# Exposing the Influencing Factors on Software Project Delay with Actor-Network Theory

**Zana Ahmedshareef, Robert Hughes and Miltos Petridis**
**University of Brighton, Brighton, UK**
Z.Ahmedshareef@brighton.ac.uk
R.T.Hughes@brighton.ac.uk
M.Petridis@brighton.ac.uk

**Abstract:** Managing large software projects through global development teams is a complex undertaking; it involves managing interdependent problems and dynamic situations that are constantly changing. 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. This research investigates the interdependent influences exerted on project progress that emerge during project execution and cause schedule delay. This paper aims to demonstrate the value of integrating different research methods and techniques from the technical and social domains in order to address such complexity, in particular the utility of actor-network theory (ANT) to expose the influencing factors on project schedule delay. The research approach (Mixed method) was applied to empirical data from a global software provider, integrating quantitative analysis (project metrics) with qualitative analysis (grounded theory) and culminating in the development of an explanation model (ANT). The findings demonstrate that considerable benefit can be gained from the fuller understanding of the management dynamics during project execution provided by this approach. ANT makes researchers look at the networks of influence at play between human and nonhuman elements of the project, thus offering a richer picture of the project.

**Keywords:** Research methods, Actor-network theory, Grounded theory, Project management, Information systems development, Software development

## 1. Introduction

Russell Ackoff (1979) describes the challenge facing project managers with the words ‘Managers are not confronted with problems that are independent of each other, but with dynamic situations that consist of complex systems of changing problems that interact with each other’ (page 99). Moreover, Donald Schön (1983) characterises the environment of practice with: ‘complexity, uncertainty, instability, uniqueness, and value conflict’ (page: 18). Today, the practice of software project management is no different. In this environment, delivering large software systems through global development teams increases these difficulties.

A challenge confronting software project management research and practice alike is producing practical solutions based on empirical data, while being hampered by ‘the split between industry practice and academic research’ (Jacobson et al. 2012). The research and professional bodies of knowledge have widely contributed to our understanding of the various areas of software project management. However, there seems still more to be learned about the interplay that emerges during actual project execution and the way it influences schedule duration. This research follows (Ralph, 2013) in arguing that more than one method or theory is needed to address the interdependent areas of software project management and that any such study should be based on empirical data to have practical relevance. The purpose of this research is to identify the causes of schedule delay in software projects. It will also illustrate the benefits of integrating various methods and theories, in particular the value of ANT, in illuminating the influences on schedule delay.

The research data used are past project progress reports created by and for project participants. They came from a global company (named ABC for anonymity) that develops software through globally distributed teams using a version of the Iterative and Incremental development method within a software factory model. ABC has an institutionalised software project management function, adopting industry standard frameworks, tools and techniques, blended with decades of practice. It is a typical major systems provider in the industry, hence, the interest in investigating its work practices to understand the causes of delivery delays.

The research adopted the mixed method approach to enquiry employing explanatory sequential design. It integrated actor-network theory, grounded theory, and project metrics to make sense of the interdependent factors of software project execution. The findings demonstrate the value of combining different approaches, in particular ANT in explaining project delay. ANT influences the way research is done. It makes researchers look at the context of a research object more carefully. This is because it focuses on networks of influence rather than 'simple' input-output models. These networks are heterogeneous, i.e. the nodes can be of many different types (e.g. human and nonhuman) and the nature of links (intermediaries) can also be very varied.

The questions relevant to the particular concern of this paper were:
RQ1 - To what extent can actor-network theory provide a useful model of the interactions between the various actors involved in a software development environment?
RQ2 - Is the application of actor-network theory compatible with a mixed methods research approach?
RQ3 - Is the application of actor-network theory compatible with a grounded theory approach?

## 2. Actor-network theory

This section outlines some ANT notions (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. As we will see, ANT is appropriate for studying the 'shape and fate of technological projects' (Law & Callon 1992, page: 46) and (Hughes, 2014, page: 186).

ANT has emerged from the social study of science and technology and attempts to make sense of the dynamics at play among disparate elements with varying degrees of flexibility. While ANT is applicable to many social settings, it is particularly suitable in explaining project behaviour. ANT focuses on the interactions occurring among the *actors* who collaborate to achieve some goal, and in doing so create an *actor-network* (Law, 2012). Project execution occurs in dynamic situations that consist of complex interactions among heterogeneous entities - (Law, 2012) calls this *heterogeneous engineering*. Hence, ANT offers the ability to describe whether the net of all interactions among these entities supports achieving the objective of the project.

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 1 a group of actors (including Design, Build and Test managers) work to perform a software development task using intermediaries (such as a functional specification and design defect reports) to coordinate their activities. Actors and intermediaries can be human or nonhuman. An example of the later 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* can itself become an *actor*; for example a software component under construction can have errors (code defects), the correction of which absorbs effort and causes delays. Thus, *actors* can be seen as elements of a project that interact through *intermediaries*.

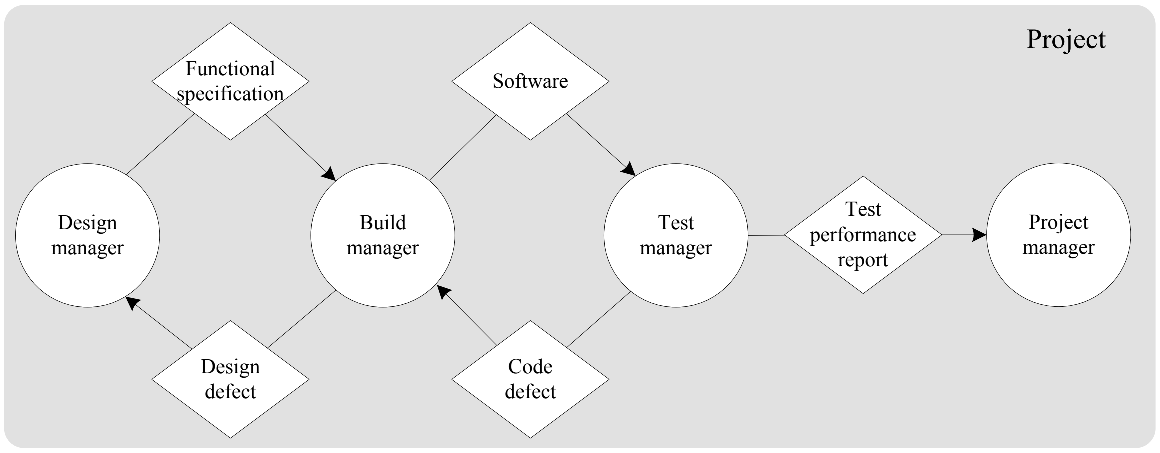


**Figure 1:** Software development process

A typical software development process (Figure 1) includes the Design, Build, and Test phases. The Design manager (*actor*) delivers functional specification (*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 specification. 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.

Some elements of an *actor-network (i.e. the actors, intermediaries, and their interaction)* can be thought of as a *black-box*. In ANT terminology, a *black-box* is an artefact that embodies a number of elements (which itself would be a *network*) where their internal interaction is concealed from the outside world. An outsider interacts only with the artefact’s external features but not with its internal constituents (Monteiro, 2001). For example, software testers may be interested in the external behaviour of a software component and not in its internal workings. They will treat the internal structure as a ‘black-box’ and will simply check that the inputs and outputs conform to the functional design. An actor-network can be very large and complicated and external actors may try to make their relationships with it easier by treating parts of the actor-network as a black-box. A similar ploy is to identify a major individual who 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 mechanism for embedding programs of action in technical artefacts (e.g. the functional specification in Figure 1), with the aim of guiding the artefact user to operate in a certain way, is called *inscription* in ANT terms. A *weakly inscribed* program of action weakens the *irreversibility* of an *actor-network*. *Irreversibility* in ANT refers to the degree of stability in an established *actor-network* and its resistance to going back and changing things already done. A *strong inscription* resists reversibility attempts (Monteiro, 2001). For example, requirements informally described by the client may be *weakly inscribed* during the Design and lead to reversibility at the Build and Test phase 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. 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.

ANT can be applied to non-projects (or even ‘pre-projects’) as well as projects. An ANT study can examine the construction of a *network* 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. The focal actor may attempt to establish a network and *mobilise* the *actors* within it to achieve particular purpose. This process is called *translation* in ANT (Callon, 1986). An ANT study can also investigate the operation of an already established actor-network (project); examining the interactions among the *actors* and *intermediaries* which are well understood and accepted, this is called *network dynamics*. Callon (1991) refers to network dynamics as ‘the complex process in which actors and their talkative (sometimes indiscreet) intermediaries weave themselves together’ (page: 144). For example, Figure 2 shows ABC’s version of the iterative and incremental development (IID) method, where a project consists of multiple increments. An increment represents one development cycle comprising the three Design, Build, and Test (DBT) *phases* which deliver a portion of the software functionality. 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.

The vertical arrow ‘Project execution’ shows the increment cycle, where work flows across phases; from design to build to test. The horizontal arrow ‘Project progress’ shows work moving to the next increment of functionality within a particular phase; for example, following completion of design work in increment 1; the team starts to design increment 2 of the functionality and so on, the same principle being 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 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 the interactions among project actors (Callon 1991, page: 152). The *network dynamics* ought to be supportive of achieving the network objectives if the *actor-network* is to succeed.

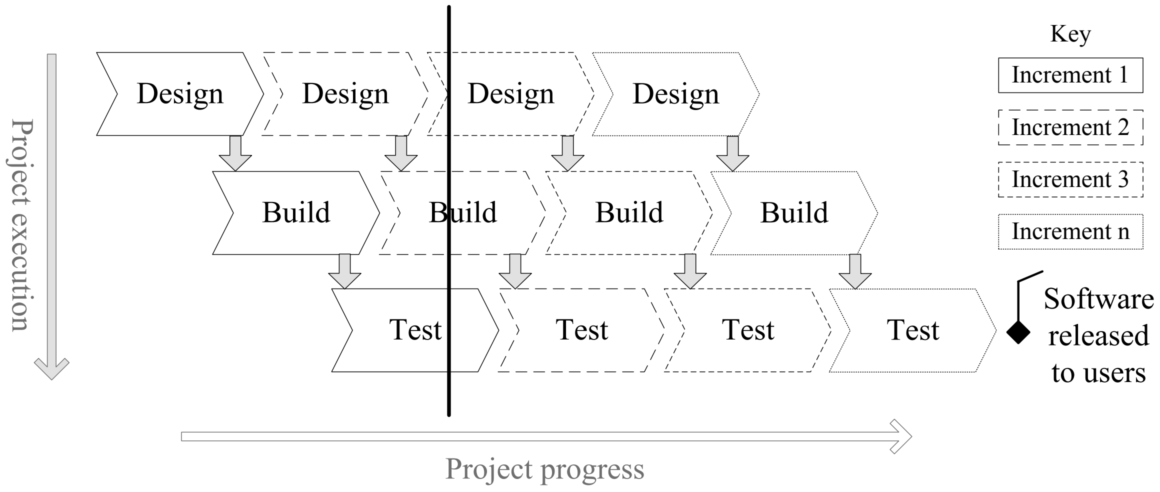


**Figure 2:** ABC's software engineering method (adapted from Ahmedshareef et al. 2013)

*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 Test manager of Increment 1 may attempt to maintain the Test phase's progress on schedule through developing agreement with the Build and Design managers. A *weakly aligned actor-network* is one which the actors' commitment is not guaranteed. This may be due to the actor being unable to commit to their role in the network (perhaps due to competing priorities), rather than not wanting to (though the latter is possible too). A weakly aligned *actor-network* exerts constraining influences on achieving the network objectives. Conversely, a *strongly aligned actor-network* is one which the actors remain committed to their role in the network, which exerts empowering influences on achieving network objectives.

*Coordination* in ANT refers to the extent to which a network is governed by rules inscribed in the interaction among the actors, aiming to stabilise the actor-network (Callon 1991, page: 146-147). For example, the Build manager (in Figure 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 timeline (rule) was agreed to provide such fixes to the Test manager of Increment 1. A network with no adoption of *rules* 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/*networks* above results in *weak coordination*. As will be seen that managing dependencies and defining and enforcing rules will become particularly challenging during product transition from the preceding *network* to the succeeding one; i.e. Design to Build to Test - see Figure 1. 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. 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 outcome of an ANT analysis can be a description, model, or explanation of the area being investigated (McLean & Hassard, 2004), aiming to 'learn from the actors' (Latour, 1999) through tracing of associations (Latour 2005, page: 8, Underwood 2014, page: 357) by following project management activities (Callon 2012, page: 92) rather than imposing existing frameworks. This, as Akrich et al. (2002) put it, helps to 'render the mechanisms of success and failure intelligible and ultimately more manageable' (page: 191). Table 1 summarises the ANT concepts applied on the case study along with possible indicators of what might be a failing actor-network.

**Table 1:** Summary of ANT concepts and possible indicators of problematic project

| ANT concept | Description | Possible indicators of problematic project |
|---|---|---|
| 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. |
| Intermediary | An element within the network of associations that facilitates interaction between actors; they are the relationships/associations the actors forge to enable interaction. | 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. |
| Actor-network | A network is formed as actors interact through intermediaries to achieve some goal. An actor-network is the actors, their intermediaries, and the interactions taking place. | The project's behaviour cannot be known or predicted to some degree (i.e. not black-boxed - see next). |
| Black-box | An element that embodies a number of elements (which itself would be a network), where their internal interaction is concealed from the outside world. | The project is not black-boxed. An effective black-box may go unnoticed. When a black box makes its presence known then that usually means it is malfunctioning and demands management action e.g. when a power supply is cut off. |
| 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 weakening irreversibility (see next). A 'strong' program of action is not just one that is detailed and enforceable. It needs to be widely accepted - i.e. contribute to alignment. |
| Irreversibility | The degree of stability of an actor-network and its resistance to going back and changing things that have already been done. | Unstable project; prevalence of disorder in project activities; disruption in producing deliverables. |
| Network dynamics | The complex processes in which actors and intermediaries entangle and interact to achieve a particular purpose; the dynamics ought to be supportive of achieving network objectives. | The interactions are constraining the achievement of the overall project objectives. |
| Alignment | The degree of agreement between the actors on, and their commitment to, their role in the network. | Weakly aligned: actors are not committed/unable to commit to their role in achieving the project objectives; which exerts constraining influences on achieving project objectives. |
| Coordination | The extent to which a network is governed by rules inscribed in the interaction among the actors, aiming to stabilise the actor-network. | Weak coordination: project rules are weakly inscribed in the interaction among the actors; rules are not widely accepted by project actors, which exert constraining influences on achieving project objectives. |

## 3. Research approach

The study integrated actor-network theory, grounded theory, and project metrics within mixed method approach (Creswell & Clark, 2011) to obtain a fuller understanding of what might be happening during project execution - see Figure 3.

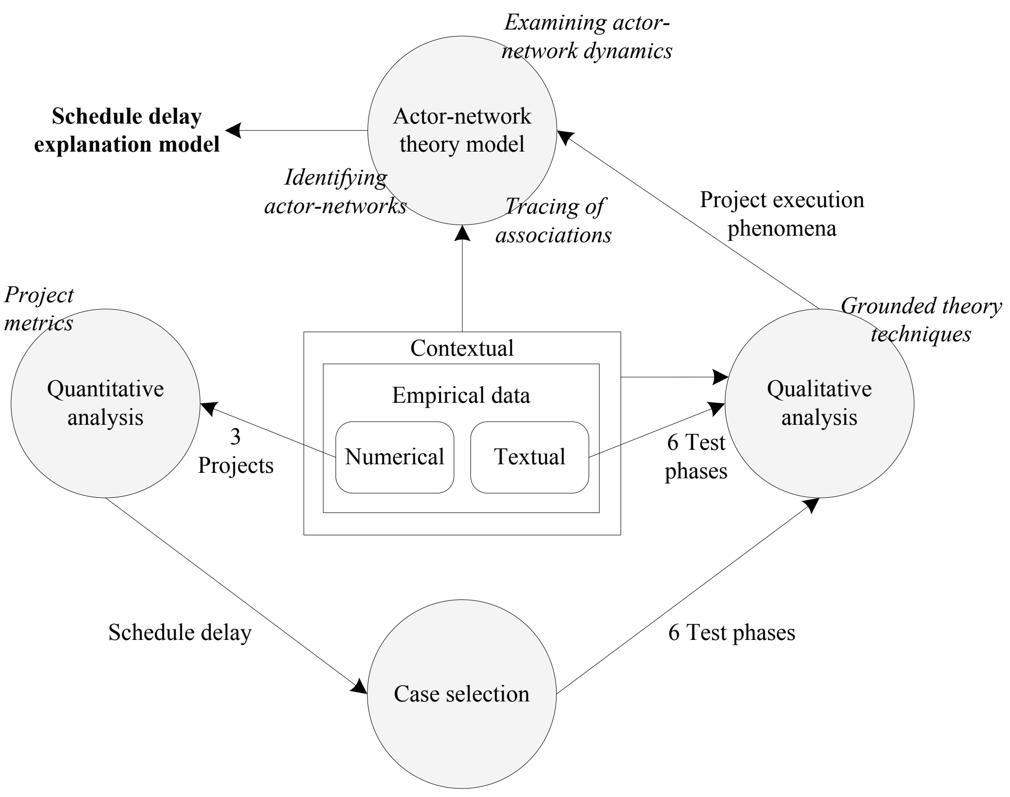


**Figure 3:** Research approach

The research data were project performance reports, produced and used by project participants, comprising numeric and textual data. This was supplemented with the contextual information obtained from members of ABC which explained the terminology used in the reports. The numeric data of three projects were analysed to determine extent of delay in the projects and their component phases. Six Test phases were identified as contributing most to schedule delay. The textual data of the reports for these phases were analysed qualitatively using grounded theory (GT) techniques. This resulted in the emergence of categories of phenomena present during the execution of the Test phases. An explanation model was then developed of schedule delay using actor-network theory (ANT) by tracing of associations among the instances of the GT categories, identifying actor-networks, and examining the network dynamics.

ANT was an appropriate technique for analysing the influences on a project resulting from the interactions among actors and intermediaries. However, where project performance was measured quantitatively then metrics-based approach was introduced to complement ANT. The metrics were mapped to the elements identified in the ANT analysis. The numeric progress data had corresponding text reflecting the reasons for the performance results perceived by the project participants. Grounded theory was an appropriate way of analysing the text. The approach is illustrated next through a case example.

## 3.1 Case example

Project 1 in ABC produces weekly performance reports (comprising numeric and text) for management meetings to track progress of work activities against plan. This research analysed the numeric data, which resulted in the project metrics shown in Figure 4 and Table 2 below.

Figure 4 illustrates that various project phases/actor-networks interacted to develop software over time. The functional design (FD) actor-network inscribed the client’s requirements into design specifications. The build actor-networks were responsible for FD/TD Transition, TD (Technical design), and Code which inscribed the design specifications into code (i.e. executable computer programs). Similarly, the test actor-network was responsible for AT (Assembly Test) and IT (Integration Test) which executed the code to test their operation. Figure 4 also shows that some activities in the planned schedule which had ‘Finish-to-start’ relationships with the following dependent activities in fact overlapped when executed. A Finish-to-start project relationship refers to the situation where the succeeding activity may not start until the preceding activity has finished.

Thus, the project management rule of starting a phase only when the preceding one was completed had clearly been flouted causing coordination and alignment complications.

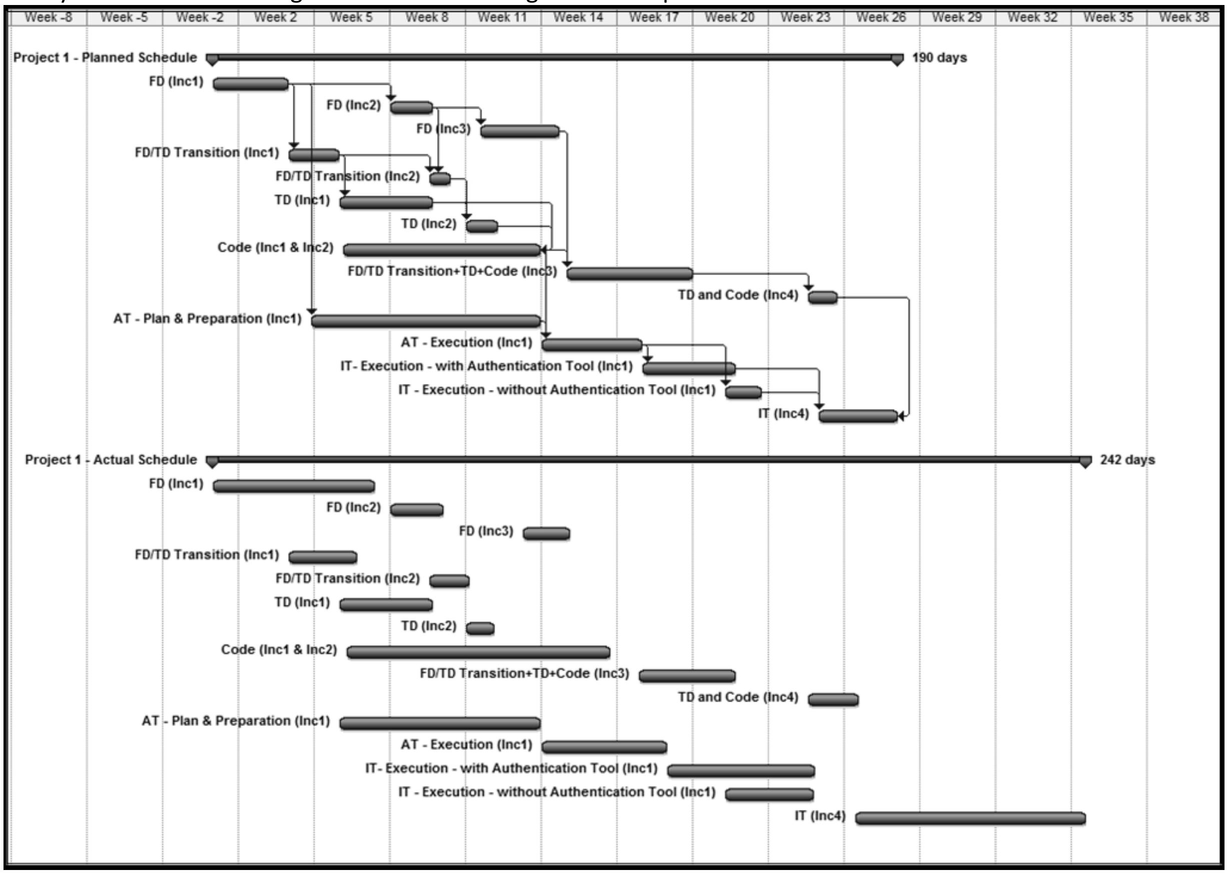


**Figure 4**: Project 1 Gantt chart

**Table 2:** Project 1 delay metrics

| # | **Project 1 Phases** | **Planned start** (day) | **Actual start** (day) | **Start variance** (actual start - planned start**)** | **Planned finish** (day) | **Actual finish** (day) | **Finish variance** (actual finish - planned finish) | **Change in duration** (Finish variance - Start variance) |
|---|---|---|---|---|---|---|---|---|
| 1 | FD (Inc1) | 0 | 0 | 0 | 20 | 44 | 24 | 24 |
| 2 | FD (Inc2) | 49 | 49 | 0 | 60 | 63 | 3 | 3 |
| 3 | FD (Inc3) | 74 | 86 | 12 | 95 | 98 | 3 | -9 |
| 4 | FD/TD Transition (Inc1) | 21 | 21 | 0 | 34 | 39 | 5 | 5 |
| 5 | FD/TD Transition (Inc2) | 60 | 60 | 0 | 65 | 70 | 5 | 5 |
| 6 | TD (Inc1) | 35 | 35 | 0 | 60 | 60 | 0 | 0 |
| 7 | TD (Inc2) | 70 | 70 | 0 | 78 | 77 | -1 | -1 |
| 8 | Code (Inc1 & Inc2) | 36 | 37 | 1 | 90 | 109 | 19 | 18 |
| 9 | FD/TD Transition+TD+Code (Inc3) | 98 | 118 | 20 | 132 | 144 | 12 | -8 |
| 10 | TD and Code (Inc4) | 167 | 165 | -2 | 174 | 178 | 4 | 6 |
| 11 | AT - Plan & Preparation (Inc1) | 27 | 35 | 8 | 90 | 90 | 0 | -8 |
| 12 | AT - Execution (Inc1) | 91 | 91 | 0 | 118 | 125 | 7 | 7 |
| 13 | IT- Execution - with Authentication Tool (Inc1) | 119 | 126 | 7 | 144 | 166 | 22 | 15 |
| 14 | IT - Execution - without Authentication Tool (Inc1) | 142 | 142 | 0 | 151 | 166 | 15 | 15 |
| 15 | IT (Inc4) | 169 | 179 | 10 | 190 | 242 | 52 | 42 |
| **Project delay** (Start vriance for phase #15+ Change in duration for phase #15) = | | | | | | | | **52 (days)** |

Table 2 shows the degree of delay for each of the project phases on the Gantt chart in Figure 4. The last column (Change in duration) shows the number of days which the particular phase was delayed (positive value) or completed earlier (negative value). The phase with the largest delay was the Integration Test phase (#15) that took 42 days more than planned; and as the phase had started 10 days later than scheduled (see Start variance column for #15), this resulted in an overall project delay of 52 days. In ABC, the key indicator of project success was delivering on schedule.

The project metrics identified the phases where schedule was delayed. However, they did not explain why progress was behind. Since the largest delay was in the Integration Test phase in row #15, research further examined the textual data of the Integration Test phases in the project.

The study used the techniques of grounded theory (GT) to code the textual data of the Test phase reports and to categorise the phenomena present during project execution - see Figure 5 for the GT process. In utilising the GT techniques, this research did not seek to generate theory. Urquhart (2013) supported decision to use GT for other than theory generation.

The following GT analyses were carried out:

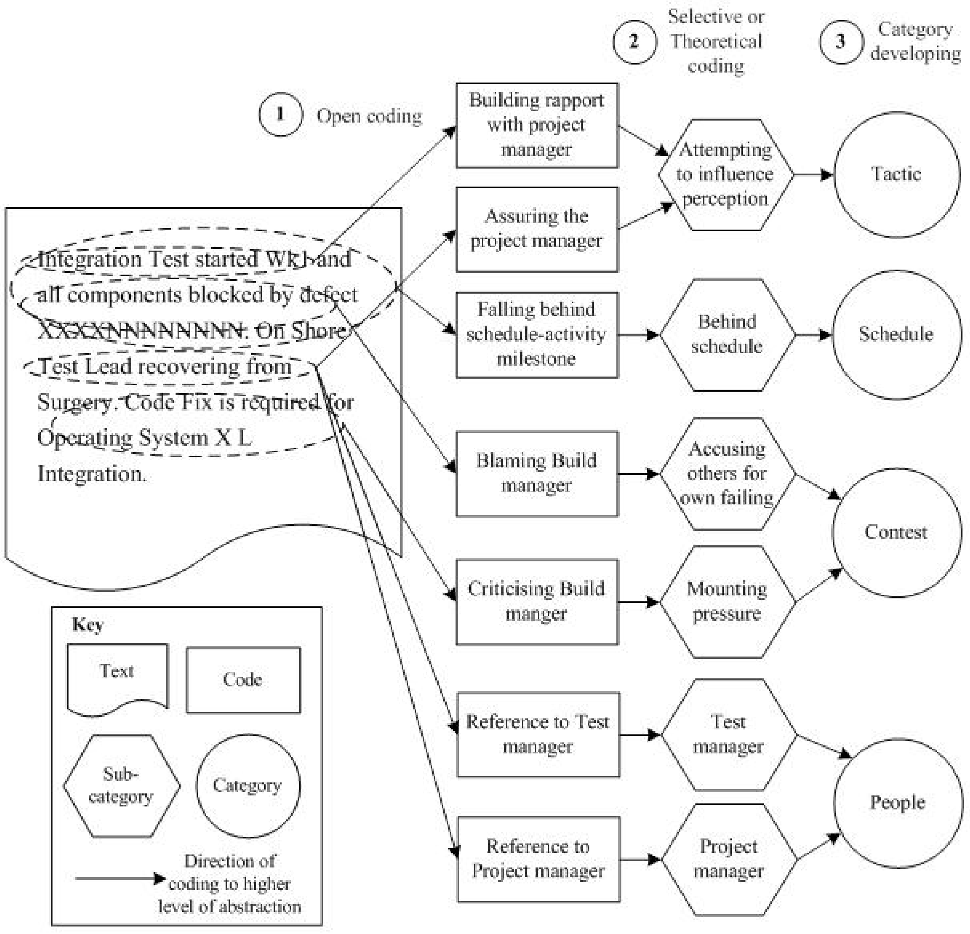

1 Open coding
2 Selective or Theoretical coding
3 Category developing
Integration Test started Wk1 and all components blocked by defect XXXXNNNNNNNN. On Shore Test Lead recovering from Surgery. Code Fix is required for Operating System X L Integration.
Building rapport with project manager
Assuring the project manager
Falling behind schedule-activity milestone
Blaming Build manager
Criticising Build manger
Reference to Test manager
Reference to Project manager
Attempting to influence perception
Behind schedule
Accusing others for own failing
Mounting pressure
Test manager
Project manager
Tactic
Schedule
Contest
People
Key
Text
Code
Sub-category
Category
Direction of coding to higher level of abstraction

**Figure 5:** Coding of the textual data - Project 1

1) Open coding - segments of text were labelled (coded) to summarise what was happening in the statement. A statement could be coded more than once.
2) Selective or Theoretical coding - the Open codes that represent similar themes were grouped into sub-categories, in which two types emerged: Selective and Theoretical. Selective code represents entity; Theoretical code represents relationship between entities.
3) Category developing - the sub-categories that represent similar concepts were grouped into categories.

The categories that emerged from coding the textual information in the subset of Project 1 performance reports relating to the Test phases are shown in Table 3. These relate to the reasons given for the delays in testing identified in Table 2.

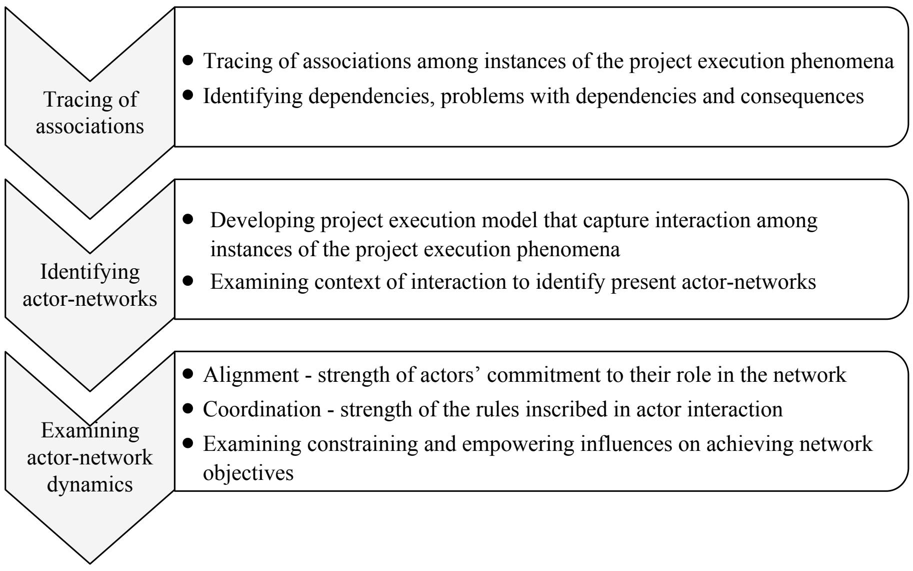
**Table 3:** Project 1 execution phenomena

| # | Category | Description | Selected example/instance |
|---|---|---|---|
| 1 | Contest | A contested environment where phase managers attempt to protect their professional reputation in front of the Project | Accusing others for own failing; Causes; |

| # | Category | Description | Selected example/instance |
|---|---|---|---|
| | | manager when actual work progress falls behind schedule. | Mounting pressure |
| 2 | Feature | The characteristics of a state or situation; for example, Code exhibiting defect. | Depending on; Exhibiting |
| 3 | People | Human participants in the project who manage project phases; representing their teams who are responsible for the day to day activities to deliver phase products and services. | Project manager; Build manager; Test manager; Design manager |
| 4 | Product | Project artefacts used and produced by phase managers; for example software Code, a product, is produced by the Build manager to be used by the Test manager. | Code; Performance report; Design |
| 5 | Quality | The expected standard of products delivered by phase managers. | Defect |
| 6 | Schedule | The means by which project events are ordered on a timeline showing what event is planned to be achieved and by when. | Behind schedule |
| 7 | Tactic | Techniques used by the Test manager to influence the Project manager on a particular aspect. | Attempting to influence perception; Portraying powerlessness |
| 8 | Turbulence | Exhibiting fluctuation in the normal course of project execution. | Contributing uncertainty; Grappling with time |
| 9 | Undertaking | Carrying out tasks related to phase execution; for example Test manager testing code. | Delivering; Using; Testing; Resolving; Registering |

## 4. An ANT model of project execution

In order to make sense of the project execution phenomena emerged from applying the grounded theory techniques (Table 3), the study developed an explanatory model by means of the three steps in Figure 6.



Tracing of associations
Tracing of associations among instances of the project execution phenomena
Identifying dependencies, problems with dependencies and consequences
Identifying actor-networks
Developing project execution model that capture interaction among instances of the project execution phenomena
Examining context of interaction to identify present actor-networks
Examining actor-network dynamics
Alignment - strength of actors' commitment to their role in the network
Coordination - strength of the rules inscribed in actor interaction
Examining constraining and empowering influences on achieving network objectives

**Figure 6:** ANT explanatory model of project execution

Figure 6 shows that each step in developing the model builds on the preceding step. These are elaborated in the following sections.

### 4.1 Tracing of associations

The narrative in the textual data of Project 1's Test progress reports, supplemented by contextual information, were used to trace the associations present among the example categories of the project execution phenomena (Table 3 - last column) during the execution of the Test phases. Earlier, Table 2 identified the Test phases as most contributing to project delay, Table 3 then identifies the phenomena present during the execution of these Test phases: consequently, the analysis below focuses on the Test manager as the focal actor in the network. This carries with it the implication of reflecting the Test manager's perspective on the actor-network, though not in isolation since it was discussed in the presence of the other actors. Nonetheless, the study is aware that the other actors' perspectives on their interaction in the network may differ.

To maintain the Test execution progress on schedule, the Test manager (TM) was dependent on various resources (i.e. constituent parts of the project) outside the Test phase, such as the developed code, the functional design, and the Build manager (BM). The TM was also dependent on the Project manager (PM) for support in setting priorities for other actors which maintained the Test execution progress on schedule.

Tracing the resources reveal that they were created and circulated by other phase managers to support the TM. The BM needed to deliver the code and, since operating as Fix manager during the Test phase execution, needed to provide timely resolutions of defects in the developed code. The Design manager (DM) needed to deliver the design and to clarify design specifications. Issues with these elements, such as the discovery of defects in the project products (in this case, developed code and functional Design) caused the Test execution progress to fall behind schedule.

Although the TM registered the defects, delays in Test progress contributed to uncertainty about the future progress of the Test phase, which were made worse by the BM not clarifying when the issues would be resolved and the DM delaying providing functional knowledge of the specifications. The TM blamed the other phase managers for failing to maintain Test execution progress on schedule. The TM also mounted pressure on the other phase managers for quick resolution by portraying himself as powerless to the PM for having no influence over them.

The TM communicated lack of progress to the PM through weekly performance reports, and attempted to influence the PM's perception on the lack of progress as being caused by factors outside their control. Furthermore, the TM conveyed to the PM that they were under time pressure as they had to deal with multiple issues simultaneously. The PM used the performance reports to resolve the above obstacles/issues by negotiating changes in priorities with the BM and DM.

### 4.2 Identifying actor-networks

The associations that emerged through tracing the instances of the project execution phenomena during the execution of the Test phases can be represented diagrammatically - see Figure 7. This uses the categories and instances that were identified by the GT approach to analysing the text of progress reports. To develop an ANT model is to model the interaction among the actors in the network which could also be a graphical representation of a network comprising sequences of points and lines (Callon 2012, page: 90). However, at this stage of analysis, the model we have is only a representation of the contents of textual data in the Test performance reports (the GT analysis), rather than the underlying physical system, and therefore it cannot be assumed to encapsulate all the contextual information surrounding the project.

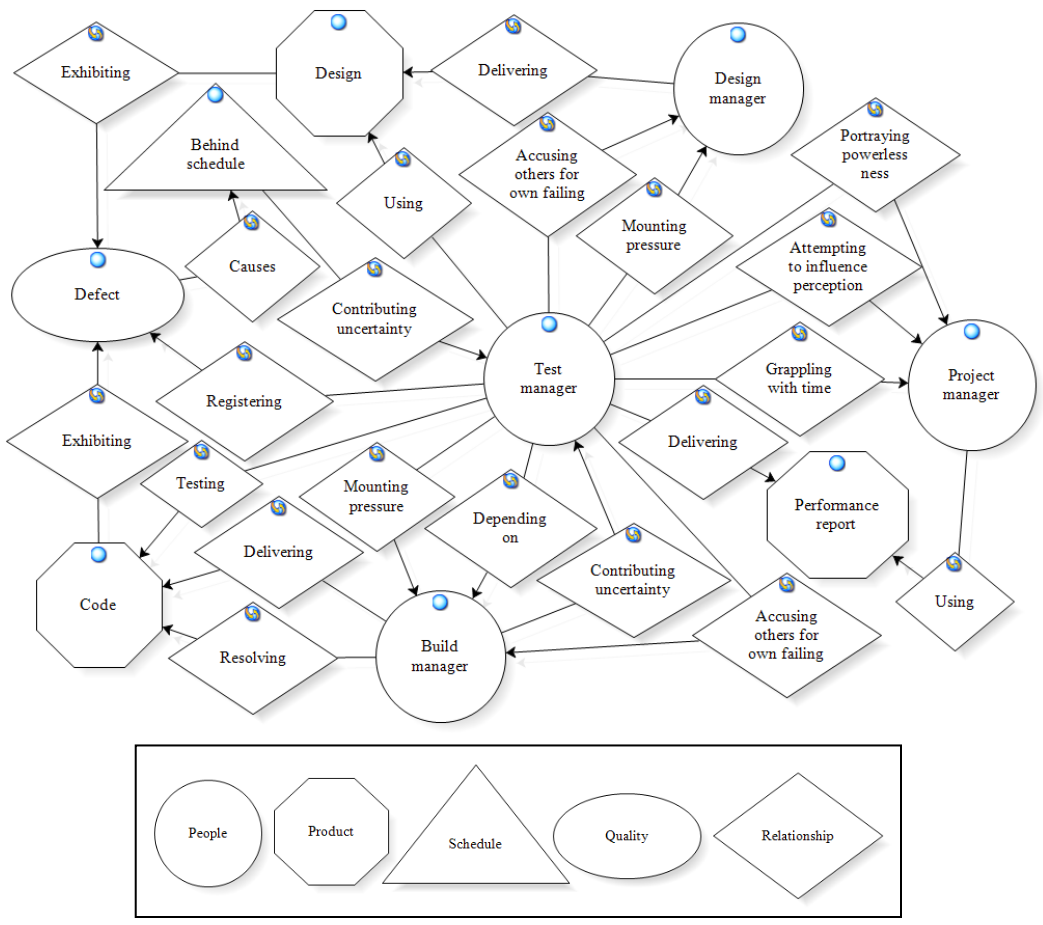


**Figure 7:** Project execution model

Figure 7 illustrates that varied phenomena are in interaction during the execution of the Test phases which are shown as nodes - people, product, schedule, and quality - and links that connect the nodes - contest, feature, tactic, turbulence, and undertaking. These phenomena correspond to the example categories of the project execution phenomena (Table 3). This suggests that the project can be reinterpreted 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 be seen as intermediaries. The model that came out of the 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.

Using the project's contextual information, the context of interaction during the execution of the Test phases was examined in the model, which indicates the presence of the following actor-networks: Test actor-network (TAN), Build actor-network (BAN), Design actor-network (DAN), and Project manager (PM). Recall that an actor in ANT usually represents a network of actors. The TM for example, among other things, speaks on behalf of a team of testers. The context of interaction between TAN and each of BAN, DAN, and PM is described next.

The TAN, represented by the Test manager (TM) in the model, comprised the TM and their team. The main objective of TAN is to maintain the progress of the Test phase execution on schedule, provided that the tested software conforms to the specification. Therefore, delays in the Test execution schedule was seen as failing to deliver on the plan by the TAN, the consequences which put the TM's competency into question (i.e. whether they are capable of managing a Test phase). The TM reports progress of the Test phase execution to the Project manager (PM) in the weekly performance meetings, highlighting any obstacles facing progress and seeking support from the PM to remove these obstacles.

The BAN, represented by the Build manager (BM) in the model, comprised the BM and their team. Part of the BAN is expected to provide code fix for the increment being tested during the Test execution phase in a timely manner. The other part of BAN continues developing the subsequent increment/s of the one being tested by the TAN (see Figure 2). The BM's main focus is to maintain the progress of the increment under development on schedule, for the same competency reasons described earlier (i.e. a factor in assessing the performance of phase managers is their ability to deliver on schedule in ABC).

The DAN, represented by the Design manager (DM) in the model, comprised the DM and their team. The DAN is expected to provide support to TAN in timely manner, during the Test phase, in the form of functional knowledge of the software specification. This support helps the TAN to prepare Test data and Test scenarios that conform to the specification and the way business users would use the final system. The DAN's main focus is designing the subsequent increment (to preserve their performance reputation) to the one being tested, whilst providing support to TAN in the increment being tested (see Figure 2).

The PM manages the overall project. The PM also has line manager responsibility over each of the TAN, BAN, and DAN. The PM helps the TAN in monitoring and controlling the Test phase execution and is expected to remove obstacles in the way of maintaining progress on schedule by negotiating changes in priorities with BAN and DAN using the weekly phase performance reports. The PM's interaction with BAN and DAN occurs during the performance meetings where specific actions are negotiated to control deviation from the Test phase schedule. However these interactions are not recorded in the Test performance reports, which the model reflects, and hence did not emerge in the original model.

Figure 7 illustrates the complexity of interdependent problems and various actor-networks interacting with each other, but contain gaps that indicate that the factors influencing schedule delay can be implicit, exerting their influence through means other than explicit interaction – for example, the BAN may have been unable to provide quick resolution of code defects to TAN, due to their focus on maintaining the progress, of the increment under development, on schedule (i.e. competing priorities). These influences are examined next in more depth.

## 4.3 Examining actor-network dynamics

The study now examines the dynamics of TAN with each of BAN, DAN, and PM to identify what may have caused schedule delay. TAN is the focal actor of interest to this study, because the Test phases were most contributing to project delay. The analysis was done by applying an ANT interpretation to the project execution model (Figure 7) through the concepts of network dynamics: alignment and coordination to ascertain the extent to which the dynamics of these actor-networks support the achievement of project objectives in maintaining the progress of the Test phases on schedule.

### *4.3.1 Alignment*

The project execution model (Figure 7) shows that the software Code exhibits defects, which indicate weak inscription of the design specification into the code, leading to reversibility of the affected Code from the Test phase back to the Build phase. The model also shows the TAN blaming BAN for delays in the Test execution progress - see the relationship 'Accusing others for own failing' associating BAN with TAN (i.e. failing to deliver on the Test execution plan), which indicates weak commitment of BAN to their role in supporting TAN on a timely manner. In addition, the association 'Depending on' linking TAN with BAN in the model indicates dependency of TAN on BAN to providing timely fix of defects. The relationship 'Contributing to uncertainty' could also be interpreted as indicating weak commitment from BAN to support TAN through not clarifying when the registered defects would be resolved. Finally, the extracts from the progress reports below, reported in the Test performance reports, showing the volume of discovered Code defects the BAN had to resolve, may indicate why BAN may appear not being able to commit to their role. The BAN may want to be committed to quick resolution of these defects but actually unable to do so because of the difficulties of the task (this is an illustration of the Code within the BAN as a nonhuman actor having influence on the project). Thus, the actor-network of TAN-BAN exerts constraining influences on achieving the project objectives in maintaining the progress of the Test phase on schedule.

Case_03: N_03: '*13 new Defects have been raised*'

Case_22: N_10: '*54 Defects have been raised in total*'
Case_22: N_11: '*64 Defects have been raised in total*'
Case_22: N_13: '*94 Defects have been raised in total*'

In relation to alignment of DAN, the project execution model shows that the Design exhibits defects, which indicates weak inscription of software requirements into the Design documents, and leading to reversibility of the related Code from the Build phase to the Design phase. The model also shows that TAN is blaming DAN for not being able to maintain the progress of the Test phase on schedule through the relationship 'Accusing others for own failing', which indicates weak commitment from DAN to their role in supporting TAN on a timely manner. The example empirical data below indicates that the TAN is seeking the help of the PM to make DAN provide functional knowledge in time. The actor-network of TAN-DAN, therefore, exerts constraining influences on maintaining progress on schedule.

Case_02: N_3: '*Outstanding points are on Inc3 scenarios, functional knowledge needed to create input message.*'

The alignment of PM and TAN in the project execution model shows a different picture compared to the preceding ones; it indicates strong alignment. The model shows the associations 'Portraying powerlessness' and 'Attempting to influence perception' that link TAN to PM, which indicate strong alignment because they get the PM to negotiate priorities with BAN and DAN in order to maintain the Test phase progress on schedule. Furthermore, the example empirical data below indicates the removal of obstacles on the way of Test phase progress from one week to the next, which is supported by the contextual information that the PM was negotiating reprioritising of activities with BAN and DAN, to bring the deviation of the Test progress back on schedule. Thus, the actor-network of TAN-PM exerts empowering influences on maintaining progress on schedule.

Case_01: N_01: '*Integration Test started Wk1 and all components blocked by defect XXXXNNNNNNNN.*'
Case_01: N_02: '*Integration Test started Wk1 and making good progress*'

### *4.3.2 Coordination*

The model shows the relationship 'Mounting pressure' from TAN on BAN, indicating weak inscription of rules governing their interaction. The inscribed rule, though may originally be accepted by BAN to fix code defects in a timely manner, may be constrained by the volume of work or difficulty of the task involved. Furthermore, the rule may only be an implicit expectation rather than explicit script. The TAN-BAN coordination, hence, exerts constraining influences on the Test phase progress. A similar situation can be observed between DAN and TAN, as the relationship 'Mounting pressure' from TAN to DAN indicates weak inscription of rules governing their interaction. That is, DAN being totally focused on delivering their increment on schedule, may conflict with any inscribed rules requiring them to provide functional knowledge to TAN on the earlier increment in a timely manner. The example empirical data shown in the earlier section indicates that the weak inscription of rules required escalation to the PM to obtain such commitment. This exerts constraining influences on the progress of the Test phase.

Moreover, Figure 4 (the Gantt chart) showed the violation of the project management rule of 'Start-to-Finish'. This indicates weak coordination in the overall project network. These problems appear to have been caused by the use of parallel incremental delivery (Figure 2), where project phases started earlier than scheduled, with the approval of the PM, so that the impact of delay in preceding phases is reduced on the subsequent phases. The consequence, however, seems to be that priority is given to delivering the next increment on schedule, whilst supporting earlier increments became a second priority. That is, weak coordination leads to weak alignment, which then exerts constraining influences on the progress. In addition, the exclusive focus on minimising project delay, in ABC, appears to have ignored the fact that there are other objectives e.g. the detection and removal of defects. In theory, the TAN could be successful in reducing testing time by reducing the quality of the end product.

In contrast, coordination between TAN-PM is strong through the circulation of Test phase performance reports by the Test manager, and the PM 'Using' the reports to removing obstacles on the way of progress.

Furthermore, the relationship 'Grappling with time' indicates strength of this interaction such that the PM understands lack of/slow progress due to internal circumstance to the TAN, since TAN is dealing with multiple issues simultaneously.

Thus, the complexity of the dynamics emerged in the preceding analyses reveals that, TAN is struggling to maintain the Test phase progress on schedule because it has limited control over schedule duration. It is the BAN and DAN who, to a large extent, exercise that control and unfortunately are the ones who exert constraining influences on project progress, eventually leading to schedule delay.

This section has demonstrated that, through ANT it was possible to explain how the dynamics that developed during project execution influenced schedule delay; as Law (2012) put it ' *the point…is to discover the pattern of forces as these revealed in the collisions that occur between different types of elements*' (page: 108).

## 5. Concluding remarks

The purpose of this research was to identify causes of software project delay, arguing that managing the execution of large software projects involves not only managing technology, people, processes, environments, and contexts; but also managing interdependent problems and dynamic situations that are constantly changing; and that research ought to match that complexity to produce practical solutions through, but not limited to, integrating different methods and techniques (Coleman & O'Connor, 2007) and investigating the interdependent aspects of managing such projects. Previous work has commended such approach in 'providing insight into software engineering behaviour' (Ralph, 2013). This paper sought to answer the following questions:

RQ1 - To what extent can actor-network theory provide a useful model of the interactions between the various actors involved in a software development environment?

The value of ANT in 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 in this study, it would have been difficult to identify causes of schedule delay in such a complex project environment. In addition, what the ANT analysis exposed contradicts the conventional assumption that the project/phase manager 'owns' and controls all the resources allocated to the phase. In fact, the system forces the phase manager to interact with other actors outside their control but who constrain the progress of the phase. Thus, there appear to be a basic flaw in the management system which makes the person responsible for an operation (Test phase) over which they have little power to control. ANT enabled investigating the interdependent areas, and dynamics, of software project management through its demand on the researcher to attend to the context of the research object more carefully.

RQ2 - Is the application of actor-network theory compatible with a mixed methods research approach?
It can be said that this paper demonstrated that applying ANT within mixed methods approach is a step towards obtaining a fuller understanding of the complexity of what is happing during software project execution. Applied to different types of data, the techniques used (ANT, GT, and project metrics) complement one another identifying the causes of schedule delay. We argue that the contribution of combining ANT and mixed methods in illuminating the influence of actor (human and nonhuman) dynamics on schedule delay cannot be overlooked in software project management research. It is worth noting that other studies have integrated ANT with various research methods successfully. For example; Hanseth et al. (2006) investigated a project of developing electronic patient record system in Norwegian hospital, combining ANT with complexity theory and reflexivity theory to make sense of the complex dynamics inherent in implementing software projects; and Greenhalgh & Stones (2010) combined ANT with the strong structuration theory to develop theoretical perspectives and understand what happens in developing large IT systems in the UK's NHS.

RQ3 - Is the application of actor-network theory compatible with a grounded theory approach?
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 latent patterns, 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 interpret the subject domain. Thus,

ANT and GT can be seen to complement one another, the GT analysis generating phenomena, whilst the ANT concepts 'glue' the phenomena through illuminating the influence of interactions among them. However, evolving GT categories into an ANT model required the development of intermediary analysis steps for this study involving the creation of an explanation model to facilitate the transition (Figure 6). Applying ANT to GT categories is not completely new; for example, Lopes (2010) used ANT to enhance and elaborate the categories that emerged from a GT analysis, and to explore nonhuman relationships in the process of making decisions under uncertainty and complexity (page: 52).

Finally, the integration of ANT, GT, and project metrics in this study to make sense of the complexity of managing software project execution raises fresh questions on the challenges facing management to resolve competing organisational and project priorities, physical resource clashes, and the escalation processes to resolve such problems - questions that create fertile ground for future research.

## References

Ackoff, R. L. (1979). "The Future of Operational Research is Past." *Journal of the Operational Research Society* 30(2): 93-104.

Akrich, M., M. Callon and B. Latour (2002). "The Key to Success in Innovation Part I: The Art of Interessement." International Journal of Innovation Management 6(2): 187–206.

Callon, M. (1986). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. *Power, action and belief: a new sociology of knowledge?* J. Law. London, Routledge: 196-223.

Callon, M. (1991). Techno-economic Networks and Irreversibility. *A Sociology of Monsters: Essays on Power, Technology and Domination*. J. Law. London, Routledge: 132-161.

Callon, M. (2012). Society in the Making: The Study of Technology as a Tool for Sociological Analysis. The Social Construction of Technological Systems. W. E. Bijker, T. P. Hughes and T. Pinch, Massachusetts Institute of Technology: 77-97.

Coleman, G. and R. O'Connor (2007). "Using grounded theory to understand software process improvement: A study of Irish software product companies." *Information and Software Technology* 49(6): 654–667.

Creswell, J. W. and V. L. P. Clark (2011). *Designing and Conducting Mixed Methods Research*. USA, SAGE Publications Inc.

Greenhalgh, T. and R. Stones (2010). "Theorising big IT programmes in healthcare: Strong structuration theory meets actor-network theory." Social Science & Medicine 70: 1285–1294.

Hanseth, O., E. Jacucci, M. Grisot and M. Aanestad (2006). "Reflexive Standardization: Side Effects and Complexity in Standard Making." MIS Quarterly 30(Special issue): 563-581.

Hughes, R. (2014). Bridging the gap Between the Social and the Technological With Actor-Networks. 13th European Conference on Research Methodology for Business and Management, London.

Jacobson, I., B. Meyer and R. Soley. (2012). "Software Engineering Method and Theory – A Vision Statement." Retrieved 17 Oct, 2013.

Latour, B. (1999). On Recalling ANT. *Actor Network Theory and after*. J. Law and J. Hassard. UK, Blackwell Publishers: 15-25.

Latour, B. (2005). *Reassembling the Social - An Introduction to Actor-Network-Theory*. USA, Oxford University Press.

Law, J. (2012). Technology and Heterogeneous Engineering: The Case of Portuguese Expansion. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. W. E. Bijker, T. P. Hughes and T. Pinch. USA, MIT: 105-127.

Law, J. and M. Callon (1992). The Life and Death of an Aircraft: A Network Analysis of Technical Change. *Shaping Technology/Building Society: Studies in Sociotechnical Change*. W. E. Bijker and J. Law. USA, MIT Press.

Lopes, E. (2010). A Grounded Theory of Decision-Making under Uncertainty and Complexity, VDM Verlag Dr. Muller Aktiengesellschaft & Co. KG.

McLean, C. and J. Hassard (2004). "Symmetrical Absence/Symmetrical Absurdity: Critical Notes on the Production of Actor-Network Accounts." Journal of Management Studies 41(3): 493-519.

Monteiro, E. (2001). Actor-Network Theory and Information Infrastructure. *From Control to Drift: The Dynamics of Corporate Information Infrastructure*. Ciborra and Associates. Oxford, Oxford University Press: 71-83.

Ralph, P. (2013). 'Possible Core Theories for Software Engineering'. *2013 2nd SEMAT Workshop on a General Theory of Software Engineering (GTSE)*. San Francisco, CA, USA, IEEE Xplore.

Schön, D. A. (1983). *The Reflective Practitioner: How Professionals Think in Action*, Basic Books.

Underwood, J. (2014). The use and Usefulness of Actor-Network Theory as a Basis for Social Research: A Consideration of Some Recent Publications. 13th European Conference on Research Methodology for Business and Management, London.

Urquhart, C. (2013). *Grounded Theory for Qualitative Research*. UK, SAGE Publications.