

Crotty, M. (2013). *The Foundations of Social Research: Meaning and Perspective in the Research Process*. London: SAGE Publications.

Curtis, B., Krasner, H., & Isoce, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31,* 1268-1287.

Damian, D., & Lanubile, F. (2004). *The 3rd International Workshop on Global Software Development.* Paper presented at the Proceedings of the 26th International Conference on Software Engineering (ICSE'04), Edinburgh, UK.

Damian, D., & Moitra, D. (2006). Global Software Development: How Far Have We Come? *IEEE Software, 23,* 17-19.

Dankert, R. (2009a). How to balance between embedding and deviation?

Dankert, R. (2009b). Is Latour's due process feasible?

Dankert, R. (2011). Using Actor-Network Theory (ANT) doing research.

Deephouse, C., Mukhopadhyay, T., Goldenson, D. R., & Kellner, M. I. (1996). Software Processes and Project Performance. *Journal of Management Information Systems, 12*(3), 187-205.

DeMarco, T. (1982). *Controlling Software Projects: Management, Measurement & Estimation*. USA: Prentice-Hall.

DeMarco, T. (2011). All Late Projects are the Same. *IEEE Software, 28,* 103-104.

DeMarco, T., & Boehm, B. (2002). The Agile Methods Fray. *Computer, 35*(6), 90-92.

Dhlamini, J., Nhamu, I., & Kachepa, A. (2009). *Intelligent Risk Management Tools for Software Development.* Paper presented at the Proceeding SACLA '09 Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association, Eastern Cape, South Africa.

Doll, W. J., Deng, X., & Scazzero, J. A. (2003). A process for post-implementation IT benchmarking. *Information & Management, 41*(2), 199–212.

Donmoyer, R. (2000). Generalizability and the Single-Case Study. In R. Gomm, M. Hammersley, & P. Foster (Eds.), *Case Study Method: Key Issues, key Texts* (pp. 45-68). SAGE Publications Ltd.

Ebert, C. (2007). The impacts of software product management. *The Journal of Systems and Software, 80*(8), 850–861.

Ebert, C., & Neve, P. D. (2001). Surviving Global Software Development. *IEEE Software, 18*, 62-69.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *The Academy of Management Review, 14*(4), 532-550.

Elbanna, A. R. (2008). Strategic systems implementation: diffusion through drift. *Journal of Information Technology, 23*, 89–96.

Er, M., Pollack, J., & Sankaran, S. (2013). Actor-Network Theory, Activity Theory and Action Research and their Application in Project Management Research. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 164-198). Universitetsforlaget: Copenhagen Business School Press.

Fan, C.-F., & Yu, Y.-C. (2004). BBN-based software project risk management. *The Journal of Systems and Software, 73*(2), 193-203.

Feldon, D. F., & Kafai, Y. B. (2008). Mixed methods for mixed reality: understanding users' avatar activities in virtual worlds. *Educational Technology Research and Development, 56*(5-6), 575-593.

Feng, N., Li, M., & Gao, H. (2009). *A Software Project Risk Analysis Model Based on Evidential Reasoning Approach*. Paper presented at the WRI World Congress on Software Engineering, 2009. WCSE '09, Xiamen.

Fitzgerald, B. (1998). An empirical investigation into the adoption of systems development methodologies. *Information & Management, 34*(6), 317–328.

Fleming, Q. W., & Koppelman, J. M. (2010). *Earned Value Project Management* (Fourth Edition ed.). USA: Project Management Institute.

Fleming, Q. W., & Koppelman, J. M. (July 1998). Earned Value Project Management: A Powerful Tool for Software Projects. *CROSSTALK The Journal of Defense Software Engineering*, 19-23.

Frederick P. Brooks, J. (1995). *The Mythical Man-Month: Essays on Software Engineering* (Anniversary edition ed.). USA: Addison Wesley.

Georgieva, S., & Allan, G. (2008). Best Practices in Project Management Through a Grounded Theory Lens. *The Electronic Journal of Business Research Methods, 6*(1), 43 - 52.

Giddens, A. (1984). *The Constitution of Society: Outline of the Theory of Structuration*. Great Briton: Polity Press.

Gilb, T. (1985). Evolutionary Delivery versus the "Waterfall model". *ACM Sigsoft Software Engineering Notes, 10*(3), 49-61.

Gilb, T. (1988). *Principles of Software Engineering Management*: Addison-Wesley.

Glaser, B. G. (1978). *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. USA: The Sociology Press.

Glaser, B. G., & Strauss, A. L. (1967). *The Discovery of Grounded Theory: strategies for qualitative research*. USA: AldineTransaction.

Goldratt, E. M. (1997). *Critical Chain: A Business Novel*. UK: Gower.

Gomm, R., Hammersley, M., & Foster, P. (2000). Case Study and Generalization. In R. Gomm, M. Hammersley, & P. Foster (Eds.), *Case Study Method: Key Issues, Key Texts* (pp. 98-115). UK: SAGE Publications Ltd.

Graham, D. R. (1992). *Incremental Development and Delivery for Large Software Systems*. Paper presented at the IEE Colloquium on Software Prototyping and Evolutionary Development, London.

Greene, J. C., & Hall, J. N. (2010). Dialectics and Pragmatism: Being of Consequences. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 119-143). USA: SAGE Publications, Inc.

Greenfield, J., & Short, K. (2003). *Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools*. Paper presented at the OOPSLA '03 Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, USA.

Greenhalgh, T., & Stones, R. (2010). Theorising big IT programmes in healthcare: Strong structuration theory meets actor-network theory. *Social Science & Medicine, 70*(9), 1285–1294.

Guba, E. G., & Lincoln, Y. S. (1989). *Fourth Generation Evaluation*. USA: SAGE Publications, Inc.

Guba, E. G., & Lincoln, Y. S. (1994). Competing Paradigms in Qualitative Research. In N. K. Denzin & Y. S. Lincoln (Eds.), *Handbook of Qualitative Research* (pp. 105-117). USA: Sage Publications, Inc.

Hammersley, M., Gomm, R., & Foster, P. (2000). Case Study and Theory. In R. Gomm, M. Hmmersley, & P. Foster (Eds.), *Case Study Method: Key Issues, Key Texts* (pp. 234-258). London: SAGE Publications Ltd.

Hansen, B. H., & Kautz, K. (2005). *Grounded Theory Applied – Studying Information Systems Development Methodologies in Practice*. Paper presented at the

Proceedings of the 38th Hawaii International Conference on System Sciences - 2005, Hawaii.

Hanseth, O., Jacucci, E., Grisot, M., & Aanestad, M. (2006). Reflexive Standardization: Side Effects and Complexity in Standard Making. *MIS Quarterly, 30*(Special issue), 563-581.

Harbers, H. (1995). We Have Never Been Modern by Bruno Latour. *Science, Technology, & Human Values, 20*(2), 270-275.

Henderson, K. (2003). Earned schedule: A breakthrough extension to earned value theory? A retrospective analysis of real project data. *The Measurable News, 1,* 13-23.

Henderson, K. (2006). *Earned Schedule in Action.* Paper presented at the Earned Value Analysis - 11 Conference, London, United Kingdom.

Henderson, K. (2007). *Earned Schedule: A Breakthrough Extension to Earned Value Management.* Paper presented at the PMI Asia Pacific Global Congress Proceedings, Hong Kong.

Herbsleb, J. D. (2007). *Global Software Engineering: The Future of Socio-technical Coordination*. Paper presented at the Future of Software Engineering (FOSE '07), Minneapolis, MN.

Herbsleb, J. D., & Grinter, R. E. (1999). Architectures, Coordination, and Distance: Conway’s Law and Beyond. *IEEE Software, 16,* 63-70.

Herbsleb, J. D., & Mockus, A. (2003). An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering, 29*(6), 481-494.

Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2001). *An Empirical Study of Global Software Development: Distance and Speed.* Paper presented at the Proceedings of the 23rd International Conference on Software Engineering. ICSE '01, USA.

Herbsleb, J. D., & Moitra, D. (2001). Global Software Development. *IEEE Software, 18,* 16-20.

Herbsleb, J. D., Paulish, D. J., & Bass, M. (2005). *Global Software Development at Siemens: Experience from Nine Projects.* Paper presented at the Proceedings of the 27th international conference on Software engineering. ICSE '05, USA.

Herroelen, W., Leus, R., & Demeulemeester, E. (2002). Critical Chain Project Scheduling: Do Not Oversimplify. *Project Management Journal, 33*(4), 48-60.

Hoda, R., Kruchten, P., Noble, J., & Marshall, S. (2010). *Agility in Context.* Paper presented at the Proceedings of the ACM international conference on Object oriented programming systems languages and applications OOPSLA '10, Reno/Tahoe, Nevada USA.

Hoegl, M., & Weinkauf, K. (2005). Managing task interdependencies in Multi-Team projects: A longitudinal study. *Journal of Management Studies, 42*(6), 1287-1308.

Holmstrom, H., Conchúir, E. Ó., Ågerfalk, P. J., & Fitzgerald, B. (2006). *Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance.* Paper presented at the IEEE International Conference on Global Software Engineering (ICGSE'06), Florianopolis.

Holton, J. A. (2007). The Coding Process and Its Challenges. In A. Bryant & K. Charmaz (Eds.), *The SAGE Handbook of Grounded Theory* (Paperback Edition ed., pp. 265-289). UK: SAGE Publications Ltd.

Hossain, E., Babar, M. A., & Verner, J. (2009). *How Can Agile Practices Minimize Global Software Development Co-ordination Risks?* Paper presented at the Proceedings of the 16th European Conference on Software Process Improvement, EuroSpi 2009, Alcala (Madrid), Spain.

Hughes, B. (2000). *Practical Software Measurement*. Cambridge: McGraw-Hill.

Hughes, B., & Cotterell, M. (2009). *Software Project Management* (Fifth Edition ed.). Berkshire: McGraw-Hill.

Hughes, T. P. (2012). The Evolution of Large Technological Systems. In W. E. Bijker, T. P. Hughes, & T. Pinch (Eds.), *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology* (Anniversary Edition ed., pp. 45-76). USA: MIT.

Hustad, E., & Lange, C. d. (2014). Service-oriented architecture projects in practice: A study of a shared document service implementation. *Procedia Technology, 16*, 684 – 693.

Ikonen, M., Kettunen, P., Oza, N., & Abrahamsson, P. (2010). *Exploring the Sources of Waste in Kanban Software Development Projects*. Paper presented at the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Lille.

Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P., & Abraha, P. (2011). *On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation*. Paper presented at the 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), Las Vegas, NV.

Ivankova, N., & Kawamura, Y. (2010). Emerging Trends in the Utilization of Integrated Designs in the Social, Behavioral, and Health Sciences. In A. Tashakkori & C.

Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 581-611). USA: SAGE Publications, Inc.

Jacobson, I., Meyer, B., & Soley, R. (2012). Software Engineering Method and Theory – A Vision Statement.

James E. Kelley, J., & Walker, M. R. (1959). *Critical-Path Planning and Scheduling*. Paper presented at the Proceeding IRE-AIEE-ACM '59 (Eastern) Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference, New York.

Janes, A., & Succi, G. (2009). *To Pull or Not to Pull.* Paper presented at the OOPSLA 2009 Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, Florida, USA.

Jenkins, A. M., Naumann, J. D., & Wetherbe, J. C. (1984). Empirical Investigation of Systems Development Practices and Results. *Information & Management, 7,* 73-82.

Jiang, J. J., Klein, G., & Ellis, T. S. (2002). A Measure of Software Development Risk. *Project Management Journal, 33*(3), 30-41.

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a Definition of Mixed Methods Research. *Journal of Mixed Methods Research, 1*(2), 112-133.

Jones, C. (1995). Patterns of large software systems: Failure and success. *Computer, 28*(3), 86-87.

Kelle, U. (2007). The Development of Categories: Different Approaches in Grounded Theory. In A. Bryant & K. Charmaz (Eds.), *The SAGE Handbook of Grounded Theory* (pp. 191-213). London: SAGE Publication Ltd.

Kerzner, H. R. (2013). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling* (Eleventh edition ed.). USA: Wiley.

Kim, E., Jr., W. G. W., & Duffey, M. R. (2003). A model for effective implementation of Earned Value Management methodology. *International Journal of Project Management, 21*(5), 375–382.

Kim, H.-W., & Pan, S. L. (2006). Towards a process model of information systems implementation: the case of customer relationship management (CRM). *Database for Advances in Information Systems, 37*(1), 59-76.

Kitchenham, B. (2006). Empirical Paradigm – The Role of Experiments. In V. R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, & R. Selby (Eds.), *Empirical Software Engineering Issues: Critical Assessment and Future Directions* (pp. 25-32). International Workshop, Dagstuhl Castle, Germany: Springer-Verlag Berlin Heidelberg.

Kitchenham, B., Pfleeger, S. L., & Fenton, N. (1995). Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering, 21*(12), 929-944.

Kitchenham, B. A. (1987). Controlling software projects. *Electronics & Power*, 312-315.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., & Rosenberg, J. (2002). Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering, 28*(8), 721-734.

Kommeren, R., & Parviainen, P. (2007). Philips experiences in global distributed software development. *Empirical Software Engineering, 12*(6), 647–660.

Kumar, R. (2011). *Research Methodology* (3rd edition ed.). London: SAGE Publications Ltd.

Kutsch, E., & Hall, M. (2009). The Rational Choice of Not Applying Project Risk Management in Information Technology Projects. *Project Management Journal, 40*(3), 72–81.

Kwak, Y. H., & Anbari, F. T. (2011). History, Practices, and Future of Earned Value Management in Government: Perspectives From NASA. *Project Management Journal, 43*(1), 77–90.

Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development - A Brief History. *Computer, 36*(6), 47-56.

Latour, B. (1986). The powers of association. In J. Law (Ed.), *Power, action and belief: a new sociology of knowledge?* (pp. 264-280). London: Routledge.

Latour, B. (1987). *Science in Action: How to follow scientists and engineers through society*. USA: Harvard University Press.

Latour, B. (1991). Technology is society made durable. In J. Law (Ed.), *A Sociology of Monsters: Essays on Power, Technology and Domination* (pp. 103-131). London: Routledge.

Latour, B. (1996). *Aramis, or, the love of technology*. Cambridge, MA: Harvard University Press.

Latour, B. (1999). On Recalling ANT. In J. Law & J. Hassard (Eds.), *Actor Network Theory and after* (pp. 15-25). UK: Blackwell Publishers.

Latour, B. (2004a). On using ANT for studying information systems: a (somewhat) Socratic dialogue. In C. Avgerou, C. Ciborra, & F. Land (Eds.), *The Social Study of Information and Communication Technology: Innovation, Actors, and*

*Contexts* (pp. 62-76). USA: Oxford University Press Inc.

Latour, B. (2004b). *Politics of nature: How to Bring the Sciences into Democracy*. UK: Harvard University Press.

Latour, B. (2005). *Reassembling the Social - An Introduction to Actor-Network-Theory*. USA: Oxford University Press.

Law, J. (1986). On the methods of long-distance control: vessels, navigation and the Portuguese rout to India. In J. Law (Ed.), *Power, action and belief: a new sociology of knowledge?* (pp. 234-263). London: Routledge.

Law, J. (1991). Introduction: monsters, machines and sociotechnical relations. In J. Law (Ed.), *A Sociology of Monsters: Essays on Power, Technology and Domination* (pp. 1-23). London: Routledge.

Law, J. (1999). After ANT: complexity, naming and topology. In J. Law & J. Hassard (Eds.), *Actor Network Theory and after* (pp. 1-13): Blackwell Publishers.

Law, J. (2012). Technology and Heterogeneous Engineering: The Case of Portuguese Expansion. In W. E. Bijker, T. P. Hughes, & T. Pinch (Eds.), *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology* (Anniversary Edition ed., pp. 105-127). London: MIT.

Law, J., & Callon, M. (1992). The Life and Death of an Aircraft: A Network Analysis of Technical Change. In W. E. Bijker & J. Law (Eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (pp. 21-52). London: MIT Press.

Leach, L. P. (1999). Critical Chain Project Management Improves Project Performance. *Project Management Journal, 30*(2), 39-51.

Lechler, T. G., & Cohen, M. (2009). Exploring the Role of Steering Committees in Realizing Value From Project Management. *Project Management Journal, 40*(1), 42–54.

Lechler, T. G., Ronen, B., & Stohr, E. A. (2005). Critical Chain: A New Project Management Paradigm or Old Wine in New Bottles? *Engineering Management Journal, 17*(4), 45-58.

Lee-Kelley, L. (2006). Locus of control and attitudes to working in virtual teams. *International Journal of Project Management, 24*(3), 234–243.

Lehtinen, T. O. A., & Mäntylä, M. V. (2011). *What Are Problem Causes of Software Projects? Data of Root Cause Analysis at Four Software Companies*. Paper presented at the International Symposium on Empirical Software Engineering and Measurement (ESEM), 2011, Banff, AB.

Lehtinen, T. O. A., Mäntylä, M. V., & Vanhanen, J. (2011). Development and evaluation of a lightweight root cause analysis method (ARCA method) – Field studies at four software companies. *Information and Software Technology, 53*(10), 1045–1061.

Lehtinen, T. O. A., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived causes of software project failures – An analysis of their relationships. *Information and Software Technology, 56*(6), 623–643.

Leigh, E. (2013). Simulation for Project Management Research. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 199-219). Universitetsforlaget: Copenhagen Business School Press.

Leonardi, P. M. (2011). When Flexible Routines Meet Flexible Technologies: Affordance, Constraint, and the Imbrication of Human and Material Agencies. *MIS Quarterly, 35*(1), 147-167.

Leonardi, P. M. (2013). Theoretical foundations for the study of sociomateriality. *Information and Organization, 23*(2), 59–76.

Leszak, M. (2006). Extending Empirical Studies to Cover More Realistic Industrial Development and Project Management Issues. In V. R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, & R. Selby (Eds.), *Empirical Software Engineering Issues: Critical Assessment and Future Directions* (pp. 131). International Workshop, Dagstuhl Castle, Germany: Springer-Verlag Berlin Heidelberg.

Lincoln, Y. S., & Guba, E. G. (2000). The only Generalization is: There is no Generalization. In R. Gomm, M. Hammersley, & P. Foster (Eds.), *Case Study Method: Key Issues, key Texts* (pp. 27-44). UK: SAGE Publications Ltd.

Lipke, W. (2003). Schedule is Different. *The Measurable News, 31,* 31-34.

Lipke, W., Zwikael, O., Henderson, K., & Anbari, F. (2009). Prediction of project outcome: The application of statistical methods to earned value management and earned schedule performance indexes. *International Journal of Project Management*, *27*(4), 400–407.

Locke, K. (2001). *Grounded Theory in Management Research*. London: Sage Publications Ltd.

Lockyer, K., & Gordon, J. (2005). *Project Management and Project Network Techniques* (Seventh edition ed.). UK: Prentice Hall.

Lopes, E. (2010). *A Grounded Theory of Decision-Making under Uncertainty and Complexity*. UK: VDM Verlag Dr. Muller Aktiengesellschaft & Co. KG.

Lu, X., Liu, H., & Ye, W. (2010). *Analysis failure factors for small & medium software projects based on PLS method*. Paper presented at the 2010 The 2nd IEEE International Conference on Information Management and Engineering (ICIME), 2010, Chengdu.

Lyneisa, J. M., & Ford, D. N. (2007). System dynamics applied to project management: a survey, assessment, and directions for future research. *System Dynamics Review, 23*(2/3), 157–189.

Mahring, M., Holmstrom, J., Keil, M., & Montealegre, R. (2004). Trojan actor-networks and swift translation: Bringing actor-network theory to IT project escalation studies. *Information Technology & People*, *17*(2), 210-238.

Malcolm, B. (1990). A Large Embedded System Project Case Study. In B. A. Kitchenham (Ed.), *Software Engineering for Large Software Systems* (pp. 96-121): Elsevier Applied Sciences.

Malone, T. W., & Crowston, K. (1994). The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, *26*(1), 87-119.

Maxwell, J. A. (1992). Understanding and validity in qualitative research. *Harvard educational review, 62*(3), 279-301.

McDermid, J. A. (1990). Integrated Project Support Environments: General principles and issues in the development of high integrity systems. In B. A. Kitchenham (Ed.), *Software Engineering for Large Software Systems* (pp. 27-81). UK: Elsevier Applied Sciences.

McGrath, K. (2002). The Golden Circle: a way of arguing and acting about technology in the London Ambulance Service. *European Journal of Information Systems, 11*(4), 251-266.

McLean, C., & Hassard, J. (2004). Symmetrical Absence/Symmetrical Absurdity: Critical Notes on the Production of Actor-Network Accounts. *Journal of Management Studies, 41*(3), 493-519.

McLeod, L., & MacDonell, S. G. (2011). Factors that affect software systems development project outcomes: A survey of research. *ACM Computing Surveys, 43*(4), Article No. 24.

Middleton, P., & Joyce, D. (2012). Lean Software Management: BBC Worldwide Case Study. *IEEE Transactions on Engineering Management, 59*(1), 20-32.

Miles, M. B., & Huberman, A. M. (1994). *An Expanded Sourcebook: Qualitative Data Analysis* (Second Edition ed.). USA: SAGE Publications, Inc.

Milosevic, D., & Patanakul, P. (2005). Standardized project management may increase development projects success. *International Journal of Project Management,* *23*(3), 181–192.

Mingers, J. (2001). Combining IS Research Methods: Towards a Pluralist Methodology. *Information Systems Research, 12*(3), 240–259.

Mingers, J., & Willcocks, L. (2014). An integrative semiotic framework for information systems: The social, personal and material worlds. *Information and Organization*, *24*(1), 48–70.

Mitchell, J. C. (2000). Case and Situation Analysis. In R. Gomm, M. Hammersley, & P. Foster (Eds.), *Case Study Method: Key Issues, key Texts* (pp. 165-186). UK: SAGE Publications Ltd.

Miyazaki, Y., Takanou, A., Nozaki, H., Nakagawa, N., & Okada, K. (1991). Method to estimate parameter values in software prediction models. *Information and Software Technology, 33*(3), 239-243.

Mockus, A., & Herbsleb, J. (2001). *Challenges of Global Software Development*. Paper presented at the Proceedings of the Seventh International Symposium on Software Metrics, 2001. METRICS 2001., London.

Moløkken-Østvold, K., & Jørgensen, M. (September 2005). A Comparison of Software Project Overruns—Flexible versus Sequential Development Models. *IEEE Transactions on Software Engineering, 31*(9), 754-766.

Monteiro, E. (2001). Actor-Network Theory and Information Infrastructure. In Ciborra & Associates (Eds.), *From Control to Drift: The Dynamics of Corporate Information Infrastructure* (pp. 71-83). Oxford: Oxford University Press.

Monteiro, E. (2004). Actor network theory and cultural aspects of interpretive studies. In C. Avgerou, C. Ciborra, & F. Land (Eds.), *The Social Study of Information and Communication Technology: Innovation, Actors, and Contexts* (pp. 129-139). USA: Oxford University Press Inc.

Montoni, M. A., & Rocha, A. R. (2010). *Applying Grounded Theory to Understand Software Process Improvement Implementation*. Paper presented at the 2010 Seventh International Conference on the Quality of Information and Communications Technology (QUATIC), Porto.

Morris, P. W. G. (2002). Science, objective knowledge, and the theory of project management. *Civil Engineering, 150*(2), 82-90.

Morris, P. W. G., & Jamieson, A. (2005). Moving from corporate strategy to project strategy. *Project Management Journal, 36*(4), 5-18.

Morris, P. W. G., Jamieson, A., & Shepherd, M. M. (2006). Research updating the APM Body of Knowledge 4th edition. *International Journal of Project Management, 24*(6), 461–473.

Muller, R., Sankaran, S., & Drouin, N. (2013). Introduction. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 19-30). Universitetsforlaget: Copenhagen Business School Press.

Münch, J. (2006). Effective Data Interpretation. In V. R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, & R. Selby (Eds.), *Empirical Software Engineering Issues: Critical Assessment and Future Directions* (pp. 83-90). International Workshop, Dagstuhl Castle, Germany: Springer-Verlag Berlin Heidelberg.

Nandhakumar, J. (1996). Design for success?: critical success factors in executive information systems development. *European Journal of Information Systems, 5*(1), 62-72.

Nasir, M. H. N., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. *Scientific Research and Essays, 6*(10), 2174-2186.

Nelson, R. R. (2007). IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices. *MIS Quarterly Executive, 6*(2), 67-78.

Nguyen, T., Wolf, T., & Damian, D. (2008). *Global software development and delay: Does distance still matter?* Paper presented at the IEEE International Conference on Global Software Engineering, 2008. ICGSE 2008, Bangalore.

Nidiffer, K. E., & Dolan, D. (2005). Evolving Distributed Project Management. *IEEE Software, 22,* 63-72.

Nijland, M. H.-J. (2004). *Understanding the Use of IT Evaluation Methods in Organisations.* (PhD), University of London, United Kingdom.

O'Cathain, A. (2010). Assessing the Quality of Mixed Methods Research: Toward a Comprehensive Framework. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 531-555). USA: SAGE Publications, Inc.

OGC. (2009). *Managing Successful Projects with PRINCE2* (2009 Edition ed.). UK: Office of Government Commerce Stationery Office Books.

Onwuegbuzie, A. J., & Combs, J. P. (2010). Emergent Data Analysis Techniques in Mixed Methods Research: A Synthesis. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 397-430). USA: SAGE Publications, Inc.

Onwuegbuzie, A. J., & Johnson, R. B. (2006). The Validity Issue in Mixed Research. *Research in the Schools, 13*(1), 48-63.

Oorschot, K. v. (2013). System Dynamics for Project Management Research. In N.

Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 220-236). Universitetsforlaget: Copenhagen Business School Press.

Orlikowski, W. J. (1993). CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly, 17*(3), 309-340.

Orlikowski, W. J. (2009). The Sociomateriality of Organizational Life: Considering Technology in Management Research. *Cambridge Journal of Economics, 34*(bep058), 125-141.

O'Cathain, A., Murphy, E., & Nicholl, J. (2008). The Quality of Mixed Methods Studies in Health Services Research. *Journal of Health Services Research & Policy, 13*(2), 92–98.

Patanakul, P. (2014). Managing large-scale IS/IT projects in the public sector: Problems and causes leading to poor performance. *Journal of High Technology Management Research, 25*(1), 21–35.

Pernstål, J., Feldt, R., & Gorschek, T. (2013). The lean gap: A review of lean approaches to large-scale software systems development. *The Journal of Systems and Software, 86*(11), 2797– 2821.

Petersen, K., Khurum, M., & Angelis, L. (2014). Reasons for bottlenecks in very large-scale system of systems development. *Information and Software Technology, 56*(10), 1403–1420.

Phan, D. D., Vogel, D. R., & Jay F. Nunamaker, J. (1995). Empirical studies in software development projects: Field survey and OS/400 study. *Information & Management*, *28*(4), 271-280.

Piri, A., & Niinimäki, T. (2011). *Does distribution make any difference? Quantitative comparison of collocated and globally distributed projects*. Paper presented at the 2011 Sixth IEEE International Conference on Global Software Engineering Workshops, Helsinki, Finland.

PMI. (2008). *A Guide to the Project Management Body of Knowledge* (Fourth Edition ed.). USA: Project Management Institute Inc.

Poppendieck, M. (2007). *Lean Software Development*. Paper presented at the 29th International Conference on Software Engineering - Companion, 2007. ICSE 2007 Companion, Minneapolis, MN.

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach* (Sixth Edition ed.). USA: McGraw-Hill.

Procaccino, J. D., & Verner, J. M. (2006). Software project managers and project success: An exploratory study. *The Journal of Systems and Software, 79*(11), 1541–1551.

Procaccino, J. D., Verner, J. M., & Lorenzet, S. J. (2006). Defining and contributing to software development success. *Communications of the ACM, 49,* 79-83.

Qureshi, S., Liu, M., & Vogel, D. (2004). *A Grounded Theory Analysis of E-Collaboration Effects for Distributed Project Management*. Retrieved from Netherlands:

Rabbi, M. F., & Mannan, K. O. B. (2008). *A Review of Software Risk Management for Selection of best Tools and Techniques*. Paper presented at the Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08, Phuket.

Rainer, A. (2010). Representing the behaviour of software projects using multi-dimensional timelines. *Information and Software Technology, 52*(11), 1217-1228.

Rainer, A. (2011). The Longitudinal, Chronological Case Study Research Strategy: A Definition, and an Example from IBM Hursley Park. *Information and Software Technology, 53*(7), 730-746.

Rainer, A. W. (1999). *An empirical investigation of software project schedule behaviour.* (Doctor of Philosophy), Bournemouth University, United Kingdom.

Ramasubbu, N., & Balan, R. K. (2007). *Globally Distributed Software Development Project Performance: An Empirical Analysis.* Paper presented at the ESEC-FSE '07 Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, Dubrovnik, Croatia.

Raz, T., Barnes, R., & Dvir, D. (2003). A Critical Look at Critical Chain Project Management. *Project Management Journal, 34*(4), 24-32.

Redmill, F. (1992). *Incremental Delivery - not all Plain Sailing*. Paper presented at the IEE Colloquium on Software Prototyping and Evolutionary Development, London.

Remenyi, D. (2013). *Case Study Research* (2nd Edition ed.). UK: Academic Conferences and Publishing International Limited.

Robertson, S., & Williams, T. (2006). Understanding Project Failure: Using Cognitive Mapping in an Insurance Project. *Project Management Journal, 37*(4), 55-71.

Robson, C. (2011). *Real World Research* (Third edition ed.). UK: Wiley.

Rook, P. (1986). Controlling software projects. *Software Engineering Journal, 1*(1), 7-

16.

Rose, J., Pedersen, K., Hosbond, J. H., & Kræmmergaard, P. (2007). Management competences, not tools and techniques: A grounded examination of software project management at WM-data. *Information and Software Technology, 49*(6), 605–624.

Royce, W. W. (1970). *Managing the development of large software systems.* Paper presented at the IEEE Wescon.

Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 31-164.

Sage, D., Dainty, A., & Brookes, N. (2011). How actor-network theories can help in understanding project complexities. *International Journal of Managing Projects in Business, 4*(2), 274-293.

Sankaran, S., Muller, R., & Drouin, N. (2013). Creative and Innovative Contemporary Approaches. In N. Drouin, R. Muller, & S. Sankaran (Eds.), *Novel Approaches to Organizational Project Management Research: Translational and Transformational* (Vol. 29, pp. 162-163). Universitetsforlaget: Copenhagen Business School Press.

Sauer, C., Gemino, A., & Reich, B. H. (2007). The Impact of Size and Volatility on IT Project Performance: Studying the Factors Influencing Project Risk. *Communications of the ACM, 50*(11), 79-84.

Schneberger, S. L., & McLean, E. R. (2003). The Complexity Cross—Implications for Practice. *Communications of the ACM, 46*, 216-225.

Schonberger, R. J. (1981). Why Projects are "Always" Late: A Rationale Based on Manual Simulation of a PERT/CPM Network. *Interfaces, 11*(5), 66-70.

Schön, D. A. (1983). *The Reflective Practitioner: How Professionals Think in Action*. UK: Basic Books.

Scott, J. E., & Vessey, I. (2002). Managing risks in enterprise systems implementations. *Communications of the ACM, 45*, 74-81.

Seaman, C. B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering, 25*(4), 557-572.

Shahzad, B., & Al-Mudimigh, A. S. (2010). *Risk Identification, Mitigation and Avoidance Model for Handling Software Risk*. Paper presented at the 2010 Second International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), Liverpool.

Shahzad, B., Ullah, I., & Khan, N. (2009). *Software Risk Identification and Mitigation in Incremental Model*. Paper presented at the International Conference on Information and Multimedia Technology, 2009. ICIMT '09., Jeju Island.

Shermer, C. L., Neighbors, M. A., & Maitland, R. (1981). *Systems engineering for medium to large software development projects*. Paper presented at the Conference Proceedings Southeastcon '81, Huntsville, AL, USA.

Silva, F. Q. B. d., Costa, C., C., A. C., & Prikladinicki, R. (2010). *Challenges and Solutions in Distributed Software Development Project Management: a Systematic Literature Review.* Paper presented at the 5th IEEE International Conference on Global Software Engineering (ICGSE), 2010, Princeton, NJ.

Sjøberg, D. I. K., Dybå, T., & Jørgensen, M. (2007). *The Future of Empirical Methods in Software Engineering Research.* Paper presented at the Future of Software Engineering, 2007. FOSE '07, Minneapolis, MN.

Sommerville, I. (2011). *Software Engineering* (Ninth Edition ed.). USA: Pearson.

Staats, B. R., Brunner, D. J., & Upton, D. M. (2011). Lean principles, learning, and knowledge work: Evidence from a software services provider. *Journal of Operations Management, 29*(5), 376–390.

Stake, R. E. (2000). The Case Study Method in Social Inquiry. In R. Gomm, M. Hammersley, & P. Foster (Eds.), *Case Study Method: Key Issues, key Texts* (pp. 19-26). UK: SAGE Publications Ltd.

Stapleton, J. (1997). *Dynamics Systems Development Method: the method in practice*. England: Addison-Wesley.

Star, S. L. (1991). Power, technology and the phenomenology of conventions: on being allergic to onions. In J. Law (Ed.), *A Sociology of Monsters: Essays on Power, Technology and Domination* (pp. 26-56). London: Routledge.

Stensrud, E., Foss, T., Kitchenham, B., & Myrtveit, I. (2002). *An Empirical Validation of the Relationship Between the Magnitude of Relative Error and Project Size.* Paper presented at the Proceedings of the Eighth IEEE Symposium on Software Metrics (METRICS'02).

Stratton, R. (2009). *Critical Chain Project Management Theory and Practice*. Paper presented at the POMS 20th Annual Conference, Orlando, Florida, USA.

Sun, S. (2009). *Study on Software Project Risk Priority Management and Framework Based on Information Management System*. Paper presented at the 1st International Conference on Information Science and Engineering (ICISE), 2009, Nanjing.

Söderlund, J. (2004). Building theories of project management: past research, questions for the future. *International Journal of Project Management, 22*(3), 183–191.

Tarawneh, M. d. M. I., AL-Tarawneh, H., & Elsheikh, A. (2008). *Software Development Projects: An Investigation into the Factors that Affect Software Project Success/ Failure in Jordanian Firms*. Paper presented at the First International Conference on the Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008., Ostrava.

Tashakkori, A., & Teddlie, C. (1998). *Mixed Methodology: Combining Qualitative and Quantitative Approaches* (Vol. 46). USA: SAGE Publications.

Tashakkori, A., & Teddlie, C. (2008). Quality of Inferences in Mixed Methods Research: Calling for an Integrative Framework. In M. M. Bergman (Ed.), *Advances in Mixed Methods Research* (pp. 101-119). London: Sage Publications Ltd.

Tashakkori, A., & Teddlie, C. (2010). Current Developments and Emerging Trends in Integrated Research Methodology. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 803-826). USA: SAGE Publications, Inc.

Tate, G., & Verner, J. (1990). Case study of risk management, incremental development, and evolutionary prototyping. *Information and Software Technology, 32*(3), 207-214.

Taylor, H. (2006). Risk Management and Problem Resolution Strategies for IT Projects. *Project Management Journal, 35*(5), 49-63.

Teddlie, C., & Tashakkori, A. (2010). Overview of Contemporary Issues in Mixed Methods Research. In A. Tashakkori & C. Teddlie (Eds.), *SAGE Handbook of Mixed Methods in Social & Behavioral Research* (Second Edition ed., pp. 1-41). USA: SAGE Publications, Inc.

Trietsch, D. (2005). Why a Critical Path by any other name would smell less sweet? Towards a holistic approach to PERT/CPM. *Project Management Journal, 36*(1), 27-36.

Urbaczewski, L., & Mrdalj, S. (2006). A comparison of enterprise architecture frameworks. *Issues in Information Systems, 7*(2), 18-23.

Urquhart, C. (2007). The Evolving Nature of Grounded Theory Method: The Case of the Information Systems Discipline. In A. Bryant & K. Charmaz (Eds.), *The SAGE Handbook of Grounded Theory* (pp. 339-359). London: SAGE Publication Ltd.

Urquhart, C. (2013). *Grounded Theory for Qualitative Research*. UK: SAGE Publications Ltd.

Urquhart, C., Lehmann, H., & Myers, M. D. (2010). Putting the 'theory' back into grounded theory: guidelines for grounded theory studies in information systems. *Information Systems Journal, 20*(4), 357–381.

Vandevoorde, S., & Vanhoucke, M. (2006). A comparison of different project duration forecasting methods using earned value metrics. *International Journal of Project Management, 24*(4), 289–302.

Venkatesh, V., Brown, S. A., & Bala, H. (2013). Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in information systems. *MIS Quarterly, 37*(1), 21-54.

Verner, J., Sampson, J., & Cerpa, N. (2008). *What factors lead to software project failure?* Paper presented at the Second International Conference on Research Challenges in Information Science, 2008. RCIS 2008., Marrakech.

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niaz, M. (2014). Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology, 56*(1), 54–78.

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2012a). *Risk Mitigation Advice for Global Software Development from Systematic Literature Reviews*. Retrieved from Staffordshire, UK:

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2012b). *Systematic literature reviews in global software development: a tertiary study.* Paper presented at the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), Cuidad Real, Spain.

Verner, J. M., Overmyer, S. P., & McCain, K. W. (1999). In the 25 years since The Mythical Man-Month what have we learned about project management? *Information and Software Technology, 41*(14), 1021–1026.

W.Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *Computer, 21*(5), 61-72.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of information systems, 4*(2), 74-81.

Walsham, G. (1997). *Actor-network theory and IS research: current status and future prospects.* Paper presented at the Proceedings of the IFIP TC8 WG 8.2 International Conference on Information Systems and Qualitative Research, 31st May–3rd June 1997, Philadelphia, Pennsylvania, USA.

Walsham, G., & Sahay, S. (1999). GIS for district-level administration in India: problems and opportunities. *MIS quarterly, 23*(1), 39-66.

Williams, K. T. G. (2005). Modelling complexity in the automotive industry supply chain. *Journal of Manufacturing Technology Management, 16*(4), 447 - 458.

Williams, R., Ambrose, K., & Bentrem, L. (2004a). *A Roadmap of Risk Diagnostic Methods: Developing an Integrated View of Risk Identification and Analysis Techniques*. Retrieved from USA:

Williams, R. C., Ambrose, K., Bentrem, L., & Merendino, T. (2004b). *Risk Based Diagnostics*. Retrieved from USA:

Winner, L. (1993). Upon opening the black box and finding it empty: Social constructivism and the philosophy of technology. *Science, Technology, and Human Values, 18*(3), 362-378.

Winter, M., Smith, C., Morris, P., & Cicmil, S. (2006). Directions for future research in project management: The main findings of a UK government-funded research network. *International Journal of Project Management, 24*(8), 638–649.

Winter, R., & Fischer, R. (2006). *Essential Layers, Artifacts, and Dependencies of Enterprise Architecture.* Paper presented at the 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06), Hong Kong, China.

Yamamoto, S., Ibe, K., Verner, J., Cox, K., & Bleistein, S. (2009). *Actor relationship analysis for the i* framework.* Paper presented at the Proceedings of the 11th International Conference on Enterprise Information Systems, ICEIS 2009, Milan, Italy.

Yeo, K. T. (2002). Critical failure factors in information system projects. *International Journal of Project Management, 20*(3), 241–246.

Yin, R. K. (2009). *Case Study Research: Design and Methods* (Fourth edition ed. Vol. 5). USA: SAGE Publications.

Zhou, L., Vasconcelos, A., & Nunes, M. (2008). Supporting decision making in risk management through an evidence-based information systems project risk checklist. *Information Management & Computer Security, 16*(2), 166-186.

Zmud, R. W. (1980). Management of large software development efforts. *MIS Quarterly, 4*(2), 45-55.

Šmite, D., Wohlin, C., Gorschek, T., & Feldt, R. (2010). Empirical evidence in global software engineering: a systematic review. *Empirical Software Engineering, 15*(1), 91–118.