# A Requirements Engineering Framework For Developing COTS GIS Applications

A thesis presented in partial fulfilment of the requirements for the degree of

Master Of Information Science

In

Information Systems

At Massey University, Palmerston North,
New Zealand

Adrienne Kim Bonnington

2002

# ABSTRACT

There has been an increase in recent years in the number of Geographic Information System (GIS) applications being developed for stakeholders using Commercial Off-The-Shelf (COTS) software. There are a lack of guidelines in both industry and the literature on how to acquire user requirements for the development of GIS applications in this COTS software environment.

This study investigates process activities in order to build a framework to address these inadequacies. The construction of the framework incorporates requirements engineering and COTS software evaluation and selection process activities from the Information Systems (IS) area. The framework is used to explore three issues related to developing GIS applications and used to determine whether: 1) a life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications, 2) standard IS requirements processes can be used for developing GIS solutions, and 3) standard IS COTS software acquisition processes can be used for developing GIS solutions. Case studies were used to analyse current practices in the GIS industry and to validate the usefulness of these activities in the framework.

The results of this investigation suggest that RE practices associated with the COTS paradigm within the IS arena are suitable for developing GIS applications based on user requirements.

# ACKNOWLEDGEMENTS

A special thanks to the four participants from the two organisations used for the multiple-case study. I know time is scarce and is an expensive commodity, so I truly appreciate the effort you all gave in contributing towards this research.

Thank you to my supervisor, Mrs Elisabeth Todd for her patience and the several cups of coffee to make light work of research meetings. I would also like to thank Mr Arthur Todd and Mr Larry Haist for helping me 'fill in the gaps', and to keep my research rigorous. Their advice has been invaluable.

I would like to thank ESRI for giving me the opportunity to work in their professional services section as an intern and gaining wonderful friendships through this experience. I would also like to express my gratitude to Massey University for the financial support of getting me to the USA to carry out the internship.

Most of all I wish to thank my parents who have always been incredibly supportive of my postgraduate studies, particularly my mother for having the patience of proof reading this thesis for me. Also, to my husband, who has been extremely positive and encouraging, and has always been there through my moments of confusion.

# PUBLICATIONS

The following refereed paper was based on part of the work in this thesis.

Bonnington, A.K., & Todd, E.G. (2001). Which GIS Product Should I Use? Proceedings from the 29th Annual Conference of AURISA, Melbourne, Australia.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER I

# INTRODUCTION

> *"Ease of use is critical, and only the user's tasks should matter; systems should reflect the needs of users, without requiring them to be concerned with technicalities"* (Goodchild, Egenhofer, Kemp, David & Sheppard, 1999, p.740)

## OVERVIEW

The geographic information system (GIS) application developer needs a rigorous strategy to be able to reflect successfully the user's (stakeholder's) needs in a system development project. In the development of any information system (IS), the process that focuses on *the needs of users* and hence their tasks (Goodchild *et al.*, 1999) is requirements engineering (RE). The primary outcome of RE is the requirements specification which "should tell a designer everything he needs to know to satisfy all the users - but nothing more" (Somerville, 1989, in Sutcliffe & Jarke, 1996, p.515). Software RE has been defined as "all the activities devoted to identification of user requirements, analysis of the requirements to drive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against the actual user needs" (Saiedian & Dale, 2000, p 420).

GIS applications are complex due to their reliance on spatial data; this differentiates them from standard business IS's. Both GIS and IS products are used to input, manipulate, query, analyse, output and communicate digital data, thereby illustrating that the requirements of both systems are similar. According to literature in the IS field the development of a successful project requires good RE practices (Davis & Leffingwell, 1995; Kosters, Pagel, Six, 1995; Maiden & Rugg, 1996; Somerville, 1989, in Sutcliffe & Jarke, 1996; Dorfman, 1997; Carney, 1999; Saiedian & Dale, 2000); therefore, so do GIS applications.

> *"...many of the standards and guidelines for general computer systems and information management are applicable to a GIS" (Guptill, 1989, p. 1585).*

Currently, there are many spatial GIS applications being developed using GIS commercial off-the-shelf (COTS) software products. With these GIS COTS products, developers can create tailored applications that meet GIS users needs. A well-defined decision making technique is required for evaluating GIS COTS products, so that products selected will best meet the requirements of the GIS user. There is a lack of literature in the GIS field discussing the evaluation and selection of GIS COTS software. Therefore, an investigation was required to determine whether RE practices associated with the COTS paradigm as reported in the IS arena would be suitable for developing GIS applications based on user requirements.

## THE RESEARCH PROBLEM

There is a lack of literature and guidelines in practice to develop GIS applications using COTS software. The main purpose of this research is to identify current processes and their associated activities used to develop GIS applications using COTS software; therefore, this study's research question is:

1. Are requirements engineering, software evaluation and selection practices associated with the COTS paradigm applicable to the development of GIS applications?

This includes examining the structure in which the practices are embedded; and also the actual processes that are used by GIS application developers. Three propositions were developed to aid answering this research question.

1. A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications,
2. Standard IS requirements processes can be used for developing GIS solutions, and
3. Standard IS COTS software procurement processes can be used for developing GIS solutions.

To test these propositions an appropriate research methodology is required.

## RESEARCH METHODOLOGY

In this study, a systems development research method was applied. Systems development research approaches have gained more acceptance in the Management Information System (MIS) field. According to Morrison, Morrison & Moore (1997), MIS journals, for example; Communications of ACM and MIS Quarterly contain a reasonable percentage (25%) of system development projects as a central part of the research. These authors advocate that this method is used for defining new methods for solving systems-related problems.

> *"Knowledge gained by researchers both in their selected research domains as well as in the general area of systems development contributes to overall MIS knowledge, theory and practice." (Morrison et al., 1997, p13)*

These authors believed that it is not the actual system being implemented that is regarded as the research object, but the ideas associated with the system development are considered more acceptable as the research effort.

Morrison *et al.,* (1997), described a research strategy that provides guidance for systems development research strategy. This strategy consists of a horizontal $x$ axis (depicting research activities) and a vertical $y$-axis (depicting research outcomes). The $x$-axis identifies three research activities:

1) ***Build***, showing the feasibility of the research,
2) ***Evaluate***, comparing with other approaches, and
3) ***Prove***, explaining why the effort is better than other approaches, thereby contributing towards knowledge or theory.

The $y$-axis describes three research outcomes:

1) ***Formalism***, looking at mathematical or logical constructs used to describe a domain,
2) ***Methods***, sets of guidelines to solve problems, and
3) ***Products***, physical tools that instantiate formalisms and methods.

There are several paths that a researcher may choose from this framework: the formalism-driven approach, the method-driven approach or the product-driven approach.

Other system development research approaches were also considered. These were the *Wharton Model* by Kimbrough & Moore (1992, in Morrison *et al.*, 1997), and the *Process for Systems Development Research* also known as the Arizona Model by Nunamaker, Chen & Purdin (1991). Both approaches involve developing either demonstration or prototype systems that validate the development approach. These approaches are similar to Morrison's *et al.*, (1997), product-driven approach.

The purpose of this study is to prove whether RE practices associated with the COTS paradigm would be suitable to apply in the development of GIS solutions; therefore the method-driven approach was selected as an appropriate research approach. Figure 1 shows the order of steps in the method-driven approach that structured this study.

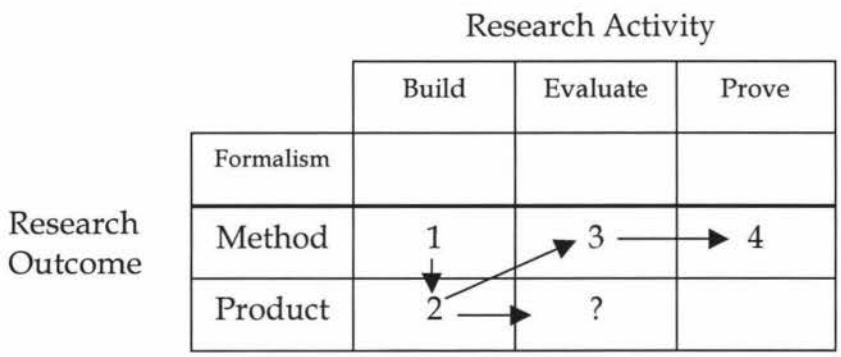


**Figure 1: Method-Driven Research (Morrison *et al.*, 1997, p. 14)**

According to Morrison *et al.*, (1997) the method-driven approach starts with a set of guidelines that can be applied to a problem to produce a solution. The next step is instantiating the methods to produce a product. The product is evaluated by the users, then proven to be better than other approaches. Morrison *et al.*, (1997), discussed the potential problem (shown as the *question*

*mark* in Figure 1), whether the method or the product is evaluated. In this study, the method (being the framework not the product) is to be built, evaluated and proven.

Using the structure of the method-driven approach by Morrison *et al.*, (1997), this study identified the ***method*** as the framework for developing GIS solutions, the ***evaluation*** will compare this framework to other research works, and the framework will be validated or ***proven*** with multiple-case studies. The primary contribution to the research is the *method* (i.e. framework), not the formalism or the product.

The building blocks of this study are depicted in Figure 2, and can be used as a roadmap for the main body of material in this report.

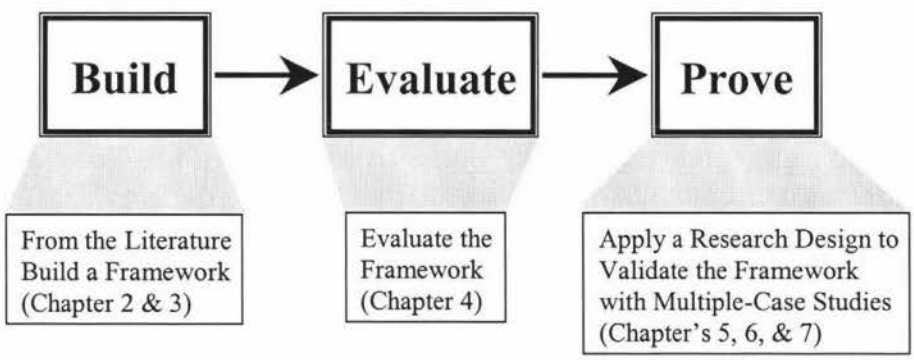


**Figure 2: Overview of Research Approach (adapted from Morrison *et al.*, 1997)**

In order to investigate the research problem, Chapter 2 identifies the processes and activities from the literature in the area of IS, COTS software procurement and GISs. These processes and activities were selected to ***build*** a framework consisting of requirements, evaluating and selecting software for developing GIS applications (Chapter 3).

Chapter 4 ***evaluates*** this framework, by comparing it with frameworks from other studies in the GIS area.

Chapter 5 introduces a case study research methodology design to conduct multiple-case studies for proving the framework. Chapter 6 consists of two parts. The first part is the qualitative data analysis from the multiple case studies that includes within-case and cross-case analysis techniques. The second part ***proves*** the framework by discussing the extent to which the research propositions were met, including study replication and generalisation issues. Chapter 7 concludes, by reflecting, identifying shortcomings and future work.

> *"The thoroughness of the user requirements analysis and the management of the organizational impacts of GIS acquisition are considered to be critical success factors"* (Clarke, 1991, p.477)

# CHAPTER II

# LITERATURE REVIEW

## INTRODUCTION

This chapter reviews literature relating to the research propositions introduced in Chapter 1. To gain an understanding of the requirements of the research propositions, information system (IS) development life cycles, requirements engineering (RE), COTS software procurement activities, IS and geographic information systems (GIS) process activities are discussed in the following sections.

## PART I: DEVELOPMENT LIFE CYCLES

> *"[A Systems Development Life Cycle (SDLC) is] a set of prescribed phases and tasks for the development of an information system. Phases of the SDLC include systems planning, systems analysis, general (conceptual) systems design, systems evaluation and selection, detailed (functional) systems design, and systems implementation"* (Burch, 1992, p.840).

Application development activities must be organised and tracked for a team to achieve the desired results. According to Leffingwell & Widrig (2000), managing requirements effectively cannot occur without well-defined software processes comprised of activities, that the team must perform to produce the final product. These authors state that the software development process "defines who (which member of the team) is doing what (what activity is being performed), when (that activity is done in relation to other activities), and how (the details and steps in the activity) in order for your team to reach its goal" (p. 213). How these activities are achieved, impact on the final time of delivery and the cost of the total project.

All software development projects go through a systematic approach called a life cycle, that consists of a number of activities, from initial realisation through

to retirement (Dorfman, 1997; McConnell, 1996). According to Dorfman (1997), user requirement processes are closely tied to the life cycle used. A life cycle specifies a number of steps to follow, including the activities associated with those steps. These steps help the development team to meet the user requirements, budget and deadline constraints. Dorfman (1997) gave a comprehensive overview of the evolution of the life cycles and where the requirements process fits into these life cycles.

To gain an understanding of the first research proposition "*a life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications*", this section describes various life cycles and process activities used for developing projects with an emphasis on the requirements aspect. Initially, traditional life cycles (i.e., waterfall models, prototyping, incremental development, evolutionary development and spiral models) are described, followed by descriptions of a hybrid life cycle and life cycle processes identified in IS standards.

## WATERFALL MODEL

One of the most well known and traditional life cycles is the *waterfall* model. In the 1970's Winton Royce defined the *waterfall* model (Figure 3), showing steps that followed a logical sequence throughout the development of the software. According to Dorfman (1997), this life cycle model required "each life cycle phase to generate defined products, which must pass a review and be placed under configuration control before the next phase begins" (p.8). Therefore requirements need to be completed before the implementation stage begins.

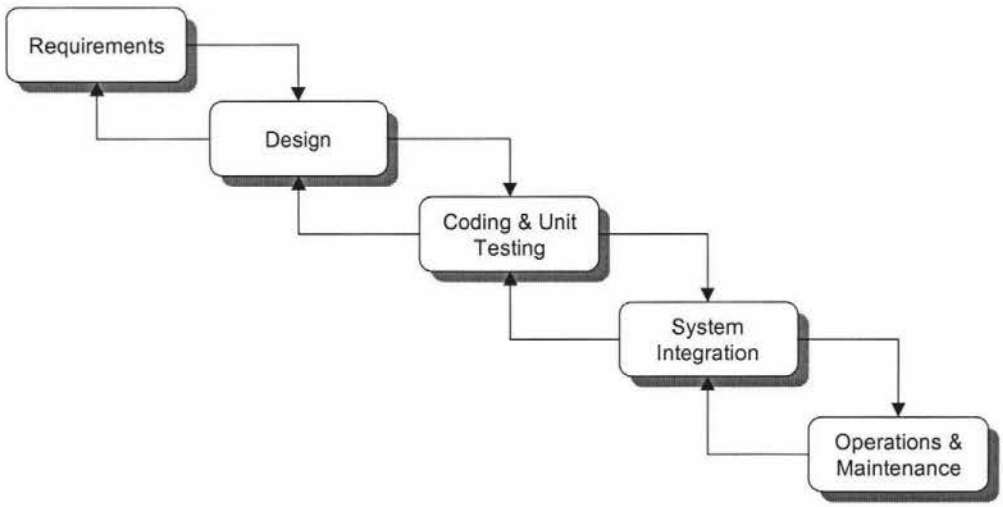


**Figure 3: Waterfall Model of Software Development (Leffingwell & Widrig, 2000)**

Leffingwell & Widrig (2000) discussed that while the *waterfall* model has successfully enforced the requirements process stage, each stage was fixed and did not overlap with any of the other activities. The nature of the waterfall model causes several problems when dealing with user requirements. These are:

- Requirements have to be 'frozen' before continuing with the rest of the development processes (Leffingwell & Widrig, 2000; Dorfman, 1997).
- Larger systems often produced requirements that were not stable enough over a long period of time (Dorfman, 1997; Yeh & Ng, 1997).
- High maintenance costs can occur where developers have to perform requirements tasks in the early stages of development (Yeh & Ng, 1997).
- Users often did not know what their requirements were initially (Dorfman, 1997); therefore, the use of a prototype could fix this problem as it could be used to identify more user requirements, but the waterfall model excluded prototype activities from this model.

## PROTOTYPING MODEL

To help determine user requirements more accurately for a project, Dorfman (1997) recommended the use of the *prototyping* life cycle, as this approach requires some of the system to be built to help determine user requirements. This is useful as the user can gain an insight into the operation of the system. The prototype life cycle is similar to the waterfall model, but the initial stage

has more steps to incorporate the prototype activities. The activities associated with prototyping a system are achieved by initially gathering some user requirements, designing a prototype based on those requirements, building a prototype, then testing the prototype on the user. Several prototypes could be built before moving on to the implementation phase of the development. Once the prototype phase is completed, the steps continue as specified in the *waterfall* model. Dorfman (1997) considered that the *prototype* life cycle is useful to generate a *valid* set of requirements.

The advantages of prototyping are: it helps determine user requirements and reduces user uncertainty about the outcome of the system development (McConnell, 1996; Sommerville, 1996; Wieringa, 1996), and produces clear signs of progress seen by the users (McConnell, 1996). The disadvantages of using the prototyping life cycle model are: important features may be overlooked as other functionality may distract users (Wieringa, 1996), time for development is increased as prototypes take time to produce (Wieringa, 1996), the developer treats the prototype as the final operational product (Wieringa, 1996); the time it takes to create an acceptable product as the changes made can corrupt the product; therefore, the developer does not know how many iterations are required (McConnell, 1996; Sommerville, 1996).

## INCREMENTAL DEVELOPMENT MODEL

According to Dorfman (1997), a strategy for phased development that considers the overlapping life cycle activities (compared to the waterfall model) is the *incremental development* model suggested by Mills *et al.*, (1980, in Sommerville, 1996). Even though the model has the same structure as the waterfall model, the main difference is that the requirements are developed as the system is delivered incrementally (Sommerville, 1996). This avoids the constant changes seen in the prototyping life cycle. As each increment is implemented it allows the user to operate the system to give feedback to improve the requirements.

There are two advantages to using the *incremental development* life cycle: errors in the system can be identified earlier on in the process therefore reduced (Sommerville, 1996; Wieringa, 1996), and the system can be delivered earlier (McConnell, 1996; Wieranga, 1996). The disadvantages with the use of this model are: the system architecture has to be established before the requirements are finished, which means these can be constrained (Sommerville, 1996; Wieringa, 1996), and also errors in the deliverable may not become apparent until later increments are implemented (Wieringa, 1996).

## EVOLUTIONARY MODEL

The *incremental development* model is very similar to another model called the *evolutionary* model, which is also a strategy for phased development. This life cycle model also delivers a product to be used in the operational environment over a period of time (Dorfman, 1997), but contrasts to the *incremental development* model by delivering a portion of the product capability before all behaviour is specified (Wieringa, 1996). This also contrasts to the *prototyping* lifecycle where each delivery not only determines and improves requirements but delivers final operational capability.

Similarly, as with the *incremental development* model, the *evolutionary model* can identify errors early in the development and the user receives deliverables earlier (Wieringa, 1996). Other advantages identified only by Wieringa, (1996), included those requirements that are most certain to reduce the errors in the requirements, consequently changes to these can be accommodated easily, as all requirements are not specified at the start. The disadvantages of the *evolutionary model* are: the developer may not have an understanding of the whole product required for development, the system architecture is also developed incrementally and that may produce a bad architecture structure, and the developer may not identify when to stop the evolutionary cycle.

## SPIRAL MODEL

Another important life cycle development is the study by Barry Boehm (1988, in Leffingwell & Widrig, 2000) where he recommended the *spiral life cycle* model (Figure 4) as guidance for the software development process. Boehm's *spiral* model included *incremental* and *prototyping* models, but also incorporated the same structure as the waterfall model, therefore also including the *waterfall* model weaknesses if misused (Leffingwell & Widrig, 2000). The *spiral* model emphasised the risk and evaluation of the project instead of a deliverable product that was emphasised by the other lifecycles (Dorfman, 1997).

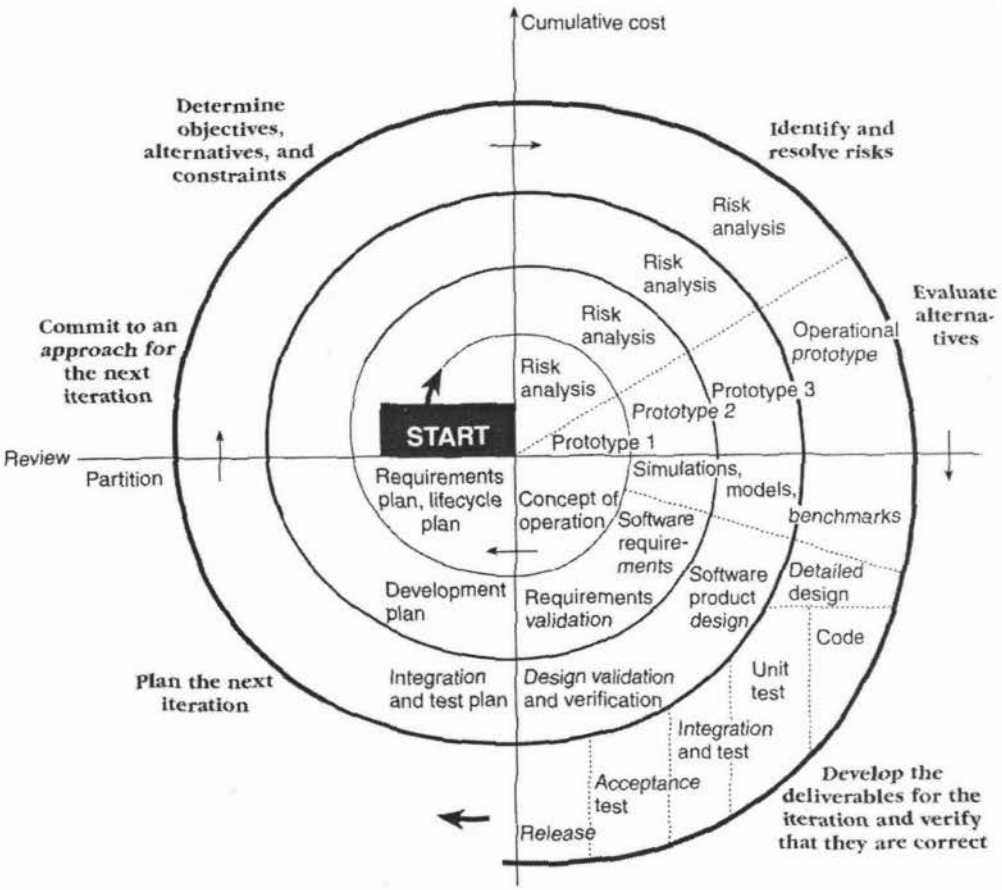


**Figure 4: Spiral Life Cycle Model (adapted from Boehm, 1988, in McConnell, 1996)**

The *spiral life cycle* starts from the middle and works clockwise through the spirals, initiating the project from requirements planning through to the final implementation. Each re-iteration involves risk analysis and the creation of

prototypes, moving the project through to a larger scale with each next iteration. According to McConnell (1996), the spiral model involves six steps:

1. Determine the objectives, alternatives, and constraints,
2. Identify and resolve the risks,
3. Evaluate the alternatives,
4. Develop the deliverables for that iteration, and verify that they are correct,
5. Plan the next iteration,
6. Commit to an approach for the next iteration (if required).

The advantages of the *spiral* model are; as costs increases, risk decreases (McConnell, 1996), and multiple feedback opportunities from the user (Leffingwell & Widrig, 2000). The disadvantages of using the *spiral* model are it is complicated to use (McConnell, 1996; Wieringa, 1996), there are no clear milestones at which products are delivered (McConnell, 1996; Wieringa, 1996; Dorfman, 1997), and it relies on the development manager to manage the risk, if performed incorrectly it may send the development team in the wrong direction (Wieringa, 1996).

## ITERATIVE APPROACH

A hybrid life cycle developed from the traditional models is the rational unified process iterative approach (Figure 5). This approach was introduced by Kruchten in 1995 and was developed to incorporate the best features from the *waterfall* and the *spiral* models (Leffingwell & Widrig, 2000). It is based on the rational unified process, which provides a way of assigning tasks and roles among the development team and allows for flexibility (i.e., the processes can be adapted to the organisation).

> *"Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget"* (Kruchten, 2000a, p17).

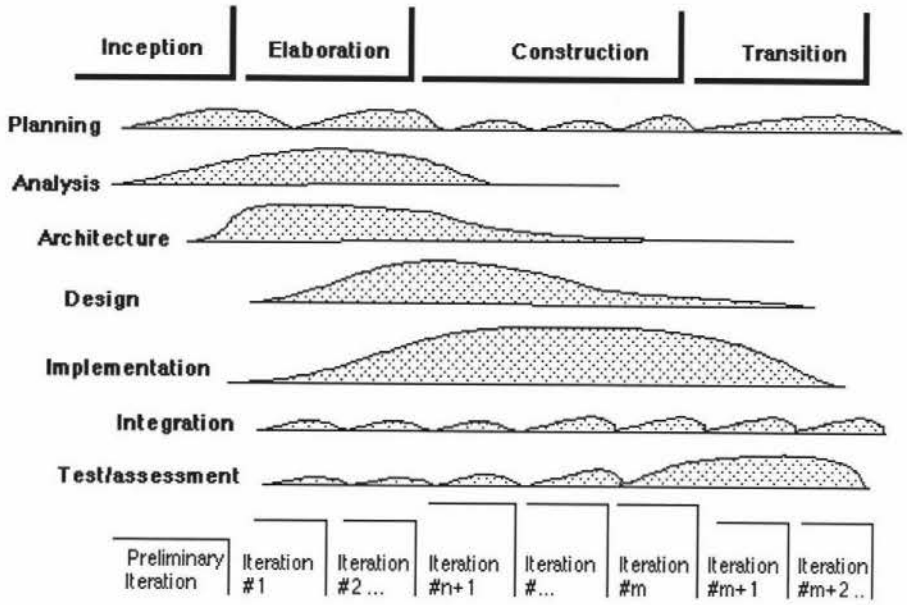


**Figure 5: The Rational Unified Process Iterative Model (Kruchten, 2000b)**

This model accommodates for the management and technical perspectives, which is shown as two dimensions in Figure 5. According to (Kruchten, 2000b), the management perspective accounts for the schedule, budget constraints, commercial and personnel factors, shown in Figure 5 as life cycle phases and workflows (grouped activities under the life cycle phases). The other dimension, is the technical perspective that involves the actual design and construction iterations of the software itself.

From the management perspective, to proceed through the life cycle phases is referred to by Leffingwell & Widrig, (2000) and Kruchten, (2000b) as the *development* cycle. These authors discussed this *development* cycle as having four phases: inception, elaboration, construction and transition.

*Inception* identifies activities with starting and managing the project. The developer then moves on to the e*laboration* phase, which looks at the feasibility of the project. The *construction* phase builds the project, finishing with the t*ransition* phase that delivers the project to the user. The rational unified process iterative approach encourages iterations amongst the phases as well as the prototype projects to be built. Kruchten, (2000b) also discussed a final phase

after the *transition* phase, called *evolution*. This refers to the time in the product's life when the development stops. The next generation can then *evolve* by repeating the *development* cycles phases. The workflows that are part of the management perspective shown in Figure 5 are activities spread incrementally across each phase and iteration of the *development* cycle.

From the technical perspective, Kruchten, (2000b) sees the development of software as evolving incrementally whereby producing a release of an executable product. Contained within each release are supporting artefacts (i.e., plans, release descriptions, documentation, etc.). The iterations may overlap or in some cases run in parallel.

Kruchten (2000b) described this approach as a *generic process,* "the part of the process that applies to all kinds of software developments, across a broad and general range of these discriminants" (p. 5). Discriminants are referred to as the types of businesses, size of projects, and types of applications required for the business. These are major characteristics that can impact on the development of a project.

The work of Kruchten, (2000a), Kruchten, (2000b) and Leffingwell & Widrig, (2000), gave an explicit list of process tasks required to conduct software development. These tasks were categorised under the life cycle phases. Table 1 shows these process tasks in accordance with the four life cycle phase of the iterative approach. Some process tasks run in parallel, within separate phases, for example, *managing changing requirements* occurs in both the elaboration and construction phase.

Based on the tasks identified in Table 1, Leffingwell & Widrig (2000) identified the tasks that would reflect on the requirements management in the rational unified process. This caused some confusion as these authors discussed the tasks as being part of the *requirements* workflow, even though this particular workflow does not appear in Figure 5. The requirements tasks (or mini-workflows as described by Leffingwell & Widrig, 2000) involved the following:

| Life Cycle Phase | Process Tasks |
|---|---|
| Inception | 1. Understand business case<br>2. Scope the project<br>3. Identify feasibility of implementation<br>4. Analyse problem<br>5. Obtain initial requirements<br>6. Create vision document<br>7. Estimate schedule and budget<br>8. Identify project risk factors<br>9. Exhibit and /or demonstrate candidate architecture<br>10. Create a domain model<br>11. Produce one or several prototypes<br>12. Understand stakeholder needs |
| Elaboration | 1. Refine requirements for system<br>2. Elaborate architecture (including the selection of components, integration and assessment).<br>3. Describe software architecture<br>4. Initiate an executable architecture prototype<br>5. Prototype developed and demonstrated<br>6. Create a preliminary user manual draft (optional)<br>7. Define the system definition<br>8. Manage the scope of the system<br>9. Refine system definition<br>10. Manage changing requirements |
| Construction | 1. Manage changing requirements<br>2. Conduct resource management, resource control, and process optimisation<br>3. Complete component development and testing against the defined evaluation criteria.<br>4. Develop the code.<br>5. Fully develop the system architecture and design.<br>6. Assess product releases against acceptance criteria for the vision.<br>7. Complete the user manuals. |
| Transition | 1. Conduct beta testing<br>2. Run system in parallel operation with the legacy system that the project is replacing<br>3. Convert operation databases.<br>4. Train users and maintainers of system on the application<br>5. Fine tune activities, including bug fixing, enhancements for performance and usability<br>6. Test baseline of the application is transitioned to the user community and deployed for use.<br>7. Assess the deployment baselines against the vision and acceptance criteria for the product. |

**Table 1: The Iterative Approach with a Synthesis of Process Tasks (Kruchten, 2000a; Kruchten, 2000b; Leffingwell & Widrig, 2000)**

1. *Analyse the problem.* The main purpose is to produce a vision document that describes user requirements for the system, which evolve through the life cycle. This document is used to agree on

the system features and its goals. This task is used in the inception and early elaboration stages.

2. *Understand stakeholders needs.* Elicit and gather information from the users. This information can be described through using the *use cases* and *scenarios* techniques. This task is performed in the inception and elaboration phases.
3. *Define the system.* This focuses on identifying actors of the system, analysing stakeholder requests by producing formal documents and models (e.g., finalising the use cases). By achieving this, the project team gains a better understanding of the system required. This is performed in iterations of the inception and elaboration stages.
4. *Manage the Scope of the System.* Defining features and use cases (and/or scenarios) that represent the functionality of the system. Also, defining the attributes and maintenance of the requirements. Included, in this task is the development of an iteration plan that defines the frequency of iterations in the project before its release.
5. *Refine the system definition.* Output is an in-depth understanding of the system functionality (defined in use cases), revised documentation, and prototyping the user interface.
6. *Manage changing requirements.* This involves tracing and updating models and documents through the changes of the requirements.

Based on these mini-workflows and the activities identified in Table 1, the following is a list of activities relevant to this study and the life cycle phase:

1. inception - analyse problem, obtain initial requirements, produce one or several prototypes, understand stakeholders needs.
2. elaboration - refine requirements for system, prototype developed and demonstrated, define the system definition, manage the scope of the system, refine system definition, and manage changing requirements.
3. construction - manage changing requirements.

As can be seen from this list a number of these activities only relate to general requirements mostly in the elaboration phase. The iterative nature of the approach can be seen by *refining* activities (particularly in the elaboration phase) and *managing changing requirements* (in both elaboration and transition phases).

Kruchten (2000b) specified that the rational process is not driven by requirements, but they evolve during a cycle, through the various artefacts (i.e., business case document, the vision document, and also through use cases and scenarios).

According to Kruchten (2000a), there are many benefits gained by the use of the *iterative* approach. It allows for: changing requirements, identification of risks early on in the development process, the correction errors found in the iterations, component architecture (COTS software components), the reuse of common code, and the growth of the developers and team members in the project. There were difficulties finding disadvantages to the use of this approach in the literature, as it is a relatively new. According to Kerov (2001), many companies are just presently trialling the *iterative* approach. A question that Kerov (2001), raised in regards to the usage of this approach was the interoperability between the organisation that uses the approach and the subcontractor (if involved with development) that does not.

## Life Cycle Processes

The final piece of literature in this area looks very closely at life cycle process activities that are part of a life cycle. This document is the IEEE Standard for Developing Software Life Cycle Processes (SLCP) IEEE Std 1074-1997. The SLCP can consist of up to sixty five (65) activities[1] mapped to a chosen software life cycle model (SLCM) of the developer's choice. The developer can also decide on the activities that are applicable to the project, therefore it is not necessary to use all of them.

Table 2 is a summary of all the available activities in the standard (IEEE, 1997). The activities are organised into seventeen (17) activity groups. These activity groups are further categorised into five sections (i.e., project management, pre-development, development, post-development, and integral). IEEE, (1997)

recommended that the *integral* section be regarded as a support activity group, that could be combined with the other activities in the project.

| Section Title | Clause | Activity Groups | Activities |
|---|---|---|---|
| Project Management | A.1 | A.1.1 Project Initiation | 1. Create software life cycle process<br>2. Perform estimations<br>3. Allocate project resources<br>4. Define metrics. |
| | | A.1.2 Project Planning | 1. Plan evaluations<br>2. Plan configuration management<br>3. Plan system transition (if applicable)<br>4. Plan installation<br>5. Plan documentation<br>6. Plan training<br>7. Plan project management<br>8. Plan integration. |
| | | A.1.3. Project Monitoring and Control | 1. Manage risks.<br>2. Manage the project<br>3. Identify SLCP Improvement needs<br>4. Retain records<br>5. Collect and analyse metric data |
| Pre-Development | A.2 | A.2.1. Concept Exploration | 1. Identify ideas or needs<br>2. Formulate potential approaches<br>3. Conduct feasibility studies<br>4. Refine and finalise the idea or need. |
| | | A.2.2. System Allocation | 1. Analyse functions<br>2. Develop system architecture<br>3. Decompose system requirements. |
| | | A.2.3. Software Importation | 1. Identify imported software requirements[2]<br>2. Evaluate software import sources (if applicable)[3]<br>3. Define software import method (if applicable)[4]<br>4. Import software (if applicable)[5] |

[1] Activity, a defined body of work to be performed, including its required input and output information (IEEE, 1997).

[2] The means to extract the software requirements that will be satisfied through importation (IEEE, 1997, p.37)

[3] to evaluate candidate sources from which the imported software might be obtained (IEEE, 1997, p.37).

[4] to determine the most appropriate method identified by which the software import source will provide the imported software (IEEE, 1997).

[5] Import the software including documentation (IEEE, 1997, p.37).

| | | | |
|---|---|---|---|
| Development | A.3 | A.3.1. Requirements | 1. Define and develop software requirements.<br>2. Define interface requirements.<br>3. Prioritise and integrate software requirements. |
| | | A.3.2. Design | 1. Perform architectural design<br>2. Design database (if applicable)<br>3. Design interfaces<br>4. Perform detailed design. |
| | | A.3.3. Implementation | 1. Create executable code<br>2. Create operating documentation<br>3. Perform integration. |
| Post-Development | A.4 | A.4.1. Installation | 1. Distribute software<br>2. Install software<br>3. Accept software in operational environment |
| | | A.4.2. Operation and Support | 1. Operate the system<br>2. Provide technical assistance and consulting<br>3. Maintain support request log. |
| | | A.4.3. Maintenance | 1. Identify software improvement needs<br>2. Implement problem reporting method<br>3. Reapply software life cycle. |
| | | A.4.4. Retirement | 1. Notify user<br>2. Conduct parallel operations (if applicable)<br>3. Retire system |
| Integral | A.5 | A.5.1. Evaluation | 1. Conduct reviews<br>2. Create traceability matrix<br>3. Conduct audits<br>4. Develop test procedures<br>5. Create test data<br>6. Execute tests<br>7. Report evaluation results |
| | | A.5.2. Software Configuration Management | 1. Develop configuration identification<br>2. Perform configuration control<br>3. Perform status accounting |
| | | A.5.3. Documentation Development | 1. Implement documentation<br>2. Produce and distribute documentation |
| | | A.5.4. Training | 1. Develop training materials<br>2. Validate the training program<br>3. Implement the training program. |

**Table 2: IEEE Std 1074-1997 Life Cycle Activities**

Table 2 shows the activity groups that are most relevant to this study. They are A.2.3 software importation (under the A.2 pre-development section) and A.3.1 requirements (under the A.3 development section).

The *software importation*[6] activity group provides sub-activities to "extract the software requirements that will be satisfied through importation, to evaluate candidate sources from which the imported software might be obtained, to determine the method of importation and to import the software, including documentation, into the project" (IEEE, 1997 p 37).

The *requirements* activity group focuses on the development of software requirements. This group does not include the development of total system requirements, or the functional allocation of those system requirements to hardware, humans, and software (IEEE, 1997).

Although IEEE, (1997) gave an explicit list to guide development tasks, it does not appear to include any activities that apply to the discriminants identified by Kruchten, (2000b). Such as, types of businesses, size of projects, and types of applications required for the business) that can impact on the development of a project.

## PART II: REQUIREMENTS ENGINEERING

> *"The so-called 'Y2K bug' wasn't a 'bug' at all. It was a requirement problem. No one told the developers that their systems had to operate past the year 1999." (Hooks & Farry, 2000)*

Requirements are fundamental to any system[7] development project. Robertson & Robertson (1999), found that "a requirement exists either because the type of product demands certain functions or qualities, or the client wants that requirement to be part of the delivered product" (p.5). Unfortunately, the literature identified that many projects failed due to inadequate requirements and analysis activities (Faulk, 1996; Dorfman, 1997; Yeh & Ng, 1997; Robertson & Robertson, 1999; Hooks & Farry, 2000). Some of these problems also stem

---

[6] Software importation are software requirements being satisfied by using existing software or acquiring software outside the project (IEEE, 1997).

[7] Yourdon (1989) defined a *system* as user workflows that are automated through software by a computer system. Yourdon (1989) described this type of *system* consisting of several components: computer hardware, computer software, people, data and procedures for operating the system.

from poor change management (Faulk, 1996; Yeh & Ng, 1997; Hooks & Farry, 2000). Dean Leffingwell (in Hooks & Farry, 2000), estimated that "requirements errors account for 70 to 85 percent of software project rework costs" (p. 8). To alleviate the issues identified with requirements it was necessary to investigate the activities associated with requirements engineering (RE) in the literature.

Sommerville & Sawyer (1997), suggest RE involved a systematic approach by using techniques to ensure that system requirements are complete, consistent, and relevant. According to Sutcliffe & Jarke (1996), there are several definitions of requirements engineering (RE). The definition of Sommerville (1989, in Sutcliffe & Jarke, 1996) focused on the outcome of RE defined as "a requirements specification [which] should tell a designer everything he needs to know to satisfy all the stakeholders - but nothing more." Bubenko (1993, in Sutcliffe & Jarke, 1996) defined the principle RE issue as "how to proceed from informal, fuzzy individual statements of requirements to a formal specification that is understood and agreed by all stakeholders" (p. 516). Both the Sommerville and Bubenko definitions emphasise the important contribution made to the RE process[8] by the stakeholders.

Other causes of project failure can be contributed to the lack of concern for the life cycle (Yeh & Ng, 1997; Hooks & Farry, 2000). Life cycles are critical as they guide the tracking and management of requirements in an efficient and predictable way. Various authors agreed that the *requirements engineering process* should start with a set of structured activities in the early stages of the system development (Davis, 1993; Sommerville & Sawyer, 1997) to derive, validate and maintain the requirements specification document[9] (Davis, 1993; Sommerville & Sawyer, 1997; Dorfman & Thayer, 1990, in Leffingwell & Widrig, 2000). The *requirements management process* not only occurs at the front end of all life cycles, but also throughout the life cycle. Life cycles encourage a

[8] Kotonya & Sommerville (1998) define a "*process* is an organised set of activities which transforms inputs to outputs." (p.25).

higher degree of participation during the design, development, and maintenance (possible rework and configuration management) processes (Andriole, 1996). This is consistent with both Sommerville and Bubenko requirements engineering definitions with the emphasis on the importance of the stakeholder. Andriole, (1996) and Hooks & Farry (2000), discussed inadequate requirements can also occur due to the lack of management support, lack of skilled personnel, or lack of the training.

## REQUIREMENT PROCESSES

The *requirements engineering process,* consists of activities that lead from necessary input to the required output. According to Kotonya & Sommerville (1998), the necessary input for the *requirements engineering process* (Figure 6) are pre-existing system information, stakeholder needs, organisational standards, regulations, and domain information. Desired outputs from the *requirements engineering process* are agreed requirements, system specification and system models. Kotonya & Sommerville (1998) describe each requirements activity as part of the *requirements engineering process* (illustrated in grey):

- *Feasibility study,* describes the estimated costs associated with the project and whether it will satisfy the users' environment.
- *Requirements analysis,* requires the analyst to collect information through observation, interviews and discussions with the users to identify their needs. The outcome of this process is the derivation of requirements. Some prototypes may also be developed to help the users understand the requirements.
- *Requirements definition,* translates the information gathered into a set of requirements that define the system.
- *Requirements specification,* is an accurate description of the system as a contract between the developer and client. The requirements specification may be carried out in parallel with the design stage.

[9] Writing requirements specifications is not part of the scope of this study, but Hooks & Farry, (2000), and various international standards (IEEE, ANSI, NASA, and US Military standards) describe in detail how to write requirements specifications.

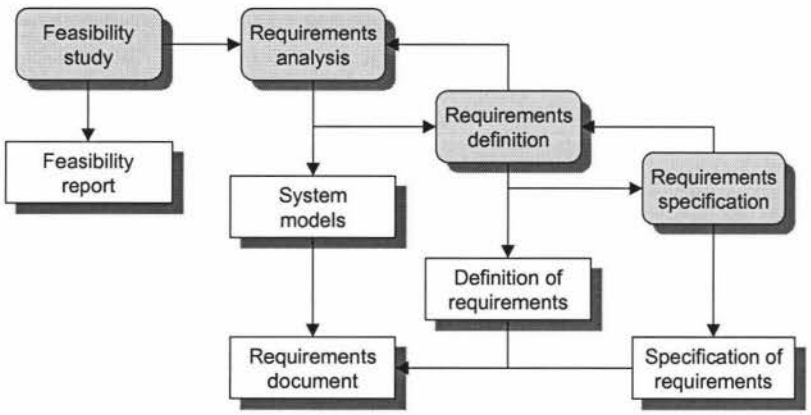


**Figure 6: The Requirements Engineering Process (Sommerville, 1996, p67)**

The requirements activities in the *requirements engineering process* are not performed in the order shown in Figure 6, but reiterated during the definition of the specification. Also, new requirements may be derived from the reiterations (Sommerville, 1996).

## CLASSIFYING REQUIREMENTS

The literature identified various classifications of requirements that caused some confusion when defining them. Most of the literature identified system and software requirements, but Andriole (1996) went one step further by stating that there was a distinction between software and user requirements. "Most of the requirements analysis conducted by systems analysts are oriented toward the specification of software functions and routines. Far less effort is expended on behalf of end users, or the process by which user requirements are identified, refined, and validated" (p. 6)

The most common requirements classifications are *system* and *software* requirements. Dorfman (1997) found that *system* and *software* requirements are treated together, but there are important distinctions between them. According to Davis (1993), system and software requirements need a different approach that affects when the requirements phase starts in a life cycle. If the project required software only, then the route taken in the life cycle would be to

identify software *requirements* first, then proceed to the system *design* phase. If the project required both hardware and software, then system *requirements* are identified first, followed by the system *design,* which are the same steps for software only. The difference appearing at the system design stage, both hardware and software *requirements* are defined separately, followed by separate hardware and software *design,* the proceeding through the rest of the life cycle stages.

**Software Requirements**

Various authors discussed what they thought software requirements are and their purpose in the development. Leffingwell & Widrig (2000), stated that software requirements should state what the software does on behalf of the users (particularly the interactions with the user) and describe the boundary of the system, that is, inputs and outputs. Dorfman (1997), believed that software requirements (along with hardware requirements if applicable to the development) are defined below the system level. According to Jackson (1995), he described the phenomenon of the application domain and relationships with the machine.

Once these requirements are defined, they must reside in a *software requirements specification* for tracking and maintenance. Davis (1993) explained the software requirements specification (SRS) as:

> *"A requirements specification for a software system or the software part of a software/hardware system; also called program performance specification and functional description"* (p.372).

Davis (1993) believed this document's purpose is to describe the external behaviour of the software *system.* It serves as communication between customers, users, analysts and designers (Davis, 1993), and in particular between developers (Dorfman, 1997). The specification also supports *system*-testing activities and controlling the evolution of the *system.* The SRS is also referred to as the functional description, functional requirements and specification by others (Davis, 1993).

*Functional & Non-Functional Requirements*

Software requirements can be classified into *functional* and *non-functional* requirements. Yeh & Ng, (1997) found that functional requirements are a core part of the software requirements document.

The main purposes of functional requirements are to capture the functionality of the system and describe what the target system does (Andriole, 1996; Yeh & Ng, 1997; Robertson & Robertson, 1999; and Leffingwell & Widrig, 2000). The *system* functionality defined by Yourdon, (1989) and Andriole, (1996), may disclose the systems' tasks and subtasks, the systems' boundaries and its connections to adjacent system, and the functions the system must perform to manipulate data (Robertson & Robertson, 1999). Various modelling techniques, dataflow diagrams, process diagrams and so forth, can be used to illustrate the system's functions.

*Non-functional* requirements contain information on how the system will operate. The relationship between non-functional and functional requirements is that non-functional requirements place constraints on how the functional requirements will be implemented (Sommerville & Sawyer, 1997). The following is a list of qualities that a product must have that are classified as non-functional requirements (including qualities identified by other authors):

- Look And Feel - The Intended Appearance (Robertson & Robertson, 1999),
- Usability (Robertson & Robertson, 1999; Leffingwell & Widrig, 2000),
- Performance (Faulk, 1996; Robertson & Robertson, 1999; Leffingwell & Widrig, 2000),
- Operational (Yeh & Ng, 1997; Robertson & Robertson, 1999),
- Maintainability (Faulk, 1996; Robertson & Robertson, 1999),
- Portability (Andriole, 1996; Robertson & Robertson, 1999),
- Security (Andriole, 1996; Yeh & Ng, 1997; Robertson & Robertson, 1999)
- Modifiability (Andriole, 1996),
- Quality (Yeh & Ng, 1997)
- Reusability (Faulk, 1996)
- Dependability (Faulk, 1996)
- Safety (Faulk, 1996)
- Supportability (Leffingwell & Widrig, 2000)
- Testability (Andriole, 1996),
- Reliability (Andriole, 1996; Leffingwell & Widrig, 2000),

- Cultural (Robertson & Robertson, 1999),
- Political As In Human Factors (Robertson & Robertson, 1999), And
- Legal conformance to applicable laws (Robertson & Robertson, 1999).

The most cited non-functional requirements from this list were performance, security, usability, operational, maintainability, portability and reliability.

Faulk (1996) considered the term's *functional* and *non-functional* to be confusing. Therefore, he preferred to use the terms *behavioural* requirements and *developmental quality attributes. Behavioural* requirements include information to determine if the runtime of an implemented system is acceptable. "Behavioral requirements specify the inputs to the system, the outputs (responses) from the system and behavioural relationships between them; also called functional or operational requirements" (Davis, 1993, p.366). It would appear that Davis (1993) is referring to functional requirements. Faulk (1996) found the *developmental quality attributes* included "constraints on the attributes of the system's static construction." Therefore testability, changeability, maintainability, and reusability are included in this area. Davis did not refer to developmental quality attributes as Faulk (1996) did. Davis classified the natural partner to *behavioural* requirements as *nonbehavioural* requirements. *Nonbehavioural requirements* describe the required overall attributes of the system including portability, reliability, efficiency, human engineering (interactive styles and error messages), testability, understandability, and modifiability. The *developmental quality attributes* and *nonbehavioural* requirements appear to have very similar purposes as non-functional requirements.

**System Requirements**

Defining *system* has been difficult considering the various definitions researchers have taken. Yourdon (1989) defined a *system* as user workflows that are automated through software by a computer system. Yourdon (1989) described this type of *system* consists of several components: computer

hardware, computer software, people, data and procedures for operating the system. *System* requirements should describe the environment and operation of the system to be built (Dorfman, 1997), without discriminating between the medium, for example, software or hardware (Davis, 1993). A different approach was taken by Robertson & Robertson, (1999), as a *system* taken in the context of a business system (or application) not the computer or software itself. For this study, system requirements will be taken in the context of Dorfman (1997) and Davis's (1993) definition.

The literature identified some terminology differences between system and global-system requirements. Sommerville, (1996) specified that requirements that covered the system over-all, were known as *global* system requirements which are essential properties of the system as a whole (Sommerville & Sawyer, 1997). According to Dorfman, (1997), system requirements contain descriptions of the external behaviour of the system (Dorfman, 1997; Leffingwell & Widrig, 2000). These requirements can refine the system into subsystems and describe the interfaces among the subsystems (Dorfman, 1997; Leffingwell & Widrig, 2000). Also, the system requirements are written into a specification to serve as communication between users and/or purchasing organisation, as well as analysts and developers concerned with the development (Dorfman, 1997).

System requirements are written into a document that describes the environment and operation of the system to be built. In this context, Davis (1993) defined a *systems requirements specification* as:

> *"A requirements specification for the over-all system without discriminating between the medium (e.g., software or hardware) in which the requirements will be satisfied"* (Davis, 1993, p.374).

Some confusion may occur with the terminology that requires some clarification. Dorfman, (1997) and Leffingwell & Widrig (2000) stated system requirements contain descriptions of the *external behaviour* of the system. This is not to be confused with the description given by Davis (1993), that software requirements describes the external behaviour of the *software system* (as

specified in the software requirements section). Also, Robertson & Robertson (1999) defined *system* in the context of a business system (or application), not computer hardware or *software*. This is not to be confused with *system requirements* containing both hardware and *software* requirements.

**Specifying Requirements**

The requirements engineering process is referred to as a 'what' process, not a 'how' process. According to Faulk, (1996), "most authors agree in principle that requirements should specify 'what' rather than 'how.' In other words, the goal of requirements is to understand and specify the *problem* to be solved rather than the *solution*" (p. 85). Hooks, (1990), identified the 'how' instead of 'what' as being one of the most common problems of poorly written requirements. Wallnau, Carney, & Pollak (1998) found that the evaluation criteria for software procurement (see next section) is preferably defined as 'what' processes (or goals), instead of 'how' as in implementation type requirements.

Hooks & Farry, (2000) have the last words on *requirements*:

> *"Bad requirements result in cost overruns, schedule slips, frustrated and overworked employees, unhappy customers, lost profitability, and limited careers. All else being equal, high-quality requirements contribute to high-quality products completed on schedule and within budget"* (p. 7).

# PART III: COTS SOFTWARE PROCUREMENT PROCESS

> *"System procurement is the process of acquiring a system for an organisation to meet some identified need. The system may be bought as a whole, may be bought as separate parts that are then integrated or may be specially designed and developed"* (Sommerville, 1996, p.26).

According to Maiden & Ncube, (1998), commercial off-the-shelf (COTS) software is generic software products produced from commercial suppliers. Many organisations are turning to the use of COTS software in their systems. The major advantage is, that it is cheaper to buy an existing system off-the-shelf, than to design, build and manufacture a system (Sommerville, 1996). Unfortunately, COTS-based systems introduce new problems, for example, when to acquire new requirements and when to reduce the number of candidate products (Maiden & Rugg, 1996).

Methods of managing COTS software procurement in a structured fashion for a development project are discussed in the following section. This will include descriptions of various COTS software procurement life cycle approaches, COTS software procurement processes, as well as instruments used for evaluating and selecting software. These will be judged in accordance with the third research proposition of this study, *"standard IS COTS software acquisition processes are required for developing GIS solutions "*.

## SOFTWARE PROCUREMENT APPROACHES

There are three different COTS software procurement process life cycle approaches:

1. COTS intensive system life cycle (Wallnau *et al.*, 1998; Carney, 1997),
2. COTS software usage life cycle (Abts, Bailey, Boehm, & Brown, 1999), and

3. IEEE standard software acquisition life cycle approach (IEEE, 1998) that includes discussion of the ISO/IEC, (1995) standard exemplar.

**Life Cycle Approaches**

Carney (1997) in conjunction with Wallnau *et al.,* (1998) found the following activities are important differences between a traditional life cycle models and a COTS-based life cycle model: examine the marketplace, qualify and select one or more products, adapt the product to some specific system requirements, assemble the system, and update the products as needed.

Carney (1997), and Wallnau *et al.,* (1998) identified three activities that did not apply to proposition 3 (i.e., not evaluating and selecting software or requirements): adapt the product to some specific system requirements, assemble the system, and update the products as needed.

Abts *et al.,* (1999) based their life cycle on a prototype development life cycle, but integrated COTS software components. A COTS software usage life cycle typically follows the following pattern and includes a similar SDLC approach:

1. Qualify COTS products,
2. Establish system requirements,
3. Administer COTS software acquisition,
4. Prototype the system including COTS software,
5. Fully integrate COTS software,
6. Test completed prototype, including COTS Software,
7. Delivery, and
8. Operation and maintenance (includes the updated product releases).

Abts *et al.,* (1999) explained that steps 1 and 2 of this approach, required iterating as certain flexibility must exist by the system users or acquirers towards changing their requirements to the available capabilities of COTS products already on the market. Another implication of this life cycle when using COTS components is the reduction of effort allocated to code

development. A tradeoff is a need for increased effort in the testing phase, which must be performed at some level during initial component qualification or assessment.

Abts *et al,*. (1999), four activities that did not apply to proposition 3 (as they did not comply with evaluating and selecting software or requirements) of this study are administer COTS software acquisition, test completed prototype, including COTS software, delivery, and operation and maintenance (including the refresh of updated product releases).

**Software Procurement Standards**

The IEEE standard called 'IEEE Recommended Practice for Software Acquisition' (IEEE, 1998), described practices for performing software acquisitions. This life cycle had nine steps that start with the decision to acquire COTS software; through to when the software is no long available for use. There are five phases to this process: planning, contracting, product implementation, product acceptance, and follow-on phases. Table 3 depicts each phase that is defined by prescribed initiation and completion milestones in association with steps in software acquisition. Each step in the software acquisition process of COTS software is described in more detail in the standard (including checklists for the developer).

The steps in software acquisition process (shown in Table 3) that will be applied to this study (in accordance with the research propositions) are: determining the software requirements (planning phase), identifying potential suppliers (contracting phase), preparing contract requirements (contracting phase) and evaluating proposals and selecting the supplier (also contract phase).

IEEE produced a standard called *The Information Technology - Guideline for the evaluation and selection of CASE Tools standard* (ISO/IEC, 1995). This standard can be applied to this study, as it contains activities for evaluating and selecting products. By exchanging references to CASE tools with COTS software

activities the standard can be applied in a generic sense to evaluating and selecting GIS COTS software.

| Phase | Phase initiation milestone | Phase completion milestone | Steps in software acquisition process |
|---|---|---|---|
| Planning | Develop the idea | Release the RFP | • Planning organisational strategy<br>• Implementing organisation's process<br>• Determining the software requirements |
| Contracting | RFP is released | Sign the contract | • Identifying potential suppliers<br>• Preparing contract requirements<br>• Evaluating proposals and selecting the supplier |
| Product implementation | Contract is signed | Receive the software product | • Managing supplier performance |
| Product acceptance | Software product is received | Accept the product | • Accepting the software |
| Follow-on | Software product is accepted | Product is no longer in use | • Using the software |

**Table 3: Software Acquisition Phase Milestones (IEEE, 1998)**

Table 4 illustrates the four process phases in accordance to the descriptions in ISO/IEC, (1995), these are: initiation, structuring, evaluation, and selection. *Initiation* defines the objectives, direction and management of the evaluation and selection process. *Structuring* defines a set of structured requirements based on the characteristics of the product. *Evaluation* results in a profile of the quality of the product. Finally, the *selection process* identifies the most suitable product. Any reference to *CASE Tools* in Table 4 has been altered to the phrase *COTS software*.

All activities in Table 4 will be used in this study with two exceptions as they do not apply to the research propositions: goal setting (initiation phase), and project planning and control (also initiation phase).

| Process phase | Activities |
|---|---|
| Initiation | Goal setting<br>Establishing selection criteria<br>Project planning and control |
| Structuring | Requirements definition<br>Organisational information gathering<br>Requirements identification<br>Requirements structuring<br>COTS software information gathering<br>Identifying final candidate COTS software |
| Evaluation | Preparing for evaluation<br>Evaluating COTS software<br>Evaluation reporting |
| Selection process | Preparing for selection<br>Applying the selection algorithm<br>Recommending a selection decision<br>Validating the selection decision[10] |

**Table 4: COTS Software Acquisition Phases (adapted from ISO/IEC, 1995)**

### Software Procurement Synthesis

Table 5 is a synthesis of the activities that best represent the processes required for proposition 3. The table consists of process activities that related to evaluating and selecting COTS software, and the reference in the literature. The activities appear in the table in no particular order.

Table 5 consists of sixteen (16) activities that were synthesised from the literature. There are two types of process activities identified in this table: 1) general IS requirements and 2) COTS software procurement activities. The first type of activity applies to proposition 2 and the second activity group applies to proposition 3. Therefore, these activities will be adopted in Chapter 3 (Build A Framework). Activities that did not appear in this table have already been recognised as not complying with research proposition 2 and 3.

[10] The original goals and selection guidelines should be reviewed and compared to the evaluation results and other data relating to the recommended selection (ISO/IEC, 1995)

| Activity Group | No. | Process Activity | Reference |
|---|---|---|---|
| Requirements | 1 | Requirements definition | ISO/IEC, (1995) |
| | 2 | Establish system requirements, | Abts *et al.*, (1999) |
| | 3 | Determining the software requirements | IEEE, (1998) |
| | 4 | Prototype the system including COTS software, | Abts *et al.*, (1999) |
| COTS Software Procurement | 5 | Establishing selection criteria | ISO/IEC, (1995) |
| | 6 | Identifying potential suppliers | IEEE, (1998) |
| | 7 | COTS software information gathering;<br>examine the marketplace | ISO/IEC, (1995)<br>Carney (1997); Wallnau *et al.*, (1998) |
| | 8 | Identifying final candidate COTS software | ISO/IEC, (1995) |
| | 9 | Preparing for evaluation, | ISO/IEC, (1995) |
| | 10 | Evaluating COTS software<br>Qualify COTS products.<br>Evaluating proposals and selecting the supplier<br>qualify and select one or more products, | ISO/IEC, (1995)<br>Abts *et al.*, (1999)<br>IEEE, (1998)<br>Carney (1997), Wallnau *et al.*, (1998) |
| | 11 | Evaluation reporting | ISO/IEC, (1995) |
| | 12 | Preparing for selection | ISO/IEC, (1995) |
| | 13 | Applying the selection algorithm | ISO/IEC, (1995) |
| | 14 | Recommending a selection decision | ISO/IEC, (1995) |
| | 15 | Validating the selection decision | ISO/IEC, (1995) |
| | 16 | Preparing contract requirements | IEEE, (1998) |

**Table 5: Synthesis of Processes for COTS Software Procurement Approaches**

There were two activities that had multiple references. First, the process activity 7 a) COTS software information gathering; and b) examine the marketplace. These activities both referred to gleaning information about COTS software and to do this the market place must be looked into. These two activities will be amalgamated into one activity called '*gather supplier information*'. Second, the process activity that refers to *evaluating or qualifying COTS software* was identified by all four references, therefore considered an important activity. These four activities will be amalgamated together and called 'evaluating and qualifying COTS software'. The activities c) *evaluating proposals and selecting the supplier* and d) *qualify and select one or more products* appear to be two separate activities (i.e., *evaluate* and *select*). The *selecting the supplier/product* part of the activity could be amalgamated with the activity '*recommending a selection*

*decision*', as the outcomes are the same. This activity will be referred to as '*select COTS software*'.

Finally, the prototyping activity (4) has a place in this table as prototypes are known to help obtain more requirements, and in this particular case this activity will help gather more requirements relating to the COTS software product.

## SOFTWARE EVALUATION INSTRUMENTS

Carney (1998) refers to the term *evaluation* as a *decision aid,* "..evaluation of commercial products *for the purpose of deciding whether to select one for use*" (p. 2). In this context, Carney (1998) specified a number of activities associated with product selection. The exemplar used in the last section ISO/IEC, (1995) (for evaluating and selecting CASE Tools) described explicitly *evaluation* and *selection* as two separate processes. The purpose of the *evaluation* process is to produce documents as input for the selection process. The purpose of the *selection* process is to identify the most suitable product among the candidate products, and to ensure that this product reaches the project's objectives. Unfortunately, Wallnau *et al.,* (1998) and Maiden & Ncube (1998), identified that COTS software procurement practices are generally poor, due to the lack of published literature on the topic. Wallnau *et al.,* (1998) suggest people must learn to make COTS decisions quickly, obtaining COTS software information, throughout the design phase and with an appropriate degree of rigour. Bonnington & Todd (2001) investigated four COTS software evaluation instruments to identify various activities required for evaluating and selecting COTS software. The following section contains the results of this investigation.

To improve the COTS software procurement process, four evaluation methods developed specifically for selecting COTS software were identified: PORE - procurement-oriented requirements engineering (Maiden & Ncube, 1998), Requirements Driven Process (Rolland, 1999), COSSET - commercial-off-the-

shelf system evaluation technique (Puma Systems, 1999), and COCOTS - Constructive COTS Model (Abts *et al.*, 1999).

**Procurement-Oriented Requirements Engineering**

Maiden & Ncube, (1998) developed PORE by extending established requirements acquisition techniques to incorporate component-based software engineering. They believed that requirement engineers would make better choices by using their approach that selects or rejects COTS software. They also found that a complete requirements specification is not always required, that it is sufficient to acquire requirements that discriminate between products, then use the selected product in a prototype for gathering more requirements.

Maiden & Ncube (1998) advocate requirements engineers should use their three templates to make better choices when selecting or rejecting COTS. Template 1 is based on supplier-provided information. Template 2 is based on supplier-led demonstrations using test cases for individual requirements and Template 3 is based on customer-led product exploration.

**Requirements Driven Process**

Rolland (1999) proposed a process that gives developers guidance in the selection and assembly of components for COTS. Rolland describes a "matching process between the requirements asked of the system on one hand and the requirements (constraints) imposed by the usage of components on the other". (p. 985). The requirements driven process (RDP) identifies intentions and strategies rather than goals as in the PORE process model. An intention identifies a service that the system or component intends to provide, whereas a strategy is the manner in which the intention can be achieved. This cyclic requirements driven process also has feedback loops between activities; therefore iterative.

### Commercial Off-the-Shelf System Evaluation Technique

Puma Systems, (1999) developed COSSET as an approach for determining viable candidate COTS software. This technique is based on market surveys that guide the refinement of an organisation's requirements. The purpose of COSSET is to generate reports that address technological, functional and programmatic considerations of the selection that is compliant with the requirements goals (Puma Systems, 1999). A client of this system reported that COSSET "sets clear guidelines [and] integrates organisational and technological considerations" (Ryberg, 2000, p 1).

### Constructive COTS Model

The COnstructive COTS model (COCOTS) by Abts *et al.*, (1999), is a cost estimation model designed to capture explicitly the important costs associated with COTS component integration. They found that "the key to success in using COTS components is being able to identify whether they fit the current procurement situation -technically, economically, and strategically" (Abts *et al.*, 1999, p. 2). The intention of COCOTS is a cost estimation model, as it only addresses the cost of the COTS components in the system. The system has four submodels used to capture the costs of the components: assessment (identifies COTS components are selected for use), tailoring (the preparation of the COTS program for use), glue code (development and testing of code for integration purposes), and volatility (frequency in which new versions or updates of the components impact development).

### COTS Evaluation Instrument Synthesis

Table 6 is a synthesis of process activities identified in the literature for COTS software evaluation instruments in accordance with the third research proposition. The results shown in Table 6 are twelve (12) activities listed no particular order.

| Process Activity | Reference |
|---|---|
| 1. Construct As-Is Map | Rolland (1999) |
| 2. Identify and prioritise requirements (construct to-be map) | (Maiden & Ncube, (1998); Abts *et al.*, (1999); Puma Systems, (1999); Rolland, (1999) |
| 3. Perform technology search | Maiden & Ncube, (1998); Puma Systems, (1999) |
| 4. Conduct Technical evaluations | Maiden & Ncube, (1998); Puma Systems, (1999) |
| 5. Conduct product demonstrations | Maiden & Ncube, (1998); Puma Systems, (1999) |
| 6. Analyse product information | Maiden & Ncube, (1998) |
| 7. Make decisions about product-requirements compliance | Maiden & Ncube, (1998); Puma Systems, (1999) |
| 8. Select COTS software (construct COTS map) | Maiden & Ncube, (1998); Abts *et al.*, (1999); Puma Systems, (1999); Rolland, (1999) |
| 9. Finalise market survey results | Puma Systems, (1999) |
| 10. Tailor COTS components for use | Abts *et al.*, (1999) |
| 11. Integrate components using glue code | Abts *et al.*, (1999); Rolland, (1999) |
| 12. Capture volatility effects in system development (e.g. updates of versions) | Abts *et al.*, (1999) |

**Table 6: Synthesis of Processes Activities for COTS Software Evaluation Instruments**

The main activities identified in Table 6 are all associated with COTS software evaluation process, there are no activities identified as general IS activities. The process activities that represent only the evaluation and selection process were identified as activities 2 through to 8. The other activities have no connection to the research propositions in this study as they represent initiation of the project (i.e. activity 1 - construct as-is map) and implementation (i.e., activities 9 through to 12).

All these activities selected for the study had multiple references, with the exception of activity 6 (i.e., analyse information) by Ncube & Maiden (1999). Activities 2 (i.e., identify and prioritise requirements) and 8 (i.e., select COTS software) had all four references, therefore an important activity recognised by the literature. The other activities had at least two references; therefore having a reasonable representation in the literature.

Bonnington & Todd (2001) identified relationships between the instruments, and these were:

1. The parallel development of requirements with evaluation results in the activity of defining the evaluation criteria for selecting a COTS component as being critical to the whole development process.
2. COCOTS was the only instrument that did not refer explicitly to creating criteria but provided a comprehensive list of attributes that the software product should be weighted against.
3. The evaluation criteria reported as being important were: product performance, product reliability, supportability, security, and costs to acquire and maintain.
4. Each of the instruments identified the categorisation and prioritisation of the product evaluation information being extracted.
5. The Maiden & Ncube (1998), Puma Systems, (1999) and Rolland (1999) methods suggested using the analytical hierarchy process (AHP) technique to help with decision-making for selection purposes. Both Maiden & Ncube (1998), and Puma Systems, (1999) suggested developers use the software package 'ExpertChoice' which is based on AHP.

## PART IV: GEOGRAPHIC INFORMATION SYSTEMS

The purpose of this section is to review GIS literature that contains requirements and COTS software procurement activities, including a section on GIS standards.

### THE COTS LINK

According to Huxhold & Levinsohn (1995), there are two ways in which a GIS project can fail. First, a project team can decide which GIS software product is the 'best', and second, they can base their decision on the price or popularity of the product. This decision making does not take into account the organisation's domain or the environment the user operates in. In both cases, the organisation's business domain is ignored and with a large number of products available in the marketplace, makes choosing a product to meet the organisations needs difficult (Huxhold & Levinsohn, 1995).

> *"It is virtually impossible to select the most appropriate system without first analysing them in terms of the important applications for the particular GIS being planned."* (Huxhold & Levinsohn, 1995, p.150)

## GIS REQUIREMENTS

The difficulty of finding current literature on application development in the GIS area could be due to different understandings of the same terms used in the GIS and IS fields, for gathering information from stakeholders about how the final system is to function is called *requirements engineering* in the IS field, but identified as *needs analysis* in the GIS field.

Bubenko (1993, in Sutcliffe & Jarke, 1996) stated the principle RE issue as "how to proceed from informal, fuzzy individual statements of requirements to a formal specification that is understood and agreed by all stakeholders" (p516). In a GIS context, Huxhold & Levinsohn (1995) classified *needs analysis* as "assessing the needs of the organisation, determining the workings of various business units, their information needs and an assessment of how GIS could be applied to identified work." (p. 87). The PTI (Public Technology, Inc.), Urban Consortium & ICMA (International City Management Association (1991), identified two components critical to assessing needs: 1) an inventory of current information resources, and 2) identification of gaps in resources to discover what could be filled by a GIS. There are also various methods used for gathering information for needs assessment, including questionnaires and interviewing personnel, observations and reviewing relevant documents (PTI *et al.*, 1991). These references to needs analysis and assessment in the GIS field are similar to requirements related activities described in the requirements engineering section.

One paper in the GIS field discussed requirements engineering in the IS context. Kosters, Pagel & Six, (1995) focused on the requirements engineering process where the problem domain is analysed and the requirements of the target software are defined. "Requirements engineering comprises the analysis of the

problem domain and the modelling, *resp.* specification, of the system in an implementation independent manner" (Kosters, & Pagel, 1996, p. 2).

Kosters *et al.,* (1995) stress the need for a "domain-tailored requirements engineering method for the development of GIS-applications" (p307). The problem is there is a high demand for user-specific applications in a broad range of problem domains and this makes the requirements engineering process more complex (Kosters *et al.,* 1995). These authors state that basic application requirements could be omitted for the following reasons:

- The broad span of application domains.
- The users do not accept standard software as they expect their organisation's domain to be reflected in the system.
- GIS applications are implemented as embedded systems; therefore existing hardware and software environments must be carefully considered.

Another issue identified by Kosters *et al.,* (1995) in the GIS field, was a lack of standardisation that implicates the interchange of information across systems.

## FUNCTIONAL REQUIREMENTS

Defining applications is initiated early in the project and is accomplished when analysing data needs. According to Huxhold & Levinsohn 1995), GIS applications rely heavily on data as input that produces output to be used by other functions. According to Maguire & Dangermond, (1997) functionality of a GIS is defined as follows:

> *"... all the data collection, storage, manipulation, analysis and presentation operations carried out by GIS"* (p.319).

Maguire & Dangermond (1997) found that the identification of such functionality is important as it helps users evaluate systems they are considering purchasing. The functions that Huxhold & Levinsohn (1995) and Maguire & Dangermond (1997) refer to, can be identified as *functional requirements* in the IS field.

Bonnington & Todd (2001) found that the identification of functionality is important as it helps users evaluate systems they consider purchasing. They proposed a framework of GIS functions based on the framework by Maguire & Dangermond, (1997), and the analysis of other GIS functions proposed by other authors (Smith, Menon, Staff, Estes, 1987; Guptill 1989; Dangermond 1990). Table 7 contains GIS functions and descriptions, with associated references:

| Function | Description | Reference |
|---|---|---|
| 1. User Interface | User interaction with the system via menus, help screens and graphical displays that gain access to the GIS database. | Guptill, (1989) |
| 2. Capture, transfer, validate and edit | Provides functions used to load digital data into a GIS. | Smith *et al.*,(1987); Maguire & Dangermond, (1997) |
| 3. Store, structure and manage these functions | Provides functions applied to a GIS database | Maguire & Dangermond, (1997) |
| 4. Management (i.e. data sets in a DBMS) | Provides the environment where GIS functions by which the data can be controlled. | Smith *et al.*, (1987); Guptill, (1989), |
| 5. Restructure, generalise and transform | Provides functions that allow data from different sources to be converted to a common format for analysis | Maguire & Dangermond, (1997) |
| 6. Map manipulation | Transforms data into Information suitable for a given application. | Smith *et al.*, (1987) |
| 7. Query and analysis | Provides functions for spatial query and analysis of GIS data. | Smith *et al.*,(1987) Maguire & Dangermond, (1997) |
| 8. Present | Provides functions for GIS data to be presented in maps, charts, reports and table formats. | Smith *et al.*,(1987); Maguire & Dangermond, 1997) |

**Table 7: GIS Functionality**

Table 7 consists of eight (8) functions that have been identified in the GIS literature that is applicable to a GIS. The first function (i.e., user interface) can also be recognised as a function of a general IS, the others are more specific to GIS data. The most common functions GIS data functions (recognised by two or more references) are: 1) capture, transfer, validate and edit, 2) management (i.e. GIS data sets in a DBMS), 3) query and analysis, and 4) present (e.g., map or report formats).

## GIS STANDARDS

> *"The aim of standardization in the field of GI* [geographic information] *is to define a family of standards which enables the definition, description, structure, query, encoding, and the transfer of GI and related information. The basic purpose is also to enable GI to be 'delivered' to different users, applications, and systems"* (Salge, 1997, p.160)

According to Crosswell, (1998), standards in the GIS field can be categorised as *independent* (or consensus) standards or *de facto* standards. *Independent* standards are recognised by a group of interested parties, for example, Open GIS consortium and FGDC. *De facto* standards are standards that are accepted because they are popular in the industry. Software vendors usually develop these de facto standards, for example, COM from Microsoft Corporation.

> *"Standards are not an end in themselves but the foundation to make information systems and databases easier to use, more flexible, and less costly" (Crosswell, 1998, p.1-3).*

Standards give the impression of 'open systems' and generally have three portability dimensions (Crosswell, 1998):

1. *Software portability* - software products are capable of running on a multitude of platforms.
2. *Common user interfaces* - software created with a standard set of functions, syntax and graphical user interfaces to allow users to query, analyse and access data, whatever software is being used.
3. *Application portability* - software produced by a set of common languages used to develop programs.

In the context of this study, the most important portability dimension is the *common user interfaces* (also identified by Guptill, 1989) The *user interface* dimension relates to the GIS functions required by the user requirements therefore impacts on the software chosen.

Crosswell (1998) organised the standards into high- and low-level categories (Table 8). Low-level standards are associated with the technical details and issues important to the operability of computer systems. The high-level

standards contain detail specific to spatial information technology that is of more concern to users. Table 8, identifies the spatial standards covered in the high-level category, with their associated links to standard agencies and a description of what is covered:

| High-level Standard Categories | Standard Agency | Description |
|---|---|---|
| User Interface | MOTIF (Unix) and Microsoft windows as basic standards | Industry standard graphic user interfaces with operating system or off-the-shelf applications (Unix and Microsoft).<br>A need for consistency within organisation for custom interface look and feel. |
| Data Format/Data Exchange. | OpenGIS Consortium, Spatial Data Transfer Standard (SDTS) and Digital Geographic Information Exchange Standard (DIGEST) | Spatial data formats and exchange (vector) (SDTS, DIGEST).<br>Spatial data formats and exchange (raster) (SDTS).<br>Attribute data exchange and access (supported by standards such as ODBC and SQL).<br>Interoperability and transparent spatial data access (OpenGIS). |
| Programming And Application Development | American National Standards Institute (ANSI), International Organisation for Standardisation (ISO) standards including Java, Visual Basic (VB) and Delphi programming languages. | Open tool development with access to GIS functions at the programming level (e.g. VB & Delphi).<br>SQL spatial extensions (ANSI and ISO).<br>Increasing use of Internet for spatial data queries, analysis and data distribution with the Java language. |
| User Design Standards | Federal Geographic Data Committee (FGDC) & United States Geological Survey (USGS) | Database schemas,<br>Spatial data coding and classification,<br>Spatial metadata (particularly FGDC),<br>Map design,<br>Map accuracy (particularly FGDC), and<br>Map presentation (particularly USGS). |

**Table 8: Descriptions of the GIS High-Level Standards (Crosswell, 1998)**

The high level standards seen in Table 8 can be associated with functionality of a GIS, but there is no mention of requirements and/or GIS software procurement activities.

An independent standard that Crosswell (1998) did not identify was the European Commission, (2000); "*Guidelines for Best Practice in User Interface for GIS*". Ten European organisations contributed towards this document with the intention to increase stakeholder awareness when developing and customising GIS applications. The main emphasis was on the design of user interaction with

the system. The group believed, "the needs and requirements of real GIS users - a pre-requisite for good user interface design - are not taken into account to a satisfactory degree for the development of GIS applications" (European Commission, 2000, p.9)

The European Commission, (2000) specifically discussed a checklist of requirements that need to be gathered for an application by the users. The general categories are:

- *General requirements:* applicable to generic projects, especially software based projects.
- *Project organisation requirements:* identify roles in the project, participation in the project and steps in the development process.
- *Project outcome requirement:* functions and output required from the application itself.
- *Map acquisition requirements:* acquisition of data required as input by the application to generate necessary output.
- *Derivation process requirements:* deriving of new input data (this may be in the form of a new map) from an analysis performed on existing data.
- *User interface requirements:* extract what the user would prefer for usability purposes, especially help and navigation capabilities of the application.
- *Map visualisation requirements:* map rendering capabilities (i.e., size, colours, fonts, symbols, legends, etc.)
- *Database organisation requirements:* getting the user to identify what attribute data they would like associated with the application.
- *Business application requirements:* specific to the user's organisation domain.

The categories that are typical to any project (i.e., GIS or IS) are general and project organisation requirements. The user interface and specific application requirements can also be applied to both the IS and GIS arenas. The other requirements in this de facto standard appear to be specific to a GIS.

By reflecting on the GIS functionality synthesised in Table 7, these functions are accounted for in the GIS requirements specified by the European Commission (2000).

## SUMMARY

This literature review was divided into four parts that related directly to the research propositions. The four parts were development life cycles, requirements engineering, COTS software procurement and GIS.

***Part I*** described various life cycle models as a way of managing and organising requirements (which is in association with this study's first and second research proposition).

Initially, the *waterfall* model was discussed as being a foundation life cycle that a number of others are based on. To help achieve more accurate user requirements the *prototyping* model was described in this study as this aspect was a major omission in the *waterfall* model. The *incremental* and *evolutionary* models are both recognised strategies for phased development and counteracted some of the shortfalls of the *prototype* and *waterfall* models. The *spiral* model makes use of the *prototyping* and *evolutionary* models, but also incorporates the structure of the *waterfall* model. The main emphasis of the *spiral* model is the risk and evaluation of the project instead of a deliverable product that is emphasised by the other lifecycles. The *iterative* approach was described as a generic approach to project development that gleaned the best from the *waterfall* and the *spiral* models. It consisted of four life cycle phases that six (6) specific requirements activities evolved from particularly in the first two life cycle phases. IEEE, (1997) identified sixty five (65) varying process activities that could be mapped to a life cycle model of the project team's choice. The activity groups of most interest in this study are the categories *requirements* and *software importation*.

***Part II***, identified some confusion in the literature regarding requirements terminology; therefore the following terms have been adopted for the purpose of this study. *Software* requirements are taken in the context of what the software does on behalf of the users (particularly the interactions with the user) and describe the boundary of the system (i.e. inputs and outputs). Software

requirements can be classified into *functional requirements* (what the system should do) and *non-functional requirements* (how the system will operate). As software requirements define functions, the terms *functional* and *non-functional* requirements will be referred to in this study. *System* requirements are taken in the context as requirements that describe the environment and operation of the system to be built without discriminating between the medium (e.g., software or hardware). Finally, requirements specify *what* the system should do, not *how* to do it as in the design process.

***Part III,*** discussed software procurement process activities associated with standard IS requirements processes, and standard IS COTS software acquisition processes required for developing GIS solutions. The first section identified activities associated with COTS software procurement life cycle approaches. This section identified both IS and COTS related activities. The second section discussed instruments used to evaluate and select COTS software. The synthesis of these types of activities from the first and second sections identified the most discussed activities in the literature, being: gathering COTS software information from the marketplace, evaluating the COTS software, identify and prioritising requirements, and selecting COTS software. Table 5 and Table 6 will be used in Chapter 3 (Build A Framework).

***Part IV*** discussed the relationship between the IS and GIS fields particularly with requirements and software procurement activities. Initially, the relationship between COT software and GIS was identified through GIS literature, but activities associated with this process were not discovered. It was identified that the terminology between IS and GIS varies when it comes to requirements engineering type activities (more commonly known as needs analysis or assessment in the GIS field). Functions that are typical to a GIS were discussed and related to the functional requirements in the IS field. GIS standards were discussed to identify any formal approaches to GIS requirements or software procurement type activities. The only standard, European Commission, (2000) identified GIS requirements and also some

standard IS requirements, but did not mention any COTS software procurement type activities.

The common theme identified throughout the GIS literature was the function of *user interfaces*. This appears to be a generic function that is common to both the GIS and IS fields.

# CHAPTER III

# BUILD A FRAMEWORK

## INTRODUCTION

The first step of the Morrison *et al.,* (1997) method-driven research approach is to ***build*** a method (Figure 7). These authors define method as a set of guidelines (methodology) applied to a problem definition to produce a solution (March in Morrison *et al.,* 1997). In this study, the method takes the form of a framework[1] that consists of requirements and COTS software procurement process activities to aid GIS application developers.

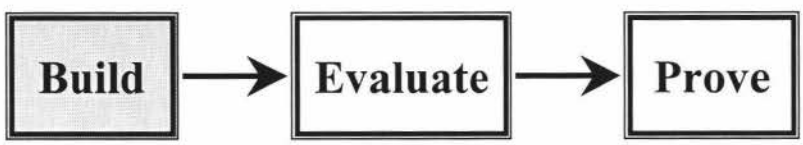


**Figure 7: First Stage of the Method-Driven Research Process (adapted from Morrison et al., 1997)**

This chapter describes how the process activities from the literature review are synthesised into a requirements acquisition and COTS software procurement framework for developing GIS applications.

## SYNTHESIS

The process activities identified in the literature were categorised as IS, COTS software procurement and GIS activities. The framework is based on a synthesis of the iterative life cycle approach and the IEEE, (1997) activities for the IS category, COTS software procurement approaches and COTS software

[1] Framework is defined as "a structure composed of parts framed together, *esp.* one designed for enclosing or supporting anything; a frame or skeleton" (Oxford English Dictionary, 1989, Vol. VI, p. 143).

evaluation instrument activities for the COTS software procurement category and the European Commission (2000) GIS requirements for the GIS category.

The synthesis of these process activities used the structure in IEEE, (1997). This IS standard suggested the developer could add activities that were not included in the standard, but recommends using the mandatory activities specified in the standard. The organisation of the activities in the framework was based on IEEE (1997), activity groups *requirements* and *software importation*.

The *requirements* activity group is described as:

> *"In the development of a system that contains hardware, human, and software components, the Requirements Activity Group follows the development of total system requirements, and the functional allocation of the system requirements to hardware, humans, and software"* (IEEE, 1997, p40).

The *software importation* activity group is described as:

> *"...the software requirements what will be satisfied through importation, to evaluate candidate sources from which the imported software might be obtained, to determine the method of importation, and to import the software, including documentation, into the project"* (IEEE, 1997, p37).

This structure was chosen as it appropriately grouped the activities based on the criteria for the research propositions. The criteria used as the basis of identifying the relevant activities for the framework were:

- Requirements (including software, system, general requirements, GIS requirements, functional, and non-functional)
- COTS procurement activities (including preparation, evaluation, selection and reporting activities).
- Prototype activities.

Prototyping was mentioned explicitly in the literature as a way of gathering and refining requirements; therefore it is included as a separate group in this framework. The prototype activities were grouped separately as the standard did not identify this as an activity group or process activity itself.

With the framework structure set in place, activities could be added under the requirements, software importation and prototype categories. The IEEE (1997) activities were first as they already exist under this scheme. The iterative approach process activities followed and were added to the groupings based on the descriptions of the activity groups in IEEE (1997). These were mostly requirements and prototype activities. The COTS software procurement approach (Table 5) had activities already categorised into requirements and COTS related groups (with the exception of the prototype activities). Placing these activities into the framework was straightforward. The COTS software evaluation instrument activities (Table 6), were placed in the software importation category with exception of one activity, that is, identify and prioritise requirements. They were placed under the requirements category. The GIS activities were all regarded as requirements and added to the this group accordingly.

Table 9 shows the synthesis of process activities placed under each category with associated references.

| Activity Group | Activity | Reference |
|---|---|---|
| Software Importation | Identify imported software requirements | IEEE, (1997) |
| | Evaluate software import sources (if applicable) | IEEE, (1997) |
| | Define software import method (if applicable) | IEEE, (1997) |
| | Import software (if applicable) | IEEE, (1997) |
| | Establish selection criteria | ISO/IEC 14102: 1995 |
| | Identify potential suppliers | IEEE, (1998) |
| | Gather supplier information | ISO/IEC 14102: 1995; Carney (1997); Wallnau et al., (1998) |
| | Identify final candidate COTS software | ISO/IEC 14102: 1995 |
| | Prepare for evaluation | ISO/IEC 14102: 1995 |
| | Evaluate and qualify software | ISO/IEC 14102: 1995; Abts et al., (1999); IEEE, 1998; Carney (1997); Wallnau et al., (1998) |
| | Evaluation reporting | ISO/IEC 14102: 1995 |
| | Prepare for selection | ISO/IEC 14102: 1995 |
| | Apply the selection algorithm | ISO/IEC 14102: 1995 |

| | | |
|---|---|---|
| | Recommend a selection decision | ISO/IEC 14102: 1995; IEEE, (1998); Wallnau et al., (1998) |
| | Validate selection decision | ISO/IEC 14102: 1995 |
| | Prepare contract requirements | ISO/IEC 14102: 1995 |
| | Perform technology search | Maiden & Ncube, (1998); Puma (1999) |
| | Conduct technical evaluations | Maiden & Ncube, (1998); Puma (1999) |
| | Conduct product demonstrations | Maiden & Ncube, (1998); Puma (1999) |
| | Analyse product information | Maiden & Ncube (1998) |
| | Make decisions about product-requirements compliance | Maiden & Ncube, (1998); Puma (1999) |
| | Select COTS software | Maiden & Ncube (1998); Abts et al., (1999); Puma, (1999); Rolland, (1999) |
| Requirements | Define & develop software requirements | IEEE, (1997) |
| | Define interface requirements | IEEE, (1997) |
| | Prioritise and integrate software requirements | IEEE, (1997). |
| | Analyse Problem | Leffingwell & Widrig, (2000) |
| | Obtain initial requirements | Kruchten (2000b) |
| | Understand stakeholder needs | Leffingwell & Widrig, (2000) |
| | Refine requirements for system | |
| | Define the system definition | Leffingwell & Widrig, (2000) |
| | Manage the scope of the system | Leffingwell & Widrig, (2000) |
| | Manage changing requirements | Leffingwell & Widrig, (2000) |
| | Requirements Definition | ISO/IEC 14102: 1995 |
| | Establish system requirements | Abts et al., (1999) |
| | Determine the software requirements | IEEE, (1998) |
| | Identify and prioritise requirements | Maiden & Ncube, (1998); Abts et al., (1999); Puma, (1999); Rolland (1999) |
| | General requirements | European Commission (2000) |
| | Project organisation requirements | European Commission (2000) |
| | Project outcome requirements | European Commission (2000) |
| | Map acquisition requirements | European Commission (2000) |

| | | |
|---|---|---|
| | Derivation requirements | European Commission (2000) |
| | User interface requirements | European Commission (2000) |
| | Map visualisation requirements | European Commission (2000) |
| | Database organisation requirements | European Commission (2000) |
| | Specific application requirements | European Commission (2000) |
| Prototypes | Produce one or several prototypes | Kruchten (2000a) |
| | Prototype developed and demonstrated | Kruchten (2000b); Leffingwell & Widrig (2000); |
| | Prototype the system (including COTS software) | Abts et al., (1999) |

**Table 9: Initial Requirements & Software Procurement Process Activity Framework**

## SYNTHESIS ISSUES

Issues were identified when synthesising the process activities into the initial framework (shown in Table 9). These were: activities in no particular logical sequence, activities spanned across categories, activities were duplicated (i.e., had same outcomes), activity omissions (e.g. non-functional requirements) and the terminology used were inconsistent.

First, groups of activities were organised into sub-groups within the main activity group as these were identified in Table 9. For the *software importation* group, the following sub-groups were identified: product information, evaluation, selection, and import activities. For the *requirements* group, the following sub-groups were: user, software, and system requirement activities. The prototype activity group remained *status quo*. Based on the definition of *software requirements* in the literature (Chapter 2), functional requirements are considered a subset of the software requirements.

Once the activities were placed into subgroups, they were put in order according to the logical sequence identified in the literature (particularly in the standards). Generally, the requirements activities could be seen as being conducted in parallel to each other or with other software importation activities.

Second, two activities could be classified under both requirements and software importation groups as they both reference COTS software requirements. These were *prepare contract requirements* (software importation, No. 16) and *make decisions about product-compliance requirements* (software importation, No.21). These activities will appear in both software importation and requirements categories in the final version of the framework.

Third, there was duplication of activities (from multiple sources); these required combining into more succinct areas. The following is a list of the duplicate activities that required attention:

- Software importation: *identify suppliers* (6), and *perform technology search* (17) have the same outcome and should be combined into one, *search for COTS software*.
- Software importation: *conduct technical evaluation* (18), *evaluate software import sources* (2) and *evaluate and qualify product* (10). The last activity was already a combination of activities identified in the literature review (Chapter 2, COTS Software Procurement Process section), therefore the first two activities will be also classified under *evaluate and qualify product.*
- Requirements: requirements definition (11), obtain initial requirements (5), and identify and prioritise requirements (14), were combined into acquire and prioritise requirements.
- Requirements: define and develop software requirements (1), prioritise and integrate software requirements (3), and determine the software requirements (13), were combined into two disparate activities that considers these requirements: 1) define and develop software requirements, and 2) prioritise and integrate software requirements.
- Requirements: *define the system definition* (8), *establish system requirements* (12) and *refine the requirements for the system* (7) were combined into one activity; define and refine system requirements.
- Requirements: *define the interface requirements* (2) and *interface requirements* (20), were combined into identify the interface requirements.

Fourth, non-functional requirements were activities not identified with functional requirements, but should exist as part of the framework. The main reason for this omission could be contributed to non-functional requirements being classified as a subset of software requirements, therefore not discussed at this level.

Fifth, the usage of verbs describing how the activity was performed varied, but invariably produced similar outcomes. For example, identify (classify, label, name), define (determine, explain), determine (find out, ascertain), establish (prove, organise), acquire (obtain, procure), and obtain (achieve, acquire) requirements. Consequently, the generic phrase used to alleviate confusion was 'identify', 'define', 'acquire' and 'develop'. Another terminology problem was the reference to 'product' and 'software', which meant the same artefact. These artefacts are referred to as *COTS software,* in context with this study.

Table 10 shows the results of the updated framework depicting activity category, sub-group, and process activity with associated reference.

| Activity Group | Activity Sub-Group | Process Activity | Reference |
|---|---|---|---|
| Software Importation | Product information | Identify suppliers | IEEE, (1997) |
| | | Search for COTS software | IEEE, (1998); Maiden & Ncube, (1998); Puma, (1999) |
| | | Gather COTS software information | ISO/IEC 14102: 1995; Carney, (1997); Wallnau *et al.*, (1998) |
| | | Analyse COTS software information | Maiden & Ncube (1998) |
| | | Conduct COTS software demonstration | Maiden & Ncube (1998); Puma (1999) |
| | | Identify final candidate COTS software | ISO/IEC 14102: 1995 |
| | Evaluation | Prepare for evaluation | ISO/IEC 14102: 1995 |
| | | Evaluate and qualify COTS software. | ISO/IEC 14102:1995; Carney (1997); IEEE, (1997); IEEE, (1998); Maiden & Ncube (1998); Wallnau *et al.*, (1998) Abts *et al.*, (1999); Puma, (1999); |
| | | Report evaluation | ISO/IEC 14102:1995 |
| | Selection | Establish selection criteria | ISO/IEC 14102:1995 |
| | | Prepare for selection | ISO/IEC 14102:1995 |
| | | Apply selection algorithm | ISO/IEC 14102:1995 |
| | | Recommend a selection decision | ISO/IEC 14102:1995; IEEE, (1998); Wallnau *et al.*, (1998) |

| | | | |
|---|---|---|---|
| | | Validate selection decision. | ISO/IEC 14102:1995 |
| | | Select COTS software | Maiden & Ncube (1998); Abts *et al.*, (1999); Puma, (1999); Rolland (1999) |
| | Import | Identify imported COTS software requirements, | IEEE, (1997) |
| | | Make decisions about COTS software-requirements compliance | Maiden & Ncube (1998); Puma, (1999) |
| | | Define COTS software import method (if applicable) | IEEE, (1997) |
| | | Prepare contract requirements | ISO/IEC 14102: 1995 |
| | | Import COTS software | IEEE, (1997) |
| Requirements | User | Analyse problem | Leffingwell & Widrig (2000) |
| | | Understand stakeholders needs | Leffingwell & Widrig (2000) |
| | | Acquire and prioritise requirements | Maiden & Ncube (1998); Abts *et al.*, (1999); Puma, (1999); Rolland, (1999) |
| | Software | Define and develop software requirements. | IEEE, (1997) |
| | | Prioritise and integrate software requirements | IEEE, (1997) |
| | | Identify functional requirements:<br>Interface requirements<br>Project organisation requirements<br>Project outcome requirements<br>Map acquisition requirements<br>Derivation requirements<br>Map visualisation requirements<br>Database organisation requirements<br>Specific application requirements (i.e., domain). | IEEE, (1997); European Commission (2000)<br>b-h) European Commission (2000) |
| | | Non-functional requirements | Andriole, (1996); Sommerville, (1996); Dorfman, (1997); Leffingwell & Widrig, (2000) |
| | System | Define and refine system requirements. | Abts *et al.*, (1999) Leffinwell & Widrig (2000); |
| Prototype | | Produce one or several prototypes | Kruchten (2000a) |
| | | Develop and demonstrate prototype | Kruchten (2000b); Leffingwell & Widrig, (2000); |
| | | Prototype the system. | Abts *et al.*, (1999) |

**Table 10: Updated Requirements & Software Procurement Process Activity Framework**

Table 10 identified a number of activities that were referenced more than twice. For software importation these were: *search for COTS software, gather COTS software information, evaluate and qualify software, recommend a selection decision,* and *select COTS software.* For requirements the only activity was *acquire and prioritise requirements.* Table 10 represents the framework that will be evaluated in the next stage of the method-driven research method. (See Chapter 4, evaluate the framework).

## SUMMARY

The process activities identified in the literature review (Chapter 2) were synthesised into a framework (Table 9). A criterion was used (based on research propositions) to identify activities required to build the framework (i.e., *software importation, requirements* and *prototype* activities). There were issues with the synthesis of activities, these were: activity order, activities spanning more than one criterion, duplications, an omission, and confusion with terminology usage. Treating these issues resulted in an updated framework (Table 10), to be evaluated in Chapter 4.

# CHAPTER IV

# EVALUATE THE FRAMEWORK

## INTRODUCTION

The second step of the Morrison *et al.,* (1997) method-driven research approach was to ***evaluate*** a method (Figure 8), by comparing the framework with other approaches.

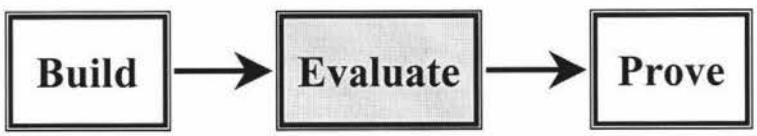


**Figure 8: Second Stage of The Method-Driven Research Process (adapted from Morrison et al., 1997)**

According to Kosters *et al., (*1995), developing a geographic information system (GIS) application requires a systematic, software engineering based development approach for maintaining reliable and efficient software that provides the functionality required by the users. It is not uncommon for IS system development methodologies to be adopted for GIS application development. Benwell, (1994) found because a GIS is similar to an information system (IS), IS development procedures are necessary for the efficiency in the GIS development and can be adapted.

There were difficulties finding current literature to evaluate the framework. In the researcher's experience, the phrase 'time is money' comes to mind and could be a main contributor to the lack of papers published by practitioners in this research area. Papers found were generally published practices performed in the early 1990s, with the exception of an object-oriented approach. This chapter describes research that incorporated requirements and GIS procurement process activities for the developments of GIS applications. This is followed by

a synthesis of process activities revealed in four GIS life cycles that will be compared to the framework ***built*** in Chapter 3.

## GIS DEVELOPMENT PROCESSES

Four life cycles were found in the literature that developed systems in the GIS arena. These are listed in chronological order: the ArcInfo Software Life Cycle (Aronson, 1985), the GIS acquisition model (Clarke, 1991), the GIS system development life cycle (Love, 1991), and the implementation plan (Huxhold & Levinsohn, 1995). The object-oriented approach for developing GISs used objects and patterns by Gordillo, Balaguer, Mostaccio & Das Neves, (1999). This paper was excluded, as these authors only described the *design* aspect of GIS development, which is not part of this study. Particular notice was taken of activities in the life cycle that pertained to requirements and COTS software procurement activities.

### ARCINFO SOFTWARE LIFE CYCLE

According to Aronson, (1985), the majority of GIS applications are large and complex. To develop systems that are well constructed, it is logical to use software engineering methodologies. The methodology discussed by Aronson (1985) was based on lessons learned from previous development of GISs. This methodology is based on the waterfall life cycle model under the Environmental Systems Research Institute (ESRI) situation and was used to development the ArcInfo software in the early 1980s. This model differs to the traditional waterfall model as it has a separate prototyping step conducted in the design stages (Figure 9).

The ArcInfo software life cycle (Figure 9) was used to continue the development of the new releases and updates of the ArcInfo software. This experience portrayed by Aronson (1985), is pertinent when also applied to GIS application development.

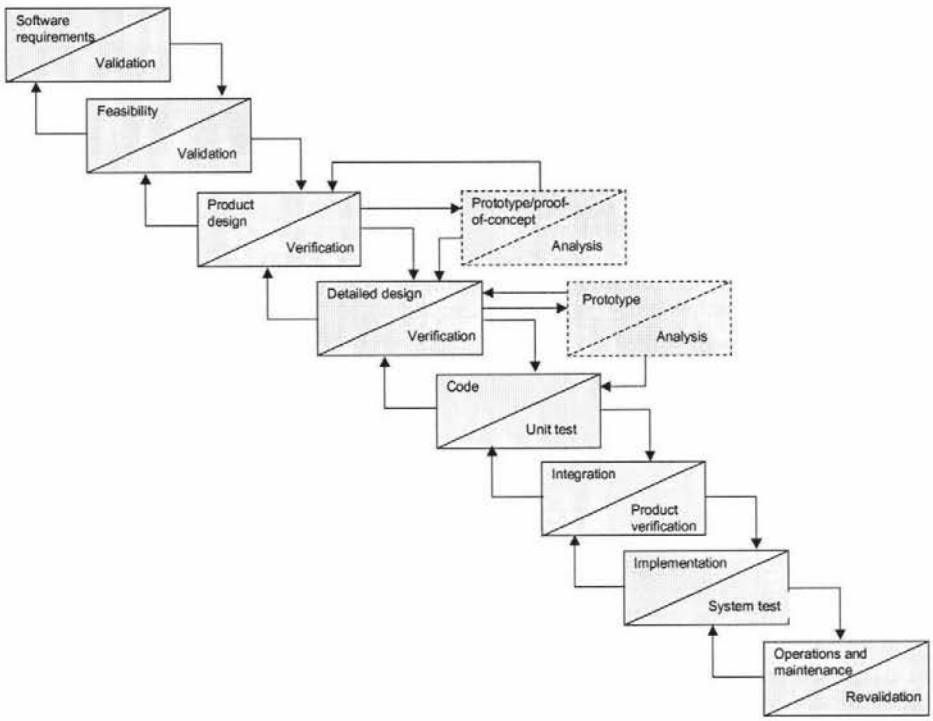


**Figure 9: The ArcInfo Software Life Cycle (Aronson, 1985)**

Two main areas of particular interest were identified in this model: *software requirements* and *prototyping* activities. Due to the nature of this model used for conducting Arc Info software development, there were no COTS software procurement activities expected. *Software requirements* originated from two sources: software modification requests identified by in-house or client users and enhancements identified by the section group or management. According to Aronson (1985), the *prototyping* activities were optional at the design stage and involve two steps: prototype proof of concept and the prototype itself. Both these activities were used to identify questions regarding the design of the product, before final coding. In this case, the requirements stage was used for handling 'bugs' identified in the software. For future software enhancements, prototypes would not be used typically to acquire more requirements or confirm existing ones as seen in other life cycle models.

## GIS ACQUISITION MODEL

Clarke (1991) agreed with Aronson (1985) that GISs are getting more complex and difficult to develop. His reason was the increasing change in GIS commercial off-the-shelf (COTS) software and the number of products available on the market to choose from. Clarke (1991) proposed a GIS acquisition model to help an organisation to analyse requirements, evaluate and implement a product. Clarke's model is a comprehensive framework that included the analysis and requirements specification, evaluation and implementation for the selected GIS.

The GIS acquisition model was based on current systems development and project management techniques of the time, but adapted to the GIS environment. Table 11, illustrates the GIS acquisition model that contains four (4) steps and a total of fourteen (14) activities (with associated activity objective). According to Clarke, (1991), the importance placed on particular steps used, varies depending on the size and complexity of the project.

In this study, the most important steps identified were stages 1 to 3. The *requirements* process activities were identified as steps 1 and 2, analysis of requirements and specification of requirements. Clarke (1991) stated that the first stage is an iterative process that helps identify and refine requirements for determining a business case. The second stage is the analysis of the requirements to produce a specification against which vendor proposals can be evaluated.

The COTS procurement process activities were identified in stage 3 evaluation of alternatives. This stage identified a potential system that is most appropriate for the organisation. According to Clarke, (1991), the increasing technological changes and the number of products to choose, shortlisting products, benchmark testing, and evaluating software help the user select a product from the many available on the market.

| Stage | Activity | Activity Objective |
|---|---|---|
| 1. Analysis of requirements | 1. Definition of objectives | To define the scope and objectives of project and obtain management and user support for them. |
| | 2. User requirements analysis | To determine the user requirements upon which the GIS will be designed and evaluated. |
| | 3. Preliminary design | To develop a preliminary design based on activity 2. This includes database and functional specification, system models and surveying the market for potential systems. |
| | 4. Cost-benefit analysis | To establish a business case for the GIS acquisition proposal and involves: identifying benefits, assessing risks, organisational impact and analysing results. |
| | 5. Pilot study | To provide a GIS working model over a limited geographic area using a wide range of user's data. |
| 2. Specification of requirements | 6. Final design | To produce a design document with finalised database, functional, performance specifications, constraints and generic system requirements. |
| | 7. Request for proposals. | To document containing the final design with contractual requirements of the organisation. |
| 3. Evaluation of alternatives | 8. Shortlisting | To determine a shortlist of feasible systems by evaluating and scoring the information submitted by vendors. |
| | 9. Benchmark testing | To test a potential system by designing a benchmark, develop data and documentation, execute benchmark and analyse results. |
| | 10. Cost-effectiveness evaluation | To evaluate the on cost-effectiveness of the benchmark. |
| 4. Implementation of system | 11. Implementation plan | To ensure smooth implementation and early delivery of benefits by developing a structured implementation plan. |
| | 12. Contract | To integrate the organisation's draft conditions of contract with the vendor's response and the implementation plan to produce a legal document. |
| | 13. Acceptance testing | To ensure that the delivered GIS meets the contracted performance. |
| | 14. Implementation | To implement system by providing training initial data capture and continue performance monitoring. |

**Table 11: The GIS Acquisition Model (adapted from Clarke, 1991)**

## GIS Systems Development Life Cycle

Love (1991) discussed a system development life cycle that Intergraph GIS Services Centre adopted through implementing GIS applications in Asia. This life cycle was based on the traditional system development life cycle (SDLC) or

waterfall model, but concentrated on 'humanistic issues' (i.e., personnel, organisational and political issues) that the author claims has achieved a high degree of success. Love (1991) discussed eight (8) stages to the GIS development life cycle:

1. Implementation plan
2. Familiarisation
3. Pilot study
4. Requirements formulation and analysis
5. System specification and design
6. Database design
7. Program design and development
8. Operational system installation and acceptance testing

The stages that were recognised as requirements processes were stages 2 through to 4. First, stage 2, *familiarisation* has a system definition component where user requirements are initially formulated, analysed, and translated into a preliminary system specification and general system design. Second, stage 3, *pilot study* helped guarantee that the user requirements have been identified and exist in the system definition document. Also user exposure to pilot studies were considered beneficial as they help the user to identify new or changing requirements. Third, stage 4, *requirements formulation and analysis* activity identified user requirements through questionnaires and workshop sessions.

There were two issues identified with this approach. Love (1991) did not include any reference to activities for the procurement of COTS software. This was probably not expected as the development life cycle was based on Integraph's GIS products, therefore not requiring COTS software procurement. Love (1991), like Aronson (1985) included prototype activities in the database design, program design and development stages. Love (1991), did not elaborate on the usage of the prototypes, but referred to them as being iterative.

## IMPLEMENTATION PLAN

Huxhold & Levinsohn (1995) discussed a strategy to accomplish tasks for implementing a GIS in an organisation through processes of an implementation plan. The objective of this implementation plan was to ensure efficiency, cost effectiveness and usefulness of the GIS in the organisation. Each step in the implementation plan depended on the scope and complexity of the GIS. This plan is a little different to the approaches discussed so far, but necessary to include as it still discusses the requirements and COTS software procurement processes.

According to Huxhold & Levinsohn (1995), the implementation plan contained the following categories of tasks: needs analysis, system design, design specifications, hardware and software procurement/installation, data conversion and training. The categories relating to requirements and COT software procurement processes were 'needs analysis' and 'hardware and software procurement/installation'. Huxhold & Levinsohn (1995) describe needs analysis as:

> *"Assessing the needs of the organization, determining the workings of various business units, their information needs and an assessment of how GIS could be applied to identified work"* (p.87).

Needs analysis has been discussed previously in the literature review (section on requirements engineering in Chapter 2), as another term for user requirements. These authors describe the hardware and software procurement process as the selection criteria for acquiring appropriate hardware and software as stated in the design specification.

There was no discussion of prototypes, pilot studies or benchmark tests as ways of eliciting new requirements or aiding the software selection process identified in the other approaches.

## GIS Process Synthesis

To ***evaluate*** the framework built in Chapter 3, initially the process activities identified in the GIS literature (previous section) were synthesised into Table 12. This table illustrates that the activities classified into two process activity groups (evaluation of software and requirements). Table 12 shows these groups with associated activities, sub-activities and references.

| Process Group | Process Activity | Sub-activities | Reference |
|---|---|---|---|
| Evaluation of software | Evaluation of alternatives | 1. Shortlisting<br>2. Benchmark testing<br>3. Cost-effective analysis | Clarke, (1991) |
| | Hardware and software procurement/ installation | | Huxhold & Levinsohn, (1995) |
| Requirements | Software requirements | | Aronson, (1985) |
| | Analysis of requirements | 1. Definition of objectives<br>2. User requirements analysis<br>3. Preliminary design<br>4. Cost-benefit analysis<br>5. Pilot study | Clarke, (1991) |
| | Specification of requirements | 1. Final design<br>2. Request for proposals | Clarke, (1991) |
| | Familiarisation | System definition | Love, (1991) |
| | Pilot study | | Love, (1991) |
| | Requirements formulation and analysis | | Love, (1991) |
| | Needs analysis | | Huxhold & Levinsohn, (1995) |

**Table 12: Synthesis of Process Activities Based on GIS Literature**

Two activity groups identified from the GIS literature (seen in Table 12) are the same as the activity groupings (i.e., software importation and requirements) in Table 10 of the framework ***built*** in chapter 3. The exception is the GIS literature did not discuss the selection of GIS software, only the evaluation aspect. This confirms that these activities identified in the GIS literature are in accordance with the research propositions of this study (propositions 1-3).

There were process activities under the requirements category, that had similar meanings and outcomes, these were: analysis of requirements (Clarke, 1991) requirements formulation and analysis, (Love, 1991), and needs analysis (Huxhold & Levinsohn, 1995). These could be amalgamated into one activity called *requirements formulation and analysis.*

There were two main issues that arose with the outcome of the synthesis, classifying the level of activities and order of activities. The biggest problem with synthesising these activities was classifying the level that each activity should appear on. The prime example is the pilot study activity, that according to Love, (1991), is a main-level activity, but a sub-level activity of analysis of requirements suggested by Clarke, (1991). The pilot study activity could also be classified as a cross-group activity as it helps identify more requirements and exposes the user to vendor software. The second issue, Clarke (1991) identified a particular order in which the activities should be performed. He believed *evaluation of software* comes after the *design* stage proceeded by requirements analysis. In comparison to IEEE, (1997) standard development processes (Table 2) have *software importation* activities prior to the *requirements* activities. The order of these processes will be acknowledged in Chapter 6. With regards to software importation activities, Clarke, (1991) and Huxhold & Levinsohn (1995) were the only authors that identified these types of activities in the GIS literature. Aronson (1985), and Love (1991), did not refer to this process as they discussed development of GISs with specific GIS software.

The synthesis of the four life cycle activities did not highlight any activities exclusive to the development of a GIS system. The main difference acknowledged in the literature between IS and GIS is the spatial nature of a GIS. Its spatial functions and data impact on requirements analysis and selecting software. These types of issues would probably be identified in lower sub-level activities.

## FRAMEWORK EVALUATION

Evaluating the framework was achieved by comparing the IS framework ***built*** in Chapter 3 (Table 10), to the GIS process synthesised in Table 12 of this chapter. The synthesis involved identifying the IS activity group with the closest match possible between the IS framework activities and GIS activities. The next section compares the software importation group followed by the requirements group.

## SOFTWARE IMPORTATION COMPARISON

The GIS activities, *evaluation of alternatives* and *hardware and software procurement/instalment* can be classified under the software importation group of the IS framework. The first GIS activity, *evaluation of alternative* contains sub-activities (i.e, shortlisting, benchmark testing and cost-effective analysis). The shortlisting sub-activity could be classified as the same as the product information sub-group (IS), as it achieves the same results of *identifying suppliers* through to *identifying final candidate software.* Benchmark testing and cost-effective analysis were not identified in the IS framework at all, although part of benchmark testing is evaluating software.

The second GIS activity, *hardware and software procurement/installation* was difficult to classify as its description is very general (i.e., selection criteria for acquiring appropriate hardware and software as specified in the design specification). This GIS activity could be classified under the IS *selection* and *import* sub-categories. The GIS activity does have a sub-activity, *cost effective analysis,* this was described as having a small part in an evaluation process (i.e., evaluating the cost-effectiveness of the benchmark), but not in the same context as *evaluation* in the IS framework.

## REQUIREMENTS COMPARISON

GIS activities *software requirements, specification of requirements, familiarisation, pilot study,* and *requirements formulation* and *analysis* can all be classified under the requirements group of the IS framework. The first GIS activity, *software requirements* can be directly related to *define and develop software requirements* in the IS framework. This GIS activity had no descriptions of what software requirements entail, therefore can be classified generally under the *software requirements* group (IS). The second GIS activity, *specification of requirements,* can be generally classified as an output of the total *requirements* process, as it will be used for the final design and requests for proposals (RFP) as stated in the sub-activities of this GIS activity. The third GIS activity, *familiarisation,* contains a sub-activity (system definition) that relates directly to *system requirements* in the IS framework. The fourth GIS activity, *pilot study,* was controversial as it could be classified as a lower level activity ('pilot study' appeared across levels in Table 12). Pilot studies did not appear in the IS framework at all. The final GIS activity, the amalgamated *requirements formulation and analysis* related to the requirements in the IS framework, but the IS framework does not contain the analysis component.

Prototyping was the only activity in the IS framework that can be indirectly related to the GIS activities. In the IS framework it is used to gather more requirements and performed earlier on in the life cycle, but in the GIS literature it was related to the design and development stages of the GIS development.

Based on the comparison discussion, the updated framework can be seen in Table 18, Appendix B.

## SUMMARY

The first portion of evaluating the framework was to identify GIS literature that discussed similar approaches to the framework developed in Chapter 3. Generally, GIS literature in this area of research was difficult to find. The literature found, discussed four different approaches to GIS development. Two approaches identified the development used specific software, the other two identified some COTS software procurement activities. Most authors referred to the extent of performing each step in their particularly approach, that relied on the size and complexity of the GIS. Pilot studies seemed to be a method used for identifying requirements as well as testing the software for GIS development, compared to prototyping activities.

The IS framework built in Chapter 3 and compared to the GIS process activities in the GIS literature had matching process activities. There were three activities identified in the GIS literature that were not included in the IS framework, these were: benchmark testing, cost-effective analysis and pilot studies. Also, the analysis aspect of requirements procurement were not included in the IS framework.

# CHAPTER V

# CASE STUDY RESEARCH DESIGN

## INTRODUCTION

It is in the researcher's interest to choose an appropriate research strategy and to conduct rigorous research to generate credible findings. This research is at an early, formative stage as it covers an area where few research studies have been performed. Also, the author noticed a lack of guidelines for the development of GIS applications using COTS software in the GIS literature. Case study research was chosen as the research strategy for this study, as it can validate theories from practice and it is useful when there is little research and theory found in the area of interest (Benbasat, Goldstein, & Mead, 1987). Applying a case study research methodology is appropriate to analyse the experience and action of practitioners in context to gain an understanding of how GIS applications are currently developed.

As this study used the Morrison *et al.* (1997) method-driven research approach (Figure 10), the case study strategy was used to validate (prove) the framework developed.

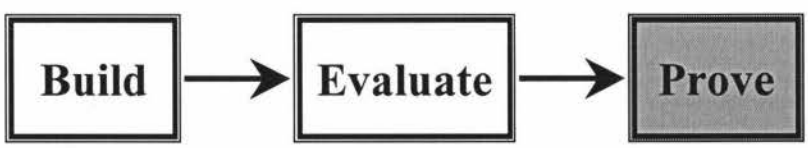


**Figure 10: Third Stage of the Method-Driven Research Process (adapted from Morrison et al., 1997)**

This chapter will initially look at an introduction to case study research followed by the case study design applied in this study. This includes descriptions of how each step of the design was achieved.

## CASE STUDY RESEARCH

Case study research is qualitative research and has been recommended as an appropriate research strategy in the information systems arena (Kaplan & Duchon, 1988; Lee, 1989; Benbasat *et al.*, 1987; Cavaye, 1996; Carroll & Swatman, 2000). This research is based on testing qualitative data in a real-life environment. According to Miles & Huberman, (1994) "[qualitative data] often have been advocated as the best strategy for discovery, exploring a new area, developing hypotheses. In addition we underline their strong potential for testing hypotheses, seeing whether specific predictions hold up" (p.10).

Case study research has been applied to the GIS arena, but what there is, is very difficult to find. "In the GIS community to date, we have seen very few theory-focused empirical studies on issues of acquisition and use of GIS" (Onsrud, Pinto & Azad, 1992, p.43). These authors state that most of the case study literature is *ad hoc* application descriptions of past successes or failures. This type of literature tends to be more *case histories* instead of case studies.

Case study research in IS studies has been recognised as having the following strengths: it is performed in a natural setting (Miles & Huberman, 1994; Yin, 1994; Benbasat *et al.*, 1987); can focus on contemporary events (Yin, 1994; Benbasat *et al.*, 1987); gives research local groundedness (Miles & Huberman, 1994); adds richness and holism (Miles & Huberman, 1994); conducted over a sustained period (hence assess causality) (Miles & Huberman, 1994); subjects and events do not require controlling or manipulating (Yin, 1994; Benbasat *et al.*, 1987); and, provides flexibility (Miles & Huberman, 1994).

Some researchers have resisted case study research. Many researchers have recognised the following difficulties when using the case study approach: contains researcher bias, (Miles & Huberman, 1994), is time consuming to collect and process data, (Miles & Huberman, 1994); produces data overload, (Miles & Huberman, 1994); can be difficult to generalise findings (Lee, 1989; Miles & Huberman, 1994; Yin, 1994); adds complexity (Miles & Huberman,

1994); the adequacy of sampling (i.e., number of cases managed), (Miles & Huberman, 1994); lack of confidence in findings (Miles & Huberman, 1994); "Writes fiction, [is] not science" ( Yin, 1994, p.10; Denzin & Lincoln, 2000, p.8).

These recognised weaknesses are addressed in this research by ensuring the quality of the research design. "The case study, like any other research strategies, is a way of investigating an empirical topic by following a set of pre-specified procedures" (Yin, 1994, p.13). Yin (1994) identified four aspects that must be considered to maintain the design quality of case study research: (a) construct validity, (b) internal validity (for explanatory or causal case studies only), (c) external validity, and (d) reliability. Table 13 shows the tests to maintain validity and reliability and the phases in which the tests occur in a research project.

| Tests | Case Study Tactic | Phase of research in which tactic occurs |
|---|---|---|
| **Construct validity** | • Use multiple sources of evidence<br>• Establish chain of evidence<br>• Have key informants review draft case study report | • Data collection<br>• Data collection<br>• Composition |
| **Internal validity** | • Do pattern-matching<br>• Do explanation-building<br>• Do time-series analysis | • Data analysis<br>• Data analysis<br>• Data analysis |
| **External validity** | • Use replication<br>• Logic in multiple-case studies | • Research design |
| **Reliability** | • Use case study protocol<br>• Develop case study database | • Data collection<br>• Data collection |

**Table 13: Research Validity & Reliability (Yin, 1994)**

Yin (1994, p.33), summarised the case study design tests as follows:

- *Construct validity* - applying the correct operational measures for the concepts being studied.
- *Internal Validity* - establishing a causal relationship, whereby certain conditions are shown to lead to other conditions.
- *External validity* - establishing the domain to which a study's findings can be generalised.
- *Reliability* - demonstrating that the operations of a study can be repeated with the same results.

## CASE STUDY DESIGN

The emphasis of this chapter is to identify an appropriate case study research design to ***prove*** the framework. According to Yin (1994), good research design should include five components:

1. A study's questions,
2. Its propositions, if any,
3. Its unit(s) of analysis[1],
4. The logic linking the data to the propositions, and
5. The criteria for interpreting the findings.

Stating theoretical propositions in the research design is regarded by Yin (1994), as theory development, which is essential to conducting case studies and can be used to build or test theory. According to Cavaye, (1996), proving or testing theory is a deductive approach to using case study design, that is advocated by Yin and Benbasat. "Since deductive logic can be used in both case and survey research, and appropriateness of employing a case research strategy for theory validation is unquestionable" (Cavaye, 1996, p.235).

The other research design components are useful as they force the researcher to think about the techniques used for collecting, analysing and interpreting the case data. Before the techniques are considered the design situations are established by identifying two governing factors single- or multiple-case studies that relate to either unitary or multiple units of analysis (Yin, 1994). Consequently he found that there are generally four types of research designs for case studies, these are: a) single-case (holistic) design, b) single-case (embedded) designs, c) multiple-case (holistic) designs, and d) multiple-case (embedded) designs. According to Yin (1994), embedded case studies occur if there is more than one unit of analysis, in contrast to holistic design, where the case study examines the global nature of the organisation. The researcher must make a decision prior to data collection, whether single- or multiple-case

[1] Unit of analysis relates directly to what the "case" is.

studies will be used to address the research questions and if it will be embedded or holistic (Yin, 1994).

To ***prove*** the framework in this study, the Eisenhardt (1989) case study research process was adopted as it specified explicit steps for conducting case study research from start (stating research questions) through to finish (reaching closure). This 'roadmap' approach has been used to guide researchers understanding of the nature of IS phenomena (Pare & Elam, 1997). "The approach, which adopts a positivist view of research, relies on past literature and empirical data as well as on the insights of the researcher to build incrementally more powerful theories" (Pare & Elam, 1997, p.544).

Table 14 illustrates the Eisenhardt (1989) case study research process and is the foundations for this case study research design. Table 14 shows each step of the process, the activities involved and a justification.

| Step | Activity | Justification |
|---|---|---|
| Getting started | • Definition of research questions<br>• Possibly *a priori* constructs.<br>• Neither theory nor hypotheses | • Focuses efforts<br>• Provides better ground of construct measures.<br>• Retains theoretical flexibility |
| Selecting Cases | • Specified population<br>• Theoretical sampling | • Sharpens external validity<br>• Focuses efforts on cases that replicate or extend theory. |
| Crafting instruments and protocols. | • Multiple data collection methods.<br>• Qualitative and quantitative data combined.<br>• Multiple investigators. | • Strengthens grounding of theory by triangulation of evidence.<br>• Synergistic view of evidence.<br>• Fosters divergent perspectives and strengthens ground. |
| Entering the field | • Overall data collection and analysis.<br>• Flexible and opportunistic data collection methods. | • Speeds analysis and reveals helpful adjustments to data collection.<br>• Allows investigators to take advantage of emergent themes and unique case features. |
| Analysing data | • Within-case analysis<br>• Cross-case pattern using divergent techniques | • Gains familiarity with data and preliminary theory generation.<br>• Forces investigators to look beyond initial impressions. |

| | | |
|---|---|---|
| Shaping hypotheses | • Replication, not sampling, logic across cases.<br>• Search evidence of "why" behind relationships. | • Confirms, extends, and sharpens theory.<br>• Build internal validity. |
| Enfolding literature. | • Comparison with conflicting literature.<br>• Comparison with similar literature. | • Builds internal validity.<br>• Sharpens generalisability. |
| Reaching closure | • Theoretical saturation when possible. | • Ends process when marginal improvement becomes small. |

**Table 14: Case Study Research Process (Eisenhardt, 1989)**

Other case study research design activities from the literature were identified. Yin (1994) described the hypothesis testing research strategy including validity and reliability issues, and Miles & Huberman (1994) and Maykut & Morehouse (1994) outlined data analysis techniques. These activities are included in the following section which is organised according to Eisenhardt's process steps. Each step is described briefly before a discussion of how the step was applied to this study.

## GETTING STARTED

To focus the case study research effort, research questions and propositions were required (introduced in Chapter 1). Defining *research questions* directs the research effort and also helps determine the quantity and type of data collected. "No matter how small our sample or what our interest, we have always tried to go into organizations with a well-defined focus-to collect specific kinds of data systematically" (Mintzberg, 1979, in Eisenhardt, 1989, p.536). 'How' and 'why' questions are typically conducive to a case study effort (Yin, 1994), but he does not preclude other types of research questions. The research question was specified in Chapter 1, but was reiterated in this section to conform to the case study design undertaken.

Eisenhardt (1989) also discussed other activities required for building theory for case study research these included *a priori*[2] constructs and the study requires neither theory nor hypotheses. Eisenhardt (1989) stated that *a priori* constructs could initially help the design of the research as this aids measuring the constructs more accurately; therefore the researchers have firmer empirical grounding for the emergent theory. Having said this, Eisenhardt (1989) also reckoned that including *a priori* constructs is rarely achieved in theory-building research.

Eisenhardt (1989) specified that theory-building research begins with no fixed theory for consideration and no hypothesis to test. "...preordained theoretical perspectives or propositions may bias and limit the findings" (Eisenhardt, 1989, p.536). She does recognise that attempting to achieve this ideal is difficult. In contrast, Yin (1994) suggested stating *propositions* from the research *questions* would bring attention to issues that need close investigation in the study. The advantage of stating propositions provides a better basis on which to construct the data collection design (Pare & Elam, 1997), and also identifies the literature to research. "The design and scoping of a case study research project requires a comprehensive literature analysis to be undertaken in order to understand the existing body of research literature within the research area and to position the research question(s) within the context of that literature" (Darke, Shanks & Broadbent, 1998, p.280).

**Application**

The main research question in this study is:

> *Are requirements engineering, software evaluation and selection practices associated with the COTS paradigm applicable to the development of GIS applications?*

[2] Involving deductive reasoning from a general principle to a necessary effect; not supported by fact; an *a priori* judgement.

In general, the aim of this question was to identify current practices, that is, the process structure and the type of processes used by GIS practitioners.

In this study, the main purpose of case study research design was to ***prove*** the framework (i.e., test the theory) built in Chapter 3 and evaluated in Chapter 4. Based on the research question, propositions were specified to help identify literature available to gain an increased understanding of the problem area.

The propositions for this study are as follows:

1. A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications,
2. Standard IS requirements processes can be used for developing GIS solutions, and
3. Standard IS COTS software procurement processes can be used for developing GIS solutions.

Based on these propositions, the topics studied in the literature were: life cycle models, IS requirement processes, software evaluation and selection processes, and GIS requirements.

## SELECTING CASES

Selecting cases are important as it means specifying the population for the study and defining the limitations for generalising the findings (Eisenhardt, 1989).

> *"As in hypothesis-testing research, the concept of a population is crucial, because the population defines the set of entities from which the research sample is to be drawn"* (Eisenhardt, 1989, p. 537).

Specifying the population impacts on design considerations identified by Yin (1994), that is, single- or multiple-case studies. Single-case studies are useful at the outset of theory generation, which are then used for testing later. It also is appropriate if it is a revelatory case (i.e. the situation is inaccessible to scientific investigation), it represents a critical case for testing theory, and it is an extreme or unique case (Benbasat *et al.*, 1987). The goal of theoretical sampling is to

replicate the study effort, which increases external validity (Yin, 1994; Eisenhardt, 1989). This can be achieved by using multiple-case study design (Yin, 1994). Benbasat *et al.,* (1987), found that conducting a multiple-case study is desirable when the research is theory testing.

To improve the quality of the research design, the unit(s) of analysis must be chosen carefully (Benbasat *et al.,* 1987; Yin, 1994). According to Benbasat *et al.,* (1987), to focus the research, the research questions need to be examined closely as these often indicate the unit(s) of analysis. This will identify whether the research calls for embedded or holistic unit(s) of analysis.

Determining case selection criteria before approaching potential sites is useful for replicating the study effort. "Research on organization-level phenomena would require site selection based on the characteristics of firms" (Benbasat *et al.,* 1987, p.373). Criteria could include: type of industry, size of company, structure of the organisation, geographic location, usage of specific technology and so forth.

According to Benbasat *et al.,* (1987), contacting the organisation requires careful planning. This requires making contact and describing the intentions of the research with relevant employees to gain co-operation. Two points must be addressed before conducting the case studies, confidentiality and benefits.

The researcher addresses the confidentiality issue by providing assurance to the organisation that information disclosed will not be harmful to the company (Benbasat *et al.,* 1987). The researcher must, also consider the anonymity issue identified by Yin (1994). The anonymity of the case can be raised at two levels: the entire case (or cases) and each individual within the case(s).

According to Benbasat *et al.,* (1987), the benefits to an organisation participating in the research are many and varied. The benefits gained could be any of the following: learning about the organisation from the researcher's feedback or

findings, getting new insight about the industry from the researcher, contributing to knowledge, recognition and publicity.

**Application**

To ***prove*** or test the research propositions, a multiple-case design was chosen. The unit of analysis was classified as holistic being 'the requirements and software acquisition processes' used globally throughout the organisation for GIS application development.

Based on the multiple-case design and the unit of analysis the organisations were selected on the following criteria (as discussed by Benbasat *et al.*, 1987). The organisations must:

- Be GIS organisations.
- Perform GIS application development.
- Use GIS COTS software for application development.

Initially, three case sites were approached, but unfortunately due to time constraints on this research only two were able to participate. The study was conducted at two different sites: Organisation A and Organisation B. Both organisations develop GIS applications using ESRI software products. These organisations use similar development process activities, therefore the activities could be predicted and replicated across the two sites.

Contact was made with each organisation through a person that gave the authority for the research to proceed in the organisation. This person then contacted various individuals that would be interested in participating in this study. Four individuals from the two organisations agreed to participate in this study. These individuals were involved in GIS application development using GIS COTS software and were willing to participate, as long as their identity was kept confidential. Each individual was given a research information sheet and a consent form to sign (see example, Appendix A). The information sheet described the research intentions, how it was to be conducted and researcher's university contact details. The consent form was used as an agreement to the conditions the individual was to participate in. Through these arrangements,

there would be no contrasts made directly between the organisations, only between the four participants who would be referred to as Participant A through to D to keep anonymity.

## RESEARCH INSTRUMENTS & PROTOCOLS

Eisenhardt (1989) referred to crafting instruments and protocols as stating a multiple data collection method, combining qualitative and quantitative data and using multiple investigators when conducting case study research for building theory.

First, there are multiple data collection methods that can be used to in the case study research. Yin (1994), identified six sources with examples and various strengths and weaknesses (Table 15). This table also includes acknowledgement of data collection methods from other authors.

The use of multiple sources of evidence in case study research is commonly referred to as data triangulation. Data triangulation strengthens the grounding of evidence (Benbasat *et al.*, 1987; Yin, 1994; Stake 2000), and can alleviate problems with construct validity as the evidence provides multiple measures of the same phenomenon studied (Yin, 1994). "The rationale is the same as in hypothesis-testing research; that is, the triangulation made possible by multiple data collection methods provides stronger substantiation of constructs and hypotheses" (Pare & Elam, 1997, p. 549).

According to Yin (1994), one of the most important sources of collecting data is through interviews. This is also seen by the number of authors that referenced 'interviews' in Table 15. Yin (1994) suggested that interviews could be open-ended or focused. Open-ended interviews consist of questions in which the response is an insight into the issue being studied, for example, a person's opinion of a matter. Focused interviews are more likely to consist of questions from a case study protocol; therefore more structured. Yins (1994), case study protocol is an instrument that contains rules and procedures for collecting case

study data. Yin advocates that the case study protocol increases the reliability of the research and should be used especially in a multiple-case study design. It contains the following (Yin, 1994): an overview of the project, field procedures, case study questions, and guidelines for the case study report.

| Source of Evidence | Examples | Strengths | Weaknesses |
|---|---|---|---|
| Documentation (Benbasat et al., 1987; Yin, 1994) | • Letters, memoranda<br>• Agendas, announcements, minutes (meetings), reports<br>• Administrative documents - proposals, reports, other internal documents<br>• Formal studies, evaluations<br>• Newspaper and mass media articles | • Stable-can be reviewed repeatedly<br>• Unobtrusive-not created as a result of the case study<br>• Exact-contains names, references, and details of an event<br>• Broad coverage - long span of time, many events, and many settings. | • Retrievability-can be low<br>• Biased selectivity, if collection is incomplete<br>• Reporting bias-reflects (unknown) bias of author<br>• Access-may be deliberately blocked. |
| Archival records (Benbasat et al., 1987; Eisenhardt, 1989; Yin, 1994) | Service records, organisational records, maps and charts, lists, survey data, and personal records. | • [same as above for documentation]<br>• insightful-provides perceived causal inferences | • [same as above for documentation]<br>• accessibility due to privacy reasons |
| Interviews (Benbasat et al., 1987; Eisenhardt, 1989; Yin, 1994; Pare & Elam, 1997) | • Open ended<br>• Focused | • Targeted-focuses directly on case study topic<br>• Insightful-provides perceived causal inferences | • Bias due to poorly constructed questions<br>• Response bias<br>• Inaccuracies due to poor recall<br>• Reflexivity-interviewee gives what interviewer wants to know |
| Direct Observation (Benbasat et al., 1987; Eisenhardt, 1989; Yin, 1994) | Field visits to observe. Observation protocols where a participant may be asked to measure incidences or behaviours of certain types | • Reality-covers events in real time<br>• Contextual-covers context of event | • Time-consuming<br>• Selectivity-unless broad coverage<br>• Reflexivity-event may proceed differently because it is being observed.<br>• Cost-hours needed by human observers |
| Participant-observation (Benbasat et al., 1987; Eisenhardt, 1989; Yin, 1994) | Researcher assumes a variety of roles within a case study situation and may participate in events being studied. | • [same as above for direct observations]<br>• insightful into interpersonal behaviour and motives | • [same as above for direct observations]<br>• bias due to investigator's manipulation of events. |

| | | | |
|---|---|---|---|
| Physical Artefacts (Benbasat et al., 1987; Yin, 1994) | Device, tool, output. | • insightful into cultural features<br>• insightful into technical operations | • selectivity<br>• availability |

**Table 15: Six Sources of Evidence Including Strengths and Weaknesses (adapted from Yin, 1994)**

Second, combining qualitative and quantitative data in case studies can be useful to the research. The data collection methods identified in Table 15, look more conducive towards gathering qualitative data, yet quantitative data may help identify relationships that are not obvious to the researcher (Eisenhardt, 1989). "The qualitative data area useful for understanding the rationale or theory underlying relationships revealed in the quantitative data or may suggest directly theory which can then be strengthened by quantitative support (Jick, 1979, in Eisenhardt, 1989, p.538). Yin (1994), suggested that researchers can use either qualitative, or quantitative, or a combination of both.

Third, Eisenhardt (1989) discussed the use of multiple investigators in case studies. The main advantage of using more that one person to investigate a case is the case can be viewed by multiple perspectives. This can add to the empirical grounding of the hypotheses where conflicting perceptions keep the research from closure.

**Application**

In this study, the research protocol developed was based on Yin's (1994) case study protocol designed for multiple-case studies. The majority of the protocol was built from the framework in Chapter 3. The research protocol for this study can be viewed in Appendix B.

Data triangulation was achieved in this study by using the following data collection methods: interviews, documentation, and participant observation. Participants were *interviewed* using a combination of open-ended and focused questions from the research protocol in Appendix B. The documentation that was actually collected was dependent on the willingness of each of the

organisations involved in the study to make its documentation available to the researcher. The researcher was involved with *participant observation* at one of the organisations chosen for this study.

The data to be collected from the two organisations was planned to be primarily qualitative data from the four participants' interviews. The interview data would be supported by counting the number of times the triangulation data occurred to identify trends.

Although there are advantages to multiple investigators, this was not possible for this study; therefore the one investigator collected all data.

## PILOT STUDY

Yin (1994) suggested, conducting a *pilot study*, as it increases the reliability of the results. A *pilot study* is a 'dress rehearsal' for the real study. It provides information about applying relevant questions and logistics to the field inquiry (Yin, 1994). The data collected from a pilot is important to ascertain the relevance and effectiveness of the interview questions for the study. As Yin (1994) stated:

> *"Thus the pilot data provided considerable insight into the basic issues being studied... [and]...provide information about relevant field questions and about the logistics of the field inquiry"* (p. 75).

### Application

In this study, the selection of the pilot study was chosen on the basis of convenience and geographic location. The participant for the pilot study had worked for a well-known GIS consulting company in New Zealand. Her role in the company was as a contract programmer and was involved in all aspects of GIS application development.

The initial contact was made through a telephone call that explained the nature of the pilot study. A time and meeting place was arranged according to wishes

of the participant. After the purpose of the study and some technical terms were explained, the interview was held.

*Questions In Question*

The interview consisted of two sections of questions for the pilot study (Appendix C) that were based on the research protocol (Appendix B): general and detailed. The *general* questions were 'straight forward' and extracted appropriate information that was relevant to the general development of projects in the participant's organisation. Some questions required clarifying and were amended for the real case studies. This was due to terminology used in the IS field compared to the GIS field (e.g., requirements engineering and what the various IS methodologies are for developing systems).

A question from the interview emerged as an important part of this study:

> *"To your knowledge, is there a methodology available (either in industry or research) for specifically developing GIS applications?"*

The *detailed* questions had some problems. The life cycle questions were unable to extract any information, as the participant's organisation did not have or use a life cycle model or methodology in the development of GIS projects. The participant was able to identify some requirements activities that were instinctively used in development projects. This identified a strong need for alternative questions, to extract information if organisations use unknown or unorthodox development life cycles or methodologies.

The pilot study also ascertained that questions needed to be more explicit to be able to identify processes currently practiced. This resulted in a new structure containing questions regarding the following activities: general, development life cycle, GIS requirements, COTS software acquisition and IS questions (see Appendix D). The final questions were designed to obtain an open-ended response from the participants.

*Pilot Study Issues*

The participant in the pilot study brought up two issues concerning developing GIS projects in the New Zealand environment. These were:

- In New Zealand, GIS consultants and developers of GIS projects were limited in developmental skills and financial flexibility. Invariably the project team consisted of one person; therefore, GIS applications were developed by a "seat of your pants" type approach.
- The participant in the pilot project also experienced problems with the users understanding of a GIS. Generally, the users did not know what they wanted and had very limited knowledge to the capability and functionality of a GIS.

## ENTERING THE FIELD

According to Eisenhardt (1989), flexible and opportunistic data collection methods need consideration at this stage of case study research. Eisenhardt (1989) recommended that data collection and analysis should overlap. Taking field notes as a means of providing a running commentary about a particular case at the time of data collection allows the researcher to reflect on the events; therefore achieving an overlap of collection and analysis. "Overlapping data analysis with data collection not only gives the researcher a head start in analysis but, more importantly, allows researchers to take advantage of flexible data collection" (Eisenhardt, 1989, p. 539). Flexible data collection referred to making alterations to the data collection process. This may include additional questions, data sources or observations. Apparently, Eisenhardt (1989) justified this type of action as being legitimate means for theory building (being an inductive form of inquiry). In the opinion of the researcher, theory or hypotheses testing (being a deductive form of inquiry) would not suite a flexible line of enquiry. Keeping a stable line of enquiry would give the researcher the ability to confirm or validate research propositions.

The approach by Yin (1994), to data collection is to ensure construct validity, where the researcher must adhere to three principles: 1) use multiple sources of

evidence, 2) construct a case study database, and 3) develop a chain of evidence. First, using *multiple sources of evidence* implies data triangulation. "Using multiple methods of data collection, however, offers the opportunity for triangulation and lends greater support to the researcher's conclusions" (Benbasat *et al.*, 1987, p.374). Second, a case study *database* is a way of organising and documenting the collected data. According to Yin (1994), the main point of a central repository for the data is that other investigators can review the evidence directly and not be limited by the written report. Pare & Elam (1997) also discussed the development of a case study database, that included analysis procedures (e.g., coding data for analysing), but placed it at the 'pre-analysis' stage (compared to Yin, 1994). Yin (1994) placed the procedures for analysis not in the database but in the research protocol. Third, a *chain of evidence* allows an external observer to be able to trace the steps of the case study, for example, from conclusions back to research questions (Yin, 1994).

**Application**

In this study, the first activity conducted was to obtain m*ultiple sources of evidence* (data triangulation) as discussed in the research instruments and protocol section. Data triangulation was achieved by gathering data through: 1) interviews, 2) documentation and 3) observation. The last two data sources had limitations. Alterations to the data collection process were kept to a minimum between cases, as this study is for validating the study's framework, not to build theory.

First, the case sites did not allow for the *interviews* to be taped. There was also an agreement to keep each organisation and the participants anonymous. Each participant had access to the questions prior to the interview and before committing to the study.

The particular sites selected impacted on the responses to the COTS software acquisition questions to be used in the interviews. The sites selected for this

study could be classified as software vendor sites, where the application environment would be totally focused on the products developed or distributed from those sites. Consequently, these case study questions would need to be answered by three different classifications of case sites: 1) GIS COTS software vendors/distributors that develop GIS applications, 2) application developers who are independent of what GIS software they can use, and 3) clients who want the GIS COTS software for their application. After careful consideration of the COTS software acquisition questions, some were identified as not being applicable for vendor case sites. These were:

1. Identify suppliers
2. Search for COTS software
3. Gather COTS software information
4. Analyse COTS software information
5. Select COTS software
6. Identify imported COTS software requirements.
7. Define COTS software import method
8. Prepare contract requirements
9. Import COTS software

These types of questions would be better to ask either the client or software independent GIS application developers.

Second, *documentation* was difficult to access in both case sites. The documentation used in this study was training material that focused on aspects of GIS design. It had an emphasis on data modelling using object-oriented analysis and design (OOAD), unified modelling language (UML) notation used to diagram the system and CASE tools to draw diagrams, generate schema and code. An appendix in this manual discussed aspects of using the iterative process (see Chapter 2) and also specified requirements process activities. This training manual is used by both organisations. The excerpt of the documentation that was relevant to the study discussed very briefly ideas and activities of requirements procurement required for geodatabase design.

Third, the researcher did not *observe* all activities related specifically to this study (i.e., framework). The types of activities observed were based on the involvement of the researcher with document development for application projects. These documents ranged from proposals through to test and acceptance plans, where clients were required to contribute and agree to them. Consequently, requirements activities observed were through colleague discussions and activities that appeared in documentation. This was the case with software procurement activities where functions had to be identified in a Gap analysis report (i.e., identifying functions that did not exist in the COTS software; therefore needed customising). This does not mean that activities not observed were not performed by the organisation, but explains the limitation of the data collected through this role of participant-observation.

Second, contact summary sheets suggested by Miles & Huberman, (1994) were used as a form of data collection and analysis overlap. The purpose of the contact summary is the researcher reviews the written field notes by identifying main concepts, themes, issues and questions seen during the contact. According to Miles & Huberman (1994), this is a way that helps the researcher to reflect and ponder the data collected. In this study, contact summary sheets were written up after interviewing the participants. Non-sensitive information will be discussed in the analysis section of Chapter 6.

All the data and information collected in the research was kept in a case study database. The *chain of evidence* can be seen in the case study database as it identifies all the documents related to data collection and analysis and also discussed in this thesis. Documents included in the *case study database* are: research protocol, pilot study data, participant consent forms, case study interview questions and transcripts, research documents, participant contact summary sheets, and data analysis information. As the data contains confidential data it may only be accessed through the researcher in the Department of Information Systems, Massey University, Palmerston North, New Zealand.

# DATA ANALYSIS

As part of the Eisenhardt (1989) research process, data analysis included *within-case* and *cross-case pattern* analysis. According to Eisenhardt (1989), the *within-case* analysis organises large volumes of data collected and gives the researcher good insight into the data before the analysis process proceeds beyond the case. "This process allows the unique patterns of each case to emerge before investigators push to generalize patterns across cases. In addition, it gives investigators a rich familiarity with each case which, in turn, accelerates cross-case analysis" (Eisenhardt, 1989, p.540). She also recognised that there are many techniques to choose from, but that the main aim is for the researcher to get familiar with their data. One technique of preparing the data for within-case analysis is the constant comparative method discussed by Maykut & Morehouse (1994). This approach inductively develops research propositions (theory building). This method can be used to logically link the data to the research propositions as discussed by (Yin, 1994) for theory testing.

According to Eisenhardt (1989), researchers generally process their data poorly. They then generate conclusions based on this limited data. The main goal of *cross-case analysis* is to use the cross-case analysis, by looking at the data in divergent ways.

**Within-Case Analysis**

The constant comparative method is a qualitative data analysis method that familiarises the researcher with their data as well as inductively coding and comparing data simultaneously to gain meaning (Maykut & Morehouse, 1994). This method was first developed by Glaser & Strauss in 1967 and expanded by Lincoln & Guba in 1985 to include the look/feel-alike criterion. Figure 11 illustrates the steps that the researcher performs in the constant comparative method.

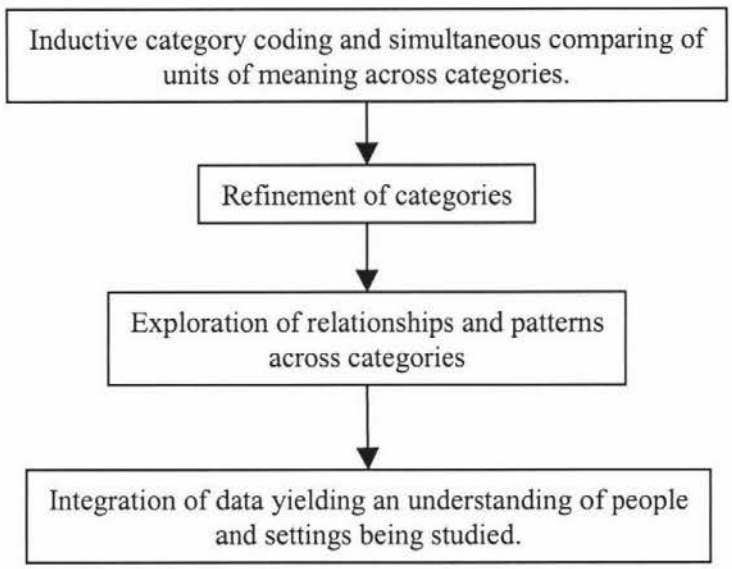


**Figure 11: Constant Comparative Method of Data Analysis (adapted from Maykut & Morehouse, 1994)**

The following are descriptions of the steps involved in achieving the constant comparative method of data analysis (Maykut & Morehouse, 1994):

Data Preparation

- Code data pages based on their original source
- Unitise the data by identifying chunks or units of meaning in the data.
- Identify (discover) the important recurring meanings (i.e. themes, ideas, concepts and so forth) in the data.

Step 1: Constant Comparative Method

- Inductive category coding - the units of data are best placed under the categories identified in the discovery process (above).

Step 2: Refinement of categories

- Writing rules of inclusion - using the look/feel criteria of the categorised data[3], rules/propositions are written to convey the meaning of the data. Data that still requires categorisation will have to meet the rule of inclusion for each category.
- Categorise positive and negative instances

[3] The researcher identifies whether the unit of meaning in one category is very similar to the unit of meaning on another.

- Coding data cards to their categories - once the data has been categorised it requires coding to that category.

Step 3: Exploration of relationships and patterns across categories

- Identify the categories that focus on the study. These are the outcome of this analysis.

Step 4: Integration of data

- Maykut & Morehouse (1994) believed that writing up the study is part of the analysis process.

Chapter 6 applies this method to the data collected in this study.

**Cross-Case Analysis**

The most popular cross-case analysis technique used is pattern matching (Campbell, 1975; Miles & Huberman, 1994; Yin, 1994). Other techniques of data analysis were identified by Miles & Huberman (1994) as tactics for generating meaning: 1) seeing plausibility, 2) clustering, 3) making metaphors, 4) counting, 5) making contrasts and comparisons, 6) partitioning variables, 7) subsuming particulars into the general, 8) factoring, 9) noting relations between variables, 10) finding intervening variables, 11) building a logical chain of evidence, and 12) making conceptual/theoretical coherence.

*Application*

Chapter 6 (part I), contains the results and discussion of these cross-case analysis techniques applied in this study.

## RESEARCH OUTCOME

***Proving*** the framework are the outcomes or results of this study. This section explains why the effort is better than other approaches, thereby contributing towards knowledge or theory. This was achieved using the three steps recommended by Eisenhardt (1989): shaping research propositions, enfolding literature, and reaching closure. Chapter 6 (part II) discusses the results and outcomes of this research.

**Shaping Research Propositions**

According to Eisenhardt (1989), shaping hypotheses requires replication logic across cases and searching evidence for 'why' behind relationships. "Shaping hypotheses in theory-building research involves measuring constructs and verifying relationships" (Eisenhardt, 1989, p.543). This author also stated that this is similar to hypotheses testing research, but it is more judgmental in theory-building research.

Pare & Elam, (1997), state that the replication logic process consists of repeatedly comparing the theory to the data. "A close fit is important to building good theory because it takes advantage of the new insights possible from the data and yields an empirically valid theory" (Eisenhardt, 1989, p541). The approach taken by Yin (1994), was in alignment with theory-testing and discussed replication logic coming from cases selected (particularly to multiple-case studies); either the researcher predicts similar results (literal replication) or produces contrasting results but for predictable reasons (theoretical replication).

*Application*

Chapter 6 (part II) discusses shaping research propositions in this study using the approach by Yin (1994).

**Enfolding Literature**

Enfolding the literature is achieved by comparing the propositions with the literature, essential to theory building (Eisenhardt, 1989). This involves identifying literature that is similar to, or conflicts with the emerging concepts, theory, or hypotheses. Comparing the propositions to other literature is important as it challenges internal validity and generalisability of the research and aids the credibility of the findings (Eisenhardt, 1989).

*Application*

This does not apply to this study, as this suggests inductive research (theory building), not deductive research (testing the propositions),

### Reaching Closure

Research closure is important, knowing when to stop adding cases and when to stop iterating between theory and data (Eisenhardt, 1989). "Theoretical saturation is simply the point at which incremental learning is minimal because the researchers are observing phenomena seen before" (Glaser & Strauss, 1967, in Eisenhardt, 1989).

*Application*

Chapter 6 (part II), discusses shaping research propositions in this study.

## SUMMARY

Table 16 is an audit trail and summary of the processes used for testing theory from case study research. The case study design used the approach by Eisenhardt (1989) as a foundation, but also had additions from other authors, particularly for testing theory. The activities achieved in this study are summarised under four research process steps (i.e., research design, data collection, and data analysis and finishing with the research outcome).

| Research Phase | Step | Activity Results |
|---|---|---|
| Research Design | Getting Started | Research questions (Eisenhardt, 1989, Yin, 1994; Pare & Elam, 1997; Darke et al., 1998)<br>1. Are requirements engineering, software evaluation and selection practices associated with the COTS paradigm applicable to the development of GIS applications?<br>Research propositions (Yin, 1994)<br>1. A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications,<br>2. Standard IS requirements processes can be used for developing GIS solutions, and<br>3. Standard IS COTS software procurement processes can be used for developing GIS solutions. |

| | | |
|---|---|---|
| | Selecting Cases | Research sample (Eisenhardt, 1989; Yin, 1994; Benbasat et al., 1987): Multiple case study<br>Holistic unit of analysis (Yin, 1994): the requirements and software acquisition process.<br>Case study criteria for case selection (Benbasat et al., 1987):<br>-Be GIS organisations.<br>-Perform GIS application development.<br>-Use GIS COTS software for application development.<br>Organisations chosen for study:<br>-Two organisations (in the US and New Zealand). Four anonymous participants from these organisations agreed to conducting this research. |
| | Research Instruments & Protocols | Research Protocol (Yin, 1994)<br>See Appendix B<br>Data Triangulation (Benbasat et al., 1987, Eisenhardt, 1989; Yin, 1994; Pare & Elam, 1997; Stake, 2000): Interviews, documentation, and participant-observation<br>Qualitative and/or quantitative data (Eisenhardt, 1989, Yin, 1994): Qualitative only<br>Multiple investigators (Eisenhardt, 1989): one only. |
| Data Collection | Preparing for data collection | Pilot Study (Yin, 1994)<br>See Appendix C |
| | Entering the field | Multiple Sources Of Evidence (Benbasat et al., 1987, Eisenhardt, 1989; Yin, 1994; Pare & Elam, 1997; Stake, 2000): interview, documentation, and participant observation<br>Collection & analysis overlap (Eisenhardt, 1989; Miles & Huberman, 1994) :Contact summary sheet<br>Collection flexibility (Eisenhardt, 1989)<br>Chain of Evidence (Yin, 1994): seen in case study database and thesis document.<br>Case study database (Yin, 1994; Pare & Elam, 1997)<br>Located at: Information Systems Department, Massey University |
| Data Analysis (Chapter 6, part I) | Analyse Data | Within-case analysis (Eisenhardt, 1989; Pare & Elam, 1997)<br>- Constant Comparative method (Maykut & Morehouse, 1997)<br>- Link data to propositions (Yin, 1994; Maykut & Morehouse, 1997)<br>Cross-case pattern analysis (Eisenhardt, 1989; Yin, 1994; Pare & Elam, 1997)<br>- Pattern matching (Miles & Huberman, 1994; Yin, 1994)<br>- Clustering (Miles & Huberman, 1994)<br>- Data counting (Miles & Huberman, 1994) |
| Research Outcome (Chapter 6, part II) | Shaping Research Propositions | Replication logic (Eisenhardt, 1989; Yin, 1994; Pare & Elam, 1997) |
| | Enfolding literature | Comparing conflicting and similar literature (Eisenhardt, 1989; Pare & Elam, 1997): not applicable |
| | Reaching closure | Theoretical saturation (Eisenhardt, 1989; Pare & Elam, 1997) |

**Table 16: Summary of Case Study Research Processes for Testing Theory (adapted from Eisenhardt, 1989)**

The *research design* was the process of identifying the case study design to use that would impact on the rest of the research processes. The *data collection* stage

identified how the researcher should go into the field to collect the data. The *data analysis* identified techniques for analysing the data from the field. The *research outcome* identified research validity and when to stop the research process. The *data analysis* and *research outcome* steps are delivered in the next chapter (6).

# CHAPTER VI

# PROVE THE FRAMEWORK

## INTRODUCTION

This chapter is divided into two sections: data analysis and research outcome. Both parts discuss approaches taken to ***prove*** the framework.

## PART I: DATA ANALYSIS

Once the data was collected, two analysis methods were identified (Chapter 5), to help ***prove*** the framework. These analysis approaches were: within-case analysis and cross-case analysis. As discussed in Chapter 5, the *within-case analysis* was identified as a method to link the data back to the research propositions. The *cross-case analysis* applied the categorised data from the within-case analysis using three divergent strategies: 1) pattern matching (Miles & Huberman, 1994; and Yin, 1994), 2) data counting (Miles & Huberman, 1994), and 3) clustering data (Miles & Huberman, 1994).

### WITHIN-CASE ANALYSIS

In this study, the constant comparative method (described in Chapter 5) was used to link the data to the research propositions. The documents generated from this technique represent an audit trail of the analysis and reside in the case study database. The following is a description of how these methods were applied in this study.

Initially, the data had to be prepared by collating and coding it according to its original data source, for example, i = interview, d = document, and o = observation. For the interview data each participant's response to a question was also coded according the initials of the participant. By analysing each piece

of data, themes were discovered amongst the data units, these were: requirements, organisation, software evaluation and selection, IS processes, and methodologies. The organisation category consists of data taken from a general question section from the questionnaire used in the interviews.

Step 1: Constant Comparative Method

By using the categories discovered in unitising the data in the previous section, a first attempt was made at placing the data under each category.

Step 2: Refinement of Categories

Reflecting on the data placed in the categories so far, the researcher allowed for 'rules of inclusion' to be made, whereby unplaced data can be categorised according to these rules (see Chapter 5, data analysis section). The rules also allowed for positive and negative instances. The rules of inclusion for each category were:

- Requirements - GIS, COTS and IS requirements used (or not used) for developing GIS applications.
- Organisation - the environment settings for developing GIS applications.
- Software evaluation and selection processes - Processes are (or are not) used for evaluating and selecting software for clients.
- IS processes - IS development activities used (or not used) when developing GIS applications.
- Methodologies - Methodologies that contain steps that are (or are not) used for organising the development of GIS applications.

This process was iterated until all data was categorised by the look/feel criteria. Some of the data overlapped between: 1) requirements and methodology and 2) requirements and software evaluation and selection categories showing relationships between categories.

Step 3: Exploration of relationships and patterns across categories

Each category was then matched to the research propositions already identified in this study. The following is the outcome of this analysis based on the propositions:

Proposition 1: Methodologies and IS processes
Proposition 2: Requirements
Proposition 3: Software evaluation and selection processes

The 'organisation' category was unmatched to a proposition, as this data did not relate directly to life cycles, requirements or evaluating and selecting software. This data is useful for specifying the population for replication purposes (see Part II).

## CROSS-CASE ANALYSIS

The data for cross-case analysis used three divergent strategies for generating meaning: 1) pattern matching (Miles & Huberman, 1994; Yin, 1994), 2) clustering data (Miles & Huberman, 1994), and 3) data counting (Miles & Huberman, 1994).

The first technique, *pattern matching* is one of the most desirable strategies to use for case study analysis (Yin, 1994). Miles & Huberman (1994), state that *pattern matching* can identify "what goes with what" (p.245). *Pattern matching* is useful for comparing empirical patterns with predicted ones. According to Yin (1994), if the patterns overlap it strengthens internal validity.

The second technique, *clustering* is closely related to the pattern matching technique. Qualitative data can be sorted into meaningful *clusters* to understand objects that have similar patterns or characters (Miles & Huberman, 1994). Clustering is a process that relies on the coding of data, comparing it and then aggregating it into groups.

The last technique, *data counting* is based on judgements of qualities made on the number of times a theme recurs (Miles & Huberman, 1994). "The 'number of times' and 'consistency' judgements are based on counting" (Miles & Huberman, 1994, p253). These authors believed that data counting is important to qualitative research for three good reasons: to see rapidly what recurs in a

large batch of data; to verify a hypothesis; and to keep the research analytically honest, protecting against bias.

The cross-case analysis described in this chapter is organised around the three research propositions, with the next section 'setting the scene' by discussing the factors associated with the sample population for this study.

**Sample Population**

The 'organisation' data extracted (from the within-case analysis) contained information about the GIS development environment from the case sites chosen. There were four important factors identified as useful for specifying the sample population for this study, they were: consulting area, COTS software, environment, and team size. The *consulting area* referred to the different business domains that the participants were associated with. These business domains could be specific (e.g. hydrology or local government areas) or general (e.g., everything). The *COTS software* factor referred to whether the case sites used the same brand of GIS product or whether the sites used multiple brands of products. The *environment* described whether the site chosen for GIS application development were exclusive GIS vendor sites or a software independent site that conducts GIS application development. The last factor was the number of people involved in a project team. Figure 12, shows a pattern of the sample population of this study.

Each axes of Figure 12 has information pertaining to the four factors described. The first axis, *team size* shows a scale from zero (0) to ten (10). According to the participants there was an average of two (2) to six (6) people per project team (depicted by a double-headed arrow). The *environment* of the multiple case studies consisted of vendor case sites, hence the arrow direction toward the vendor. The GIS COTS software used at these sites were the same products in all application development projects (i.e., ESRI software). The consulting area that each participant worked in varied from specific through to general

business domains. Identifying the sample population of this study is important as it impacts on the research outcome and replication of this study.

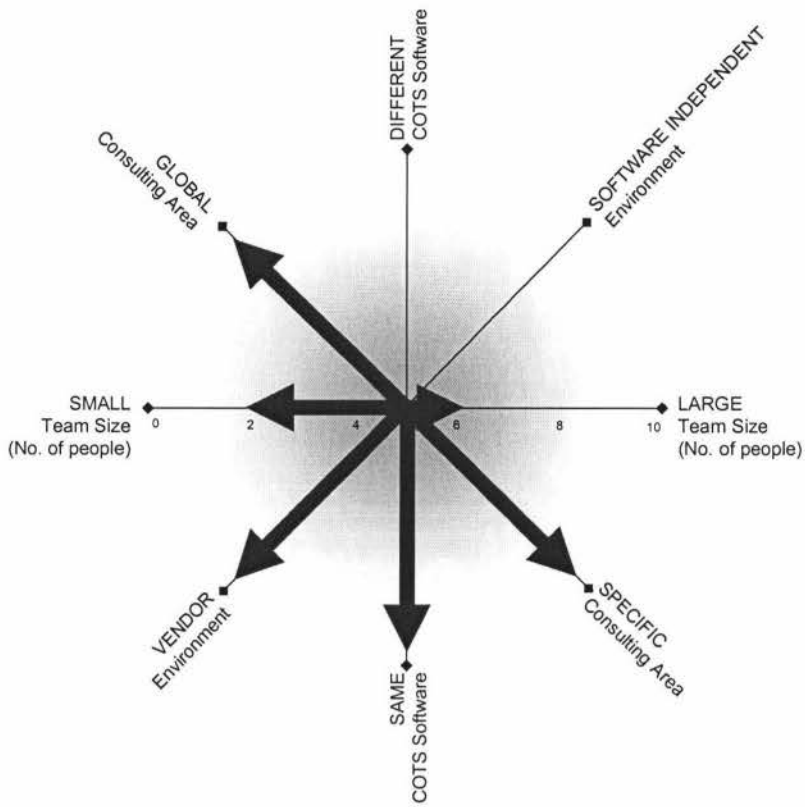


**Figure 12: Sample Population of this Study**

## GIS Application Life Cycle

> *Proposition 1: A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications.*

The data analysed for proposition 1 came from participant interviews, documentation and observations, and were classified under the 'methodology' and 'IS process' categories in the within-case analysis. The open questions in the case interviews extracted data about the steps the participants took when developing GIS applications. The data extracted from the *documentation* and *observation* in this area was sparse. For documentation, this may be due to activities described at a more detailed level compared to the abstract level

identified in life cycle processes. The researcher mainly *observed* the types of documents used for development of applications within a project management environment. Relationships were identified between these documents, that is, functional requirements in the gap analysis, design document, final/revised functional specification, and test and acceptance plan.

Upon examining the *methodologies* data, recurring themes were identified, coded and then clustered into groups of meaning. Figure 13, shows five clustered data groups under the process categories of requirements gathering, requirements analysis, design, development (for applications) and implementation. Data identified within these categories were classified as subgroup activities, for example, under *requirements analysis*, data was counted for three subgroup activities: requirements analysis, application analysis and database analysis. Other data associated with each of these data groups were identified as either input or output activities. Data that was specific to input were associated to only two data categories; these were *requirements gathering* and *development*. All categories had output mainly collections of diagrams/models and documentation.

Participant's description of their development methods flow from indicated life cycle behaviour. This can be seen in Figure 13, from requirements analysis through to implementation. The only iterations identified in the life cycle, according to one participant was the *review* activity in the *requirements gathering* and *development* steps.

Data counting was represented in Figure 13 in the form of bar graphs. These showed the number of times a concept recurred. The *Y*-axis shows the activities associated with the process category the data is associated with. The *X*-axis shows the number of instances in which the data occurred from the three data sources. The maximum number of data collected for each activity could be a total of six (6), that is, four participants, one document source, and one observation data source. These data are shown as different colour saturations

in the graphs: observation - light grey, document - dark grey, and interviewees - medium grey).

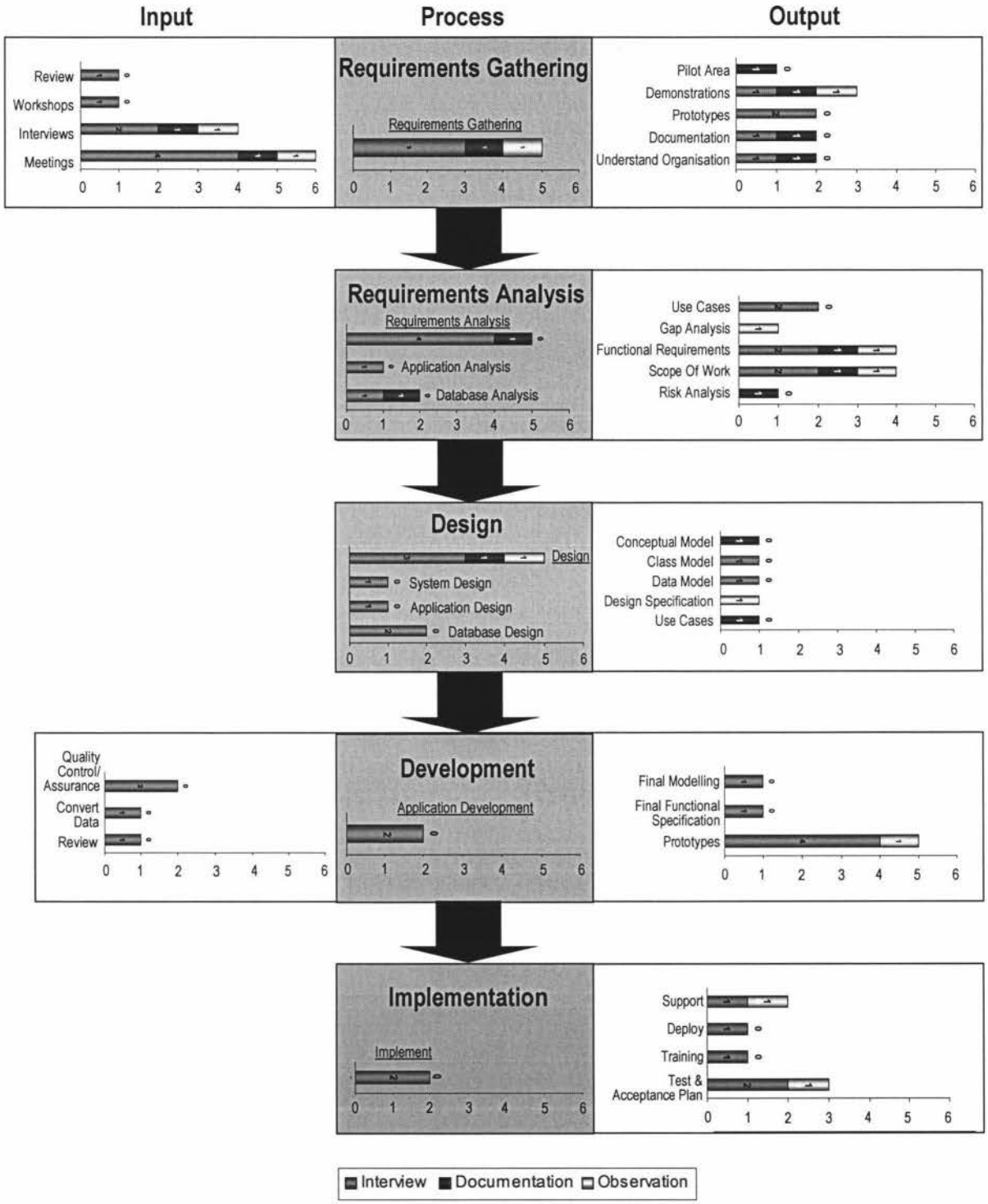


**Figure 13: GIS Application Development Life Cycle Model from Data Triangulation**

The most important processes that relate directly to this study are *requirements gathering* and *requirements analysis*. These categories have the most data and data diversity compared to the other processes. Three participants said they were directly involved with *gathering requirements*. This was also *observed* and found in the *documentation* sources. Interviews and meetings with users and stakeholders were the most common activities performed to gather requirements. Products or already existing applications were usually demonstrated to users and stakeholders to give them ideas for their requirements. One participant recognised that it was important to review the requirements gathering output with the users. The output from *requirements gathering* flows through to the *requirements analysis* process. Data was counted for three activities under the requirements analysis category: requirements analysis (itself), application analysis, and database analysis. Requirements analysis was the one activity performed by all participants and also found in the documentation, but not observed. The functional requirements and scope of work documents were the most referenced documents (i.e., data triangulation consisted of documentation, observation and two participants). The data aggregated for the other activities were scattered throughout the rest of the life cycle, this is probably due to how involved the participant is with this aspect of the application development. One aspect that was not identified in the development of applications processes were activities for the evaluation or selection of COTS software. This absence is discussed in the analysis of data for Propositions 2 and 3.

Prototype activities appeared in the *requirements gathering* and the *development* processes. These processes also included the *review* activity. It is appropriate to reiterate these processes (particularly the prototype activities) for gathering new, or changing user requirements. Participant A referred to prototypes in the *requirements gathering* process as quick mock-up prototypes, consequently giving feedback to the developer for establishing requirements. Participant B also referred to prototyping in the requirements gathering area for identifying the graphical user interface (GUI). The prototype activity in the *development*

process was referenced by all four participants as well as being observed. Therefore, prototyping appears to be performed further into the development of GIS application projects.

*Discussion*

Figure 13 is important to proposition 1 as it illustrates that the approach to GIS application development in practice does follow a life cycle structure. Although half the participants stated they did not follow a specific life cycle model, the other half said they followed the waterfall model concept. The life cycle shown in Figure 13 typically follows a similar structure to the traditional waterfall life cycle model, but with some differences. First, the prototype activities are prominent in this life cycle model compared to the waterfall model; therefore would be more comparable with a prototype life cycle. Second, iteration is not emphasised in this life cycle, whereas the majority of life cycles discussed in the literature review have iteration throughout the life cycle processes. Third, there are several activities that are the same as the activities stated in Boehm's (1988, in Leffingwell & Widrig, 2000) spiral model, these are: risk analysis, prototype, requirements, design, develop, and acceptance test. This life cycle could also adopt the structure of the iterative approach. Each process could be regarded as workflows with iterations of activities throughout the life cycle process.

Data extracted from the documentation used in this study referenced the iterative life cycle model (see Chapter 2, section on the hybrid life cycle model), which consisted of the following life cycle phases: inception, elaboration, construction, and transition. This showed an inconsistency with the participant views to development versus the documentation data. The participant data generally referred to a waterfall life cycle model. Participant D believed that the life cycle approach is "dependent on the size and complexity of the project, your experience, and clients expectations."

All participants stated that they were not aware of any specific GIS methodology used for GIS application development. The only GIS specific

methodology found in the literature was Kosters *et al.,* (1995), GeoOOA approach (see Chapter 2, GIS section). All participants referred to object oriented analysis (OOA)/object oriented design (OOD) as their preferred approach to application development. This appears in the data as output in the form of use cases (under requirements analysis and design processes in Figure 13) and models (particularly class models in the design process of Figure 13). Participant A believed that the GIS field was not big enough to warrant a separate GIS methodology from standard IS methodologies. The tools used for the development of GIS applications were generally Microsoft Visio (three participants and observation), and Microsoft Visual Basic for customisation of the software (two participants, and observation).

**Requirements & COTS Software Procurement**

This section discusses the data analysis for the last two propositions"

> *Proposition 2: Standard IS requirements processes can be used for developing GIS solutions, and*
>
> *Proposition 3: Standard IS COTS software procurement processes can be used for developing GIS solutions.*

The data analysed for this section also came from participant interviews, documentation and observations. The data used was organised initially from the within-case analysis. The data for Proposition 2 was classified under the *requirements* category. The data for Proposition 3 was classified under the *software evaluation and selection processes* category. The case study questions used for the interviews were based on the activities built into a framework in Chapter's 3 and 4. These included open and focused questions (see Appendix D, sections on GIS, COTS and IS), to identify if practitioners actually performed these activities when developing GIS applications. There was minimal data extracted from the documentation or through observations for this area.

The results of using the pattern matching, data clustering and data counting techniques, are shown in a Venn diagram (Figure 14). Each circle of the Venn

diagram represents requirements activities from the IS area (Proposition 2), and software evaluation and selection (Proposition 3), and their relationships to GIS. The relationships of these data groups are shown at the intersections at each circle in the Venn Diagram (Zones 1-4). The criteria for placing the activities in the intersection zones was based on the overlap of coded data between the requirements and software evaluation and selection categories from the within case analysis. Zone 1 shows the requirements activities used in common between GIS and IS, Zone 2 shows the requirements activities used in common between GIS and COTS software procurement, Zone 3 shows the requirements activities used in common between GIS, IS and COTS software procurement, and Zone 4 shows the activities in common between IS and COTS software procurement.

The data counting technique was once again represented in Figure 14 as bar graphs (as in Figure 13). These graphs show the aggregated data for each of the activities in the framework.

The following discussion is based on three individual areas (i.e., IS, COTS and IS) and the zones that intersect showing the activity relationships between these areas.

*Information Systems Data*

The IS requirements activities are shown as the top circle in the Venn Diagram (Figure 14). There were seven (7) activities that relate to the IS requirements area including the intersection zones 1, 3 and 4. There were no activities exclusive to the IS environment, these were all recognised by the COTS and GIS fields. The majority of participants had heavy involvement with activities shown in this area, with the exception of the *software evaluation* activity (Zone 3, showed only three participants who performed this activity). The IS area (including intersections) had the most volume of data collected across the multiple data sources, particularly from the documentation and observation data. The majority of the observation and documentation data showed five

activities in zones 1 and 3, with the exception of one activity (i.e., the evaluation preparation activity under the software evaluation and selection).

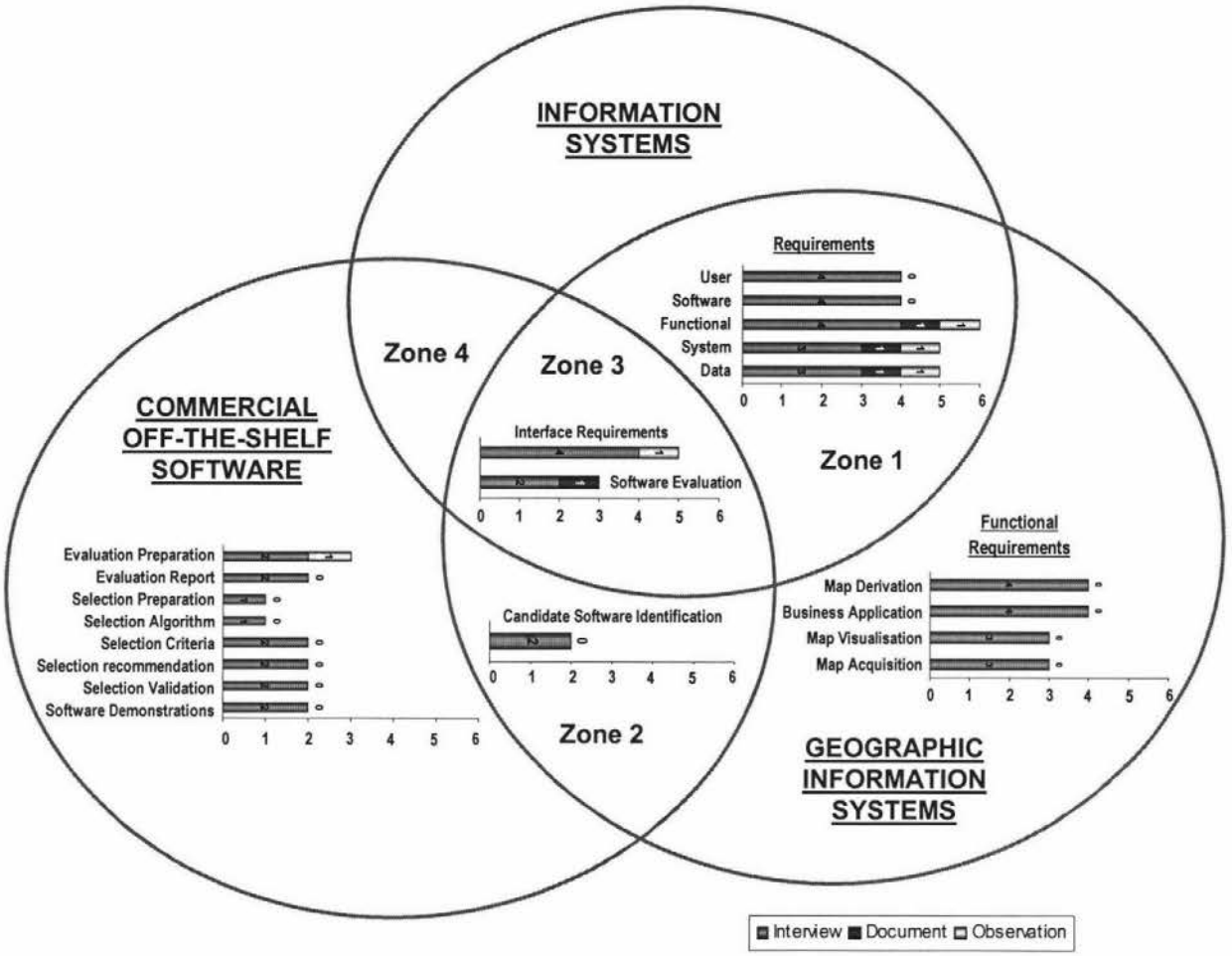


**Figure 14: IS Requirements and Software Procurement Processes and their Relationships with GIS**

*COTS Software Data*

The COTS software procurement activities are shown as the left-hand circle in the Venn Diagram (Figure 14). There were eleven (11) activities that relate to the COTS area including the intersection zones 2, 3 and 4. The COTS circle shows only participant data that supports each of the activities with the exception of *evaluation preparation* that had observation acknowledgement. According to Participant C, "qualifying products for use in an application can be achieved in a prototype."

Data from the documentation was not expected for this area since the documentation is a training manual that used ESRI software. The lack of observation data can also be attributed to this, but also the role of the researcher in the observations did not permit these activities to be observed. Although these activities were not observed it does not mean that these activities are not performed by this organisation.

Overall, two of the four participants performed evaluation and preparation activities, to the extent of informally choosing GIS software modules (from a suite of GIS module) for clients. Participant C stated that "the client selects the vendor first and the vendor helps the client to select a particular software package." Participant D, who stated they were "not normally involved in selection of different GIS software", also supported this view. Just selecting the most appropriate from ESRI's [software] family[1] which is usually straight forward." The other two participants were not involved with the COTS software procurement process at all. Participant A commented in reference to the COTS software procurement process, that these activities were "probably important to the client. This has already happened via the Request For Proposal process for us."

*GIS Data*

The GIS activities are shown as the right-hand circle in the Venn diagram (Figure 14). There were twelve (12) activities that relate to the GIS area including the intersection zones 1, 2, and 3. There are four activities that were considered functional requirements specific to a GIS: map acquisition, map visualisation, business domain and map derivation. This data was extracted only from interview data and ranged from three to four responses for each activity. Two participants stated that they did not gather GIS requirements or they were not directly involved in this process, but a different section in their organisation gathered requirements. There were no specific GIS activities

[1] ESRI's family refers to software modules (components) within GIS software package solutions.

observed or extracted from the documentation, only those that intersects with the other areas shown in Zones 1, 2, and 3.

*Zone 1*

Zone 1 shows the intersection of requirements between IS and GIS. The participants all stated their involvement with user, software and functional requirements, with a majority (3 of 4) being involved with system and data requirements as well. Observation and documentation data were identified for functional, system and data requirements. Functional requirements had good coverage from data triangulation in this section of the study; this was closely followed by system and data requirements.

*Zone 2*

Zone 2 shows the intersection between COTS and GIS areas. There was one activity identified by two participants being *candidate software identification.* According to these participants, this was the activity that they performed to identify the final product, particularly after a prototype was developed (discussed later).

*Zone 3*

Zone 3 shows the intersection between all three areas - IS, GIS and COTS. There were two activities identified in zone 3, these are *interface requirements* and *software evaluation.* All four participants stated their involvement with performing *interface requirements,* this was also observed. Two of four participants said they evaluated their own products for clients, as they were vendor sites. The documentation also recommended that software must be evaluated. Participant C believed that the differences in the evaluation and selection processes of software packages between IS and GIS were small.

*Zone 4*

Zone 4, shows no intersections between IS and COTS activities.

**Other Activities**

Some activities specified in the framework did not fit into the data categories used for the Venn diagram. To explain these omissions, a discussion regarding a comparison between the Venn diagram activities (Figure 14) and life cycle activities (Figure 13) is necessary. By reflecting on life cycle processes it would appear that the activities identified in the framework relate to the requirements analysis stage of the life cycle. The rationale behind this is the requirements gathering activities would need to be conducted first before performing the activities shown in Figure 14.

Two activities in the framework that do not appear in the Venn diagram are prototyping and pilot studies. According to the life cycle prototyping is performed in the *requirements gathering* and *development* stages, not in the requirements analysis stage. This is also the case for pilot studies; it appears in the requirements gathering stage of the life cycle. This pilot study activity under requirements gathering was for defining a pilot area, but the participants did not state when the pilot study was actually performed. Therefore, the likelihood of conducting a pilot study could well be within the *requirements analysis* stage. As Participant B stated that prototyping was useful for assessing risk.

Other activities that appeared in the framework, but did not aggregate data through data triangulation were:

1. Make decisions About COTS Software-Requirements Compliance,
2. Analyse Problem,
3. Understand Stakeholders Needs
4. Project Requirements, and
5. Non-Functional requirements.

The first three activities were unfortunately overlooked when the framework was updated with a new set of IEEE Standards. These activities could be applied in the framework if the study was to be repeated. The fourth activity, *project requirements* was not identified by any of the data triangulation sources.

The participants did not gather requirements specifically under the headings of project outcome or project organisation requirements. These types of activity output may appear in other sections of requirements' documents. The fifth activity, *non-functional* requirements may not have been identified due to the possibility of a terminology problem (as observed by the researcher). What is known as *non-functional* requirements to an IS person are recognised by GIS practitioners as system architecture or configuration requirements; therefore there may be terminology misunderstandings between GIS and IS practitioners. This was shown in the question regarding whether participants conducted functional and non-functional requirements. No one explicitly identified non-functional requirements.

**Discussion**

The *organisation* data shown in Figure 12 determines the data collected hence the research outcome. Particularly, the environment (vendor, independent or client) and whether the same software products are used or different. This is important, as if different software products are used in an application then evaluation and selection of products will be more critical than the same products used.

Proposition 2: the triangulation of data and their relationships showed that the current practice in these organisations, used for this case study research, can be seen to be using standard IS activities for the development of GIS applications. This is particularly noticeable in Zone 1 where the requirements activities used were between GIS and IS. This was reflected generally by two participants' comments that the standard IS methodologies available were sufficient for GIS development. The specific order of these activities was not identified clearly by any of the participants.

Proposition 3: although the participants in this study acknowledged using some of the COTS software procurement activities, these specific activities were not identified in GIS literature, but were referenced in general terms. The GIS

literature predominantly discussed the importance of requirements when acquiring software, but participants identified this as the responsibility of the user organisation. Huxhold & Levinsohn (1995) clearly specified that a project team can decide which GIS software product is the 'best'. The case sites chosen also affected the data collected for this area. It could be said that software evaluation and selection activities are not strongly performed in vendor organisations, unless these developers were required to select a package from one of the software suites or a third-party software package.

## PART II: RESEARCH OUTCOME

This step of the Morrison *et al.,* (1997) method-driven research approach is a continuation of ***proving*** the method (Figure 10), explaining why the effort is better than other approaches, thereby contributing towards knowledge or theory. Using the Eisenhardt (1989) process of building theory from case study research, this chapter describes shaping the propositions and reaching closure.

### SHAPING RESEARCH PROPOSITIONS

Eisenhardt (1989) and Pare & Elam (1997) described replication logic as a tactic to shape hypotheses when building theory. This study's purpose is to test the research propositions proposed in Chapter 1. The approach taken by Yin (1994) to testing theory identified *replication logic* as a study's findings that can be generalised to another similar population. Applying replication logic to this study, the same research design was used over two case studies. The chain of evidence can be seen in the research protocol (Appendix B) and the research database. This raised the question of whether the design could be used at the same sites in the future to achieve the same results.

According to the Oxford English Dictionary (1989), *replication* is defined as "to repeat (an experiment or trial) and obtain a constant result" (Vol, XIII, p.647). The Collins English Dictionary (1994), had a similar definition, but included

that replicating procedures reduced errors. In the context of case study research, Yin (1994) stated:

> *"If two or more cases are shown to support the same theory, replication may be claimed"* (p.31).

The multiple-case sites identified for this study were two organisations from different countries; these sites were carefully chosen to ensure similar results; therefore literal replication. Figure 12, discussed the sample population for this study from the organisation data identified in within-case analysis. To show replication logic (not sampling logic) a comparison was made between the sites shown in Table 17. This brief summary was structured on the research propositions and had to comply with the anonymity requirements of each organisation.

| Research Propositions | Replication Across Cases |
|---|---|
| 1. A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications. | Both sites used a form of a life cycle model for developing GIS applications, even though the activities varied between sites. |
| 2. Standard IS requirements processes can be used for developing GIS solutions. | Both sites tended to adopt IS requirements approaches for developing GIS applications. |
| 3. Standard IS COTS software procurement processes can be used for developing GIS solutions. | Both sites acknowledged that the users chose the software first, before their involvement with the development. The participants helped the user identify a particular software product from their software suite. |

**Table 17: Summary of Replication Across Multiple-Case Study Sites.**

Table 17 highlights the similarities identified across each case site; therefore could this design be applied in the future and return similar results? The biggest changes that would occur over time in organisations like the ones chosen for this study would be due to technological advancement. New releases of GIS COTS software with additional functionality and also tools/software used in the development process. The changes to software functionality would possibly have an impact on the functional requirements, but not the requirements engineering activities. As shown in the cross-case

analysis, software functionality does not impact on these other activities at all. For example, interface requirements between 'man and the machine' will always need to be specified. Also, new or updated development tools would supposedly improve the efficacy of the development, but this would not impact the order in which tasks needed to be completed.

According to Yin, (1994), demonstrating that the operations of a study that can be repeated with the same results impacts on the reliability of a study. The chain of evidence for these purposes can be achieved by using a research protocol (Appendix B), and a case study database (as discussed in Chapter 5). Yin (1994), also found that a research framework that states conditions under which a particular phenomenon is likely to be found is useful for generalising in new cases (Yin, 1994). These reliability tests reside in the framework adapted from the framework developed by Eisenhardt (1989). Another tactic to further increase reliability and validity was identified by Pare & Elam (1997), for the researcher to ask the key participants of the study to review the research document. A draft copy of the research analysis was given to all participants to *review*. The feedback received by Participant C was very favourable. Also, based on informal discussions, Participant D believed that this framework had potential if applied to the right tool to help the developer make decisions about the development of a project.

## REACHING CLOSURE

According to Glaser & Strauss, (1967, in Eisenhardt, 1989) reaching research closure is the point at which learning is minimal because the researcher is observing phenomena seen before. Also, for pragmatic reasons, the research stops due to time, money and opportunity constraints. Time and opportunity were reasons why no more cases were added to this study. From the theoretical saturation viewpoint, the data was compared with the theory by using several techniques in the following order:

1. The *within-case analysis* used the *constant comparative analysis* approach for validating the data collected against the propositions.
2. The cross-case analysis used *pattern matching, data counting* and *data clustering* techniques.
3. By comparing the data collected across the cases to the research propositions was considered *literal replication.* This concept also included a framework that identified reliability tests (i.e., research protocol, case study database and obtaining reviews from participants).

The last step showed that there was no new evidence achieved; therefore reaching theoretical saturation.

## SUMMARY

Part I, discussed the analysis of the case study data using the within-case and cross-case analysis approaches. The within-case analysis used the constant comparative technique to link data via unitised categories to the research propositions. The results of the within-case analysis were used to perform the cross-case analysis. The cross-case analysis approach used pattern matching, data clustering and data counting techniques to analyse data for identifying the study's sample population and research propositions

*Sample population* - the case sites used for this study were portrayed as vendor sites that used software exclusive to those sites. The participants involved with GIS application development had a wide range of business application areas (from specific to general). The project teams were generally small in size, ranging from 2-6 people per team.

*Proposition 1* - requirements analysis is part of a life cycle process and requires input, processing and output (as necessary documents), before the next stage. The most common activities performed are: requirements gathering (input), requirements analysis (process), functional requirements specification (output) and scope of work document (output). The life cycle determined in this

analysis is similar in structure to the waterfall model, but has additional activities that could be adopted in other life cycle approaches.

*Proposition 2* - there were no activities exclusive to IS development, but were included as part of GIS requirements. The requirements specific to GIS were spatial forms of functional requirements. General IS functional requirements can be considered part of the IS and GIS arenas. The participants agreed that there were little differences between IS and GIS; therefore, standard IS requirements activities are performed in GIS application development.

*Proposition 3* - standard IS evaluating and selection software process activities were not necessarily used by GIS software vendor sites. Participants believed the client has already chosen the required software package for the development of GIS applications. The outcome may have been different if the case study sites chosen were either client sites or software independent application developer sites.

Overall, the framework built in this study can be classified as the requirements analysis stage of the life cycle built on participant data.

Part II, described shaping the research propositions and reaching closure. Shaping the propositions looked at the replication logic across the case studies. Reaching closure identified when the research stopped. The main reasons were due to time, money and opportunity constraints.

# CHAPTER VII

# CONCLUSIONS

This study has investigated a number of issues within the area of requirements and software procurement processes for developing GIS applications. There are a large number of IS standards, researchers, and practitioners that specify these processes in detail. GIS practitioners have also undertaken similar exercises, but published work only appears in the early 1990's, and does not specifically identify this research area. Although these GIS practitioners discuss the use of existing IS methodologies at the time, there are differences in the usage of terminology between the two areas.

## THE RESEARCH PROBLEM

The purpose of this research was to prove whether RE practices associated with the COTS paradigm would be suitable to apply in the development of GIS solutions. Research propositions were proposed and tested by using multiple case studies:

1. A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications,
2. Standard IS requirements processes can be used for developing GIS solutions, and
3. Standard IS COTS software procurement processes can be used for developing GIS solutions.

The first proposition was tested using the interview data from the cross-case analysis section of Chapter 6. The majority of the data collected for this proposition came from participant's interviews. Although, the GIS literature did discuss the use of traditional IS life cycle models as a vehicle for performing tasks for software and application development, half of the participants stated they did not follow a specific life cycle model. The case study data did show

characteristics of a life cycle model used specifically for gathering and analysing requirements.

A closer look at the life cycle discussed by Aronson (1985) was necessary as his life cycle is based on the development of ESRI software and is relevant to the sample population of this study (even though this study was conducted seventeen years later). Aronson (1985) based his life cycle on the waterfall model and included software requirements at the beginning of the model before the feasibility stage. This is similar to the GIS application life cycle model built in this study. The prototyping activity in the Aronson (1985) model appeared at the design and coding stages (or development) as a means to prove the concept. This is also true for the GIS application life cycle, with an exception of prototyping being used in the requirements gathering stage for proving the concept, as well as acquiring more requirements. A comparison of life cycles showed that there were no significant changes to the processes of developing of software or applications.

The second and third propositions used the Morrison *et al.,* (1997) method driven approach of building a framework from the IS literature of requirements and software importation activities. This framework was evaluated against GIS literature and tested in the GIS environment using the case study research methodology by Eisenhardt (1989). The case study data came from three different sources: interview, documentation and observation.

An issue arose from building the framework as to the level and order at which the activities were placed within the life cycle model. The activities of concern were the ones recognised in the COTS software procurement area. These activities were very specific in their nature (e.g., apply selection algorithm); therefore making it difficult to identify whether they were either a process or sub-level activity. Overall, the COTS software procurement activities could have been classified into two other sub-categories: evaluation and selection. GIS activities were also implicated by the levelling problem. For example, pilot studies (Table 12) appeared as a main process activity as well as a sub-level

activity of requirements analysis. The order to which the activities should be performed were not identified by the participants, but were placed in accordance of the logical sequence specified in the literature (particularly in the IS standards).

The second proposition looked at requirements gathering and analysing processes identified from a variety of IS literature. Terminology difficulties were encountered from the interviews, as the framework was based in IS terms compared to terms used by participants in the GIS field (e.g., non-functional requirements). Generally, the participants stated that they adopted IS methods for use in their field. Overall, differences between IS and GIS were insignificant. The only activities that would be important to the development of a GIS would require specific GIS functional requirements (e.g., map visualisation requirements). The sample population did not have any significant impact on these types of questions asked of participants. The most volume of participant data was collected based on IS activities from the framework. According to the analysis of this study IS requirements' activities appear to be adopted and well-used by GIS practitioners.

The research outcome of this study showed GIS practitioners using requirements engineering practices. The framework activities for this proposition was based on the requirements category identified in IEEE (1997), with the majority of activities being classified as output of analysing requirements. The gathering of requirements can be ascertained as the initial stages of a project and were better identified in the life cycle compiled from the interview data (in Figure 13). The output of requirements analysis was in the form of a specification document. This process of gathering and analysing requirements is considered requirements engineering as stated by Bubenko (1993, in Sutcliffe & Jarke, 1996). Although the GIS literature does not discuss the term 'requirements engineering', in this study the two gathering and analysing requirements stages of the life cycle (Figure 13), can be considered the RE process for GIS application developers.

The third proposition looked at COTS software procurement processes identified from a variety of IS literature. The GIS literature referred to selecting software, as an important aspect to project development, but did not elaborate the tasks required for performing software acquisition. This was not seen in the case study data, as there was no substantial evidence for evaluating and selecting COTS GIS software. The contributing factor to this weakness was identified as the sample population. The sample population was established as vendor organisations that used the same type of COTS software in all applications. These particular case sites had a significant impact on the questions used in this study. Initially, nine activity types were identified not applicable (Chapter 5, entering the field section), as they were activities that helped a developer identify a software product for a project. In the future these questions would need to be categorised and applied according to GIS application development for: client, vendor and independent consultant. The only GIS application development life cycle activity seen to relate directly to the data for this proposition was the gap analysis document. This document contains information required for customisation of the product to comply with the requirements. Overall, the COTS procurement could be considered a weakness in this study.

A trend identified from perusing the case study data was the prototyping activity. The participants and GIS literature identified that this activity important to the requirements and COTS procurement. It is interesting to note that the IS standards did not identify this activity, but other IS researchers did. The prototyping activity appeared in the requirements gathering and development stages of the GIS application life cycle. Prototyping is an activity that appeared well practiced in the GIS field.

## THE RESEARCH APPROACH

The method used to prove the framework using multiple-case studies was based on the Eisenhart (1989) case study research process for building theory.

This framework identified explicit steps for performing case study research. Although this case study research process is normally used to build theory from case study data, Eisenhardt (1989) and other researchers identified that some of the theory building steps could also be used for theory testing. This did cause some confusion, but in most cases Yin (1994) had theory testing activities to counteract the theory building aspect. Unfortunately, this was not so prevalent in the final research outcome step, particularly with variances identified in replication logic definitions. Generally, the Eisenhardt (1989), research process with the contributions by Yin (1994) was found to be adequate for conducting case study for theory testing.

The issues raised by the pilot study (Chapter 5, pilot study section) did not impact on the multiple-case studies. First, according to the participants, project teams consisted of more than one person (two to five people per team). Second, if users did not know what they wanted in a system, a demonstration of a system or prototypes helped the user identify what they wanted.

Overall, this study could be repeated, replicated and applied to the IS population. The framework (built and evaluated in Chapter's 3 and 4 respectively), consisted of process activities that are not likely to be dependent on technological changes usually faced by organisations. The waterfall model is still being referred to today even though it has gone through various changes from its original conception. If the study was repeated at a later time frame, technology should not impact on the results. This study could also be replicated by using the research protocol in Appendix B. The researcher must take special care with choosing the research population in which to conduct their research. In this case, the researcher must be aware of the implications the research questions may have on the results due to the chosen study population (i.e., client, vendor, and software independent developer). The study could also be applied to the IS field, as the majority of the framework was built from IS research. The only change required to the framework would be the elimination of the GIS component.

## SHORTCOMINGS

The shortcomings seen in this study could be identified in the data triangulation sources. One of the major concerns was the sample population used for the multiple-case studies. Initially, three case sites were chosen as they had similarities (especially for developing GIS projects), but due to time constraints, the third site was unable to contribute to this study. Also, the data collected for the COTS software procurement processes may have been more insightful if the study was conducted at organisations not bound by particular GIS software packages (i.e., not vendor sites).

The data from the observations and documentation was very limited. The capacity of the observer was restricted due to her role in the organisation and not being privy to all aspects encountered in this research. This organisation may conduct these activities; they just were not observed. Also, due to confidentiality reasons, gaining access to documentation was a problem. These issues were discussed in chapters 5 and 6.

There were also difficulties gaining access to participants for conducting interviews. One reason was that time is a scarce resource for practitioners, making access to participants restricted. The other reason was that data extracted from organisations or participants must be treated with care, as this data is seen to identify organisational processes and consequently is guarded rigorously. Unfortunately, this may be more of a problem for case study researchers in the future.

## FUTURE WORK

There are several areas that could be developed from this study. The most obvious addition would be to develop a complete framework for developing GIS applications, incorporating all aspects of the development life cycle. This would also require the breakdown of process activities into further levels of

detail. This framework would require regular updates to be compliant with the latest standards.

Further studies would include repeating this study at different case sites (for example, non-vendor sites). This would identify another sample population from that displayed in this study in Figure 12. The framework would include activities that were not applicable in this study (see Entering Field section, in Chapter 5) and also the activities that were overlooked, that ie, *analysing the problem, understanding stakeholders needs,* and *making decisions about COTS software-requirements compliance* (see Other Activities section in Chapter 6). Continuation from this work would be to conduct a survey (based on the framework) of various GIS organisations to find out the most critical activities used in developing GIS applications.

Further investigation could include the development of tools required by a GIS consultant as guidance in the development of GIS projects. The tool may assist the consultant evaluate the level of difficulty of the problem and help them decide which process activities would be most appropriate for the project. Areas of investigation would be: availability of current tools for decision support systems, determine the processes used in GIS application development, and establish the suitability of the process to the project. A major aspect of this work would be to apply decision making techniques to development processes on projects based on the type of business and application, size and complexity (these being the disciminants described by Kruchten, 2000b in Chapter 2, iterative approach section).

Overall, this study has revealed the importance of conducting requirements processes in GIS application development.

# REFERENCES


Abts, C., Bailey, E., Boehm, B.W. & Brown, A.W. (1999). COCOTS (COnstructive COTS). Retrieved February 1, 2000, from http://sunset.usc.edu/research/COCOTS/index.html

Andriole, S. (1996). Managing System Requirements: Methods, Tools and Cases. New York: McGraw-Hill.

Aronson, P. (1985). Applying Software Engineering To A General Purpose Geographic Information System. Proceedings Auto-Carto 7, American Congress on Surveying and Mapping, Washington D.C., March, 11-14, 23-31.

Benbasat, I., Goldstein, D.K. & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. MIS Quarterly (September), 369-386.

Benwell, G.L. (1994). A System Development Methodology for Geomatics as Derived From Informatics. (Report No. 94/6). Dunedin: University Of Otago, Department of Information Science.

Bonnington, A.K., & Todd, E.G. (2001). Which GIS Product Should I Use? Proceedings from the 29th Annual Conference of AURISA, Melbourne, Australia.

Booch, G., Rumbaugh, J., & Jacobson, I. (1998). The Unified Modeling Language User Guide. Reading: Addison-Wesley Longman, Inc.

Burch, J.G. (1992). Systems Analysis, Design, and Implementation. Boston: Boyd & Fraser Publishing Company.

Campbell, D.T. (1975). Degrees of Freedom and the Case Study. Comparative Political Studies, 8(2), 178-193.

Carney, D. (1997). Assembling Large Systems from COTS Components: Opportunities, Cautions, and Complexities. Retrieved February, 2000, from http://www.sei.cmu.edu:80/cbs/paper...ling-systems/assembling.systems. htm

Carney, D. (1998). Evaluation of COTS Products: Some Thoughts On the Process. SEIinteractive [On-Line serial], 1(2). Retrieved from http://interactive.sei.cmu.edu/Columns/COTS_Spot/1998/September/COTS.sept98.htm

Carney, D. (1999). Requirements and COTS-Based Systems: A Thorny Question Indeed. SEIinteractive [On-Line serial], 2(2). Retrieved from http://interactive.sei.cmu.edu/Columns/COTS_Spot/1999/June/COTS.html

Carrol, J.M. & Swatman, P.A. (2000). Structured-case: A Methodological Framework For Building Theory In Information Systems Research. Proceedings European Conference on Information Systems, Vienna, July 3-5, 116-123.

Cavaye, A.L.M. (1996). Case Study Research: A Multi-Faceted Research Approach For IS. Information Systems Journal, 1996(6), 227-242.

Clarke, A.L. (1991). GIS Specification, Evaluation and Implementation. In D.J. Maguire, M.F. Goodchild, & D.W. Rhind (Eds.), Geographic Information Systems Principles & Applications (Vol. 1, pp. 477-488). New York: Wiley.

Coad, P. & Yourdon, E. (1991). Object-Oriented Analysis (2nd ed.). New Jersey: Yourdon Press.

Collins English Dictionary (1994). (3rd ed., p. 1314). Glasgow: HarperCollins. (Original work published 1991)

Crosswell, P.L. (1998). Spatial Information Technology Standards Concepts, Issues and Industry Status. Spatial Information Technology Standards and System Integration: Tutorial and Annotated Bibliography (Vol. 1, pp. 3-16). Illinois: URISA.

Dangermond, J. (1990). A Classification Of Software Components Commonly Used In GIS. In D.F. Marble & D.J. Peuquet (Eds.), Introductory Readings in GIS's (pp. 30-51). London: Taylor & Francis Ltd.

Darke, P., Shanks, G. & Broadbent, M. (1998). Successfully Completing Case Study Research: combining rigour, relevance and pragmatism. Information Systems Journal, 1998(8), 273-289.

Davis, A.M. (1993). Software Requirements: Objects, Functions, and States. New Jersey: Prentice Hall, Inc.

Davis, A.M. & Leffingwell, D.A. (1995). Using Requirements Management To speed Delivery of Higher Quality Applications. Retrieved July 2, 1999, from http://rational.com/products/reqpro/prodinfo/whitepapers/dynamic.jtmpl

Denzin, N.K. & Lincoln, Y.S. (2000). The Discipline and Practice of Qualitative Research. In N.K. Denzin & Y.S. Lincoln (Eds.), Handbook Of

Qualitative Research (2nd ed., pp. 1-28). Thousand Oaks: Sage Publications.

Dorfman, M. (1997). Requirements Engineering. In R.H. Thayer & M. Dorfman (Eds.), Software Requirements Engineering (2nd ed., pp. 7-22). Los Alamitos: IEEE Computer Society Press.

Eisenhardt, K.M. (1989). Building Theories From Case Study Research. Academy of Management Review, 14(4), 532-550.

European Commission. (2000). Guidelines For Best Practice In User Interface For GIS. European Commission DGIII-Industry ESPRIT Programme, (ESPRIT/ESS Project No. 21580). Retrieved July 2, 2001, from http://www.gisig.it/best-gis.

Faulk, S.R. (1996). Software Requirements: a tutorial. In D. Thayer (Ed.), Software Engineering (pp. 82-103). Los Alamitos: IEEE Computer Society Press.

Goodchild, M.F., Egenhofer, M.J., Kemp, K., David, M.M., & Sheppard, E. (1999). Introduction To the Varenius Project. International Journal Of Geographic Information Science, 13(8), 731-745.

Gordillo, S., Balaguer, F., Mostaccio, C., & Das Neves, F. (1999). Developing GIS Applications With Objects: A Design Patterns Approach. GeoInformatica, 3(1), 7-32.

Guptill, S.C. (1989). Evaluating Geographic Information Systems. Photogrammetry Engineering and Remote Sensing, 55(11), 1583-1587.

Hooks, I.F. (1990). Why Johnny Can't Write Requirements. Retrieved March 29, 1999 from http://www.complianceautomation.com/writingreqs.

Hooks, I.F, & Farry, K.A. (2000). Customer-Centered Products: Creating Successful Products through Smart Requirements Management. New York: Amacom.

Huxhold, W.E., & Levinsohn, A.G. (1995). Managing Geographic Information System Projects. New York: Oxford University Press.

IEEE. (1997). IEEE Standard For Developing Software Life Cycle Processes (Std 1074-1997). New York: IEEE, Inc. (Original work published 1995)

IEEE. (1998). IEEE Recommended Practice For Software Acquisition (Std 1062-1998). New York: IEEE, Inc. (Original work published 1993)

ISO/IEC. (1995). Information Technology - Guideline For the Evaluation And Selection Of CASE Tools (ISO/IEC 14102: 1995). New York: IEEE, Inc. (Original work published 1995)

Jackson, M. (1995). Software Requirements & Specification: a lexicon of practice, principles and prejudices. Wokingham, England: Addison-Wesley Publishing Company.

Kaplan, B., & Duchon, D. (1988). Combining Qualitative and Quantitative Methods In Information Systems Research: A Case Study. MIS Quarterly, 12(4), 71-87.

Kerov, A. (2001). Iterative Approach as Basis For Effective Testing [On-line]. Proceedings 14th International Internet & Software Quality Week 2001, San Francisco, California, USA, May29 - June 1. Retrieved February 13, 2002, abstract from: http://www.soft.com/QualWeek/QW2001/papers/9W2.htm

Kosters, G., & Pagel, B. (1996). The GeoOOA-Tool And Its Interface To Open GIS-Software Development Environments. ACMGIS '96 International Workshop On Advances in GIS, Rockville, USA, November 15-16.

Kosters, G., Pagel, B., & Six, H. (1995). Object-Oriented Requirements Engineering For GIS-Applications. Proceedings of 3rd ACM International Workshop On Advances in GIS, Baltimore, Maryland, USA. 61-69.

Kotonya, G. & Sommerville, I. (1998). Requirements Engineering: Processes and Techniques. Chichester: Wiley & Sons Ltd.

Kruchten, P. (2000a). The Rational Unified Process: An Introduction. Reading: Addison-Wesley.

Kruchten, P. (2000b). A Rational Development Process [On-Line serial]. Retrieved July 25, from http://www.rational.com/products/rup/prodinfo/whitepapers/index_bydate.jtmpl

Lee, A.S. (1989). A Scientific Methodology For MIS Case Studies. MIS Quarterly, March, 33-50.

Leffingwell, D., & Widrig, D. (2000). Managing Software Requirements: A Unified Approach. Reading: Addison-Wesley Longman, Inc.

Love, W.R. (1991). GIS Design And Implementation: A Successful Methodology. Proceedings of AURISA 1991, Wellington, New Zealand, November 19-22, 475-484.

McConnell, S. (1996). Rapid Development: Taming Wild Software Schedules. Redmond: Microsoft Press.

Maiden, N.A., & Ncube, C. (1998). Acquiring COTS Software Selection Requirements. IEEE Software, March/ April, 45-56.

Maiden, N.A., & Rugg, G. (1996). ACRE: Selecting Methods For Requirements Acquisition. Software Engineering Journal, May, 183-192.

Maguire, D.J., & Dangermond, J. (1997). The Functionality Of GIS. In D.J. Maguire, M.F. Goodchild, D.W. Rhind (Ed.), Geographical Information Systems: Principles and Applications (Vol. 1, pp. 319-335). New York: Wiley & Sons Ltd.

Maykut, P., & Morehouse, R. (1994). Beginning Qualitative Research: A Philosphic And Practical Guide. London: The Falmer Press.

Miles, M.B., & Huberman, A.M. (1994). Qualitative Data Analysis: An Expanded Sourcebook (2nd ed.). Thousand Oaks: Sage Publications.

Morrison, J., Morrison, M. & Moore, S.A. (1997). System Development In MIS Research: Roles, Approaches And Strategies. Journal Of Computer Informations Systems, 1997(Summer), 13-17.

Ncube, C., & Maiden, N.A.M. (1999). Guiding Parallel Requirements Acquisition and COTS Software Selection. Proceedings IEEE International Symposium on Requirements Engineering, Limerick, Ireland, June 7-11, 133-140.

Nunamaker, J.F., Chen, M., & Purdin, T.D.M. (1991). Systems Development In Information Systems Research. Journal Of Management Information Systems, 7(3), 89-106.

Onsrud, H.J., Pinto, J.K., & Azad, B. (1992). Case Study Research Methods For Geographic Information Systems. URISA Journal, 4, 32-44.

Oxford English Dictionary, (1989). (2nd ed.). Oxford: Clarendon Press.

Pare, G., & Elam, J.J. (1997). Information Systems And Qualitative Research. Proceedings of International Conference On Information Systems And Qualitative Research, Philadelphia, Pennsylvania, USA, May 31- June 3, 542-567.

Puma Systems. (1999). Commercial Off-The-Shelf Systems Evaluation Technique. Retrieved June 1, 2001, from http://www.pumasys.com/cosset.html

PTI (Public Technology, Inc.), Urban Consortium & ICMA (International City Management Association. (1991). The Local Government Guide To Geographic Information Systems: Planning & Implementation. Washington: ICMA.

Robertson, S., & Robertson, J. (1999). Mastering The Requirements Process. Harlow: Addison-Wesley.

Rolland, C. (1999). Requirements Engineering For COTS Based Systems. Information And Software Technology, 1999(41), 985-990.

Ryberg, D. (2000). Source Selection Case Study. Retrieved March, 2000, from http://www.expertchoice.com/casestudies/bdcasestudy.htm

Saiedian, H., & Dale, R. (2000). Requirements Engineering: Making The Connection Between The Software Developer And Customer. Information And Software Technology, 2000(42), 419-428.

Salge, F. (1997). International Standards And The National Mapping Organizations. In D. Rhind (Ed.), Framework For The World (pp. 160-173). Cambridge: GeoInformation International.

Smith, T.R., Menon, S., Staff, J.L., & Estes, J.E. (1987). Requirements And Principles For The Implementation And Construction Of Large-Scale Geographic Information Systems. International Journal Of Geographic Information Systems, 1(1), 13-31.

Sommerville, I. (1996). Software Engineering (5th ed.). Workingham: Addison-Wesley Publishing Ltd.

Sommerville, I., & Sawyer, P. (1997). Requirements Engineering: A Good Practice Guide. Chichester: Wiley & Sons Ltd.

Stake, R.E. (2000). Case Studies. In N.K. Denzin & Y.S. Lincoln (Eds.), Handbook Of Qualitative Research (pp. 435-454). Thousand Oaks: Sage Publications.

Sutcliffe, A., & Jarke, M. (1996). Defining Visions In Context: Models, Processes & Tools For Requirements Engineering. Information Systems, 21(6), 515-547.

Wallnau, K.C., Carney, D., & Pollak, B. (1998). How COTS Software Affects the Design of COTS-Intensive Systems. Retrieved May 9, 1999, from http://jn.sei.cmu.edu/Features/1998/June/COTS_Software/COTS_Software.htm

Wieringa, R.J. (1996). Requirements Engineering: Frameworks For Understanding. Chichester: Wiley & Sons, Ltd.

Yeh, R.T., & Ng, P.A. (1997). Software Requirements - A Management Perspective. In R.H. Thayer & M. Dorfman (Eds.), Software Requirements Engineering (2nd ed., pp. 377-388). Los Alamitos: IEEE Computer Society Press.

Yin, R.K. (1994). Case Study Research: Design And Methods (2nd ed.). Thousand Oaks: Sage Publications.

Yourdon, E. (1989). Modern Structured Analysis. New Jersey: Prentice-Hall, Inc.

# GLOSSARY

***activities:*** a defined body of work to be performed, including input and output information (IEEE, 1997).

***elaboration phase:*** planning the necessary activities and required resources; specifying the features and designing the architecture. The focus is on requirements, but some software design and implementation is aimed at prototyping the architecture, mitigating certain technical risks by trying solutions, and learning how to use certain tools and techniques. The final product is an executable architectural prototype that will serve as the base line for the next phase (Kruchten, 2000a).

***commercial off-the-shelf software:*** software that is defined by a market-driven need, it is commercially available, and whose use has been demonstrated by a broad range of commercial users. (IEEE, 1998)

***construction phase:*** building the product, evolving the vision, developing the architecture and the plans until the completed vision is ready for delivery to its user community. The focus is on design and implementation with the evolution of the initial prototype into the first operational product (Kruchten, 2000a).

***inception phase:*** the good idea of specifying the end-product vision and its business case and defining the scope of the project. The focus is on understanding the overall requirements and determining the scope of the development effort. (Kruchten, 2000a).

***joint application development (JAD):*** is a requirements-definition and user-interface design methodology in which end-users, executives, and developers attend intense off-site meetings to work out a system's details. (McConnell, 1996).

***life cycle model:*** the framework that maps a number of steps to follow (guidance) and the activities associated with those steps. These steps help the development team to meet the project's user requirements, budget and deadline constraints. (IEEE, 1998)

***object oriented analysis (OOA):*** consists of objects, attributes, wholes, and parts, classes and members. The concepts of these components typically apply to the real-world. Coad & Yourdon (1991)

***process:*** is an organised set of activities, which transforms inputs to outputs. (Kotonya & Sommerville, 1998).

***prototype:*** a simulation of a product using either software prototyping tools, or low fidelity whiteboard or paper mock-ups. The purpose of the prototype is to prompt the users for more requirements (Robertson & Robertson, 1999).

***rapid application development (RAD):*** refers to *speedy developments* or *shorter schedules*, the development of software faster than achieved under traditional approaches (McConnell, 1996).

***rational unified process:*** is a software engineering process developed and commercialised by the Rational Software Corporation (1999). It captures some of the best practices of the industry for software development. It is use case driven and takes an iterative approach to the software development life cycle. It incorporates object-oriented techniques, and many of its activities focus on the development of *models*, using the unified modelling language. (Leffingwell & Widrig, 2000)

***requirements gathering/analysis:*** all the activities devoted to the identification of user requirements and analysis of the requirements. To drive additional requirements, documentation of the requirements as a specification, and the validation of the documented requirements against the actual user needs. (Saiedian, & Dale, 2000)

***transition phase:*** transitioning the product to its users, which includes manufacturing, delivering, training, supporting, and maintaining the product until users are satisfied. The focus is on ensuring that the system has the right level of quality to meet objectives; fix bugs, train users, adjust features, and add missing elements. The final product is produced and delivered. Kruchten (2000a).

***scenario:*** a specific sequence of actions or events that illustrates behaviour (Booch, Rumbaugh, & Jacobson, 1998)

***structured analysis and design methodology (SADM):*** a disciplined, engineered approach to the development of systems and software. The features of the structured approach are modelling tools, modularization, top-down decomposition, iteration, parallel activities, and system development automation (Burch, 1992).

***structured approach***: see structured analysis and design methodology.

***system development life cycle (SDLC):*** is a set of prescribed phases and tasks for the development of an information system. Phases of the SDLC include systems planning, systems analysis, general (conceptual) systems design, systems evaluation and selection, detailed (functional) systems design, and systems implementation. (Burch, 1992).

***task:*** is a single path of execution through a program, a dynamic model, or a representation of control flow; being a thread or a process (Booch *et al.*, 1998).

***unified modelling language (UML):*** is a language for visualising, specifying, constructing, and documenting the artefacts of a software system (Booch *et al.*, 1998).

***use cases:*** is a description of a set of sequences of actions/events, that a system performs that yields an observable result of value to an actor. (Booch *et al.*, 1998)

***workflows:*** consists of a logically related set of activities, and each defines how the activities must be sequenced to produce a viable work product, or artefact. (Leffingwell & Widrig, 2000).

# APPENDIX A

## RESEARCH INFORMATION SHEET

REQUIREMENTS GATHERING FOR GIS COMMERCIAL OFF-THE-SHELF SOFTWARE APPLICATIONS

## INTRODUCTION

Adrienne Bonnington is currently studying a Masters degree of Information Science at Massey University in New Zealand, and is carrying out this research.

Adrienne can be currently contacted at Massey University, Private Bag 11 222, Palmerston North Telephone: (06) 356 9099, extension 2852 or Email: A.K.Bonnington@massey.ac.nz.

Adrienne is being supervised by Mrs. Elisabeth Todd who can be contacted at the Institute of Information Sciences and Technology, Massey University, Private Bag 11-222, Palmerston North, New Zealand. (Telephone: +64 6 356 9099, extension 2455 or Email E.Todd@massey.ac.nz)

## THE RESEARCH

This research investigates whether Requirements Engineering (RE) practices associated with the Commercial Off-The-Shelf (COTS) paradigm (as reported in the Information Systems (IS) arena) would be suitable to apply to the development of Geographic Information System (GIS) solutions. I am particularly interested in the synthesis of good Requirements Engineering practice, COTS selection processes, and GIS spatial requirements, and how all of these individual criteria work together.

This research is based on case study methodology and has implications for understanding the suitability of criteria used in the development of GIS solutions using COTS software. The selection of participants for this research is based upon those who manage and conduct the development of GIS solutions, and for this reason, I am interested in the organisation with which you are associated.

Among the issues I wish to examine are:

- What are the general IS and COTS criteria required for performing Requirements Engineering in a GIS application development project.
- Whether there exists a set of common GIS criteria that is sufficient for gathering requirements for a GIS application development project.
- If IS, GIS and COTS criteria are structured in an ordered format that would aid the development of a GIS application development project.

To shed light on these issues I wish to obtain a personal account from individuals about their experience in the development of GIS solutions using COTS software. Therefore, I would be interested in talking to persons who manage GIS solution projects who could provide information about the requirements gathering processes for developing GIS solutions in your organisation. I anticipate that this would involve completing a questionnaire (approximately 0.5 - 1 hour). If the person does not wish to be named in any publications from the research, I would respect this, but the information given by them may not make it possible to guarantee absolute anonymity. The research methodology used to obtain information would be governed by protocols laid down by Massey University's Human Ethics Committee.

This means that prospective participants in the research would have the right:

- to decline to participate;
- to refuse to answer any particular questions;
- to withdraw from the study at any time;
- to ask any questions about the study at any time during participation;
- to provide information on the understanding that your name will not be used unless you give permission to the researcher;
- to be given access to a summary of the findings of the study when it is concluded.

I wish to include information obtained from this research in my Masters thesis and in academic articles.

The information gathered from this research will reside in the case study database and kept at the Department of Information Systems, Massey University, Palmerston North, New Zealand. The database can be accessed by the researcher and the researcher's supervisor.

Please feel free to get in touch with me should you require further information.

Adrienne Bonnington

## CONSENT FORM

I have read the Information Sheet and have had the details of the study explained to me. My questions have been answered to my satisfaction, and I understand that I may ask further questions at any time.

I understand I have the right to withdraw from the study at any time and to decline to answer any particular questions.

I agree to provide information to the researcher on the understanding that my name will not be used without my permission. (*The information will be used only for this research and publications arising from this research project*).

I agree to participate in this study under the conditions set out in the Information Sheet.

**Signed:** .....

**Name:** .....

**Date:** .....

# APPENDIX B

# RESEARCH PROTOCOL

## PURPOSE

Currently in New Zealand, there are a large number of Geographic Information Systems (GIS) solutions being developed for stakeholders (users) using GIS Commercial Off-The-Shelf Software (COTS) products. The author's experience identified a lack of formal procedures in developing GIS solutions with GIS COTS to meet users needs. This lead to projects not satisfying user's expectations and in some cases project failure.

The Information Systems (IS) field has identified that the development of successful IS solutions with COTS requires good Requirements Engineering (RE) practice, therefore, so do GIS projects. GIS and IS products are similar in nature, as they can both input, manipulate, query, analyse, output and communicate digital data. The difference between an IS solution and a GIS solution is that a spatial data dimension is required for a GIS project. Therefore the GIS developer should be able to successfully reflect the stakeholders GIS needs when developing a project with GIS COTS using IS RE techniques.

## KEY FEATURES OF THE CASE STUDY METHOD

An investigation is required to prove whether RE practices associated with the COTS paradigm (as reported in the IS arena) would be suitable to apply in the development of GIS solutions. The literature will be examined in the areas of IS RE best practice for developing software systems, methodologies for evaluating COTS products, and GIS spatial requirements. An investigation of four evaluation instruments will be performed to identify processes for selecting COTS software that will be validated against GIS COTS products. A synthesis of good RE practice, COTS selection processes, and GIS spatial requirements

will be placed into a framework. This framework is to be validated through case studies in GIS organisations that develop GIS solutions.

## ORGANISATION OF THIS PROTOCOL

This protocol contains an overview, field procedures, case study questions and a guide for the case study report.

## PROCEDURES

### INITIAL SCHEDULING OF FIELD VISIT

#### Review of Preliminary Information

Check sites library and any standards based on gathering requirements for GIS related projects. Identify any other documents and ask for permission to use them in this report.

#### Verification of Access Procedures

All sites have already been given the research questions, to help ascertain whether the intended participants would have time available within their normal working hours to participate in the study.

#### Special Documents

One of the sties has required that a confidentiality agreement be entered into. All sites have already been given a Massey University information sheet describing the research and a consent form for participants to sign.

## DETERMINATION OF PERSONS TO BE INTERVIEWED

The case sites that have been selected had to satisfy the following criteria. They had to:

- Be a GIS organisation.
- Perform GIS application development.

- Use GIS COTS software for application development.

## CASE STUDY PROTOCOL AND QUESTIONS

Questions for the investigator. The main questions of this research are:

> *1. Determine whether requirements, software evaluation and selection practices associated with the COTS paradigm are applicable to the development of GIS applications?*

## THE DEVELOPMENT LIFECYCLE MODEL

> ***PROPOSITION 1***:
>
> A life cycle model is used to guide the gathering and analysing of requirements when developing GIS applications.

Sources of Data:

- Project Manager -Interview
- Development lifecycle - Documentation
- Observation

**Sample strategies:**

- Obtain or draw a document explaining or showing the lifecycle model used (including stages).
- Obtain an explanation or description of each stage in the lifecycle model.
- Who uses the lifecycle model?
- Why is this lifecycle model used?
- How is the lifecycle model used?
- How well structured and organised is the requirement's process?

## IS/GIS/COTS Processes

The following is a combination of propositions based on GIS, COTS and IS requirements and software acquisition.

***Proposition 2:***

Standard IS requirements processes can be used for developing GIS solutions.

***Proposition 3:***

Standard IS COTS software acquisition processes can be used for developing GIS solutions.

What Standard IS/GIS/COTS Criteria Is Used?

**Sources of Data:**

- Project Manager -Interview
- Development lifecycle - Documentation

**Sample Strategies:**

- Identify criteria used for gathering IS/GIS/COTS requirements and software acquisition (including descriptions)
- Obtain documentation that identifies IS/GIS/COTS criteria for requirements and software acquisition.
- How are the criteria used?
- Fill in the following table by identifying what criteria is used, and requirements to satisfy the criteria.

Table 18 represents the framework built in Chapter 3 and evaluated in Chapter 4. It serves as the basis for the case study questions to be applied in this study.

| Processes | Do you use the following criteria? | Used? Yes/No | IS/GIS/ COTS or integrated? |
|---|---|---|---|
| COTS Software procurement | Identify suppliers | | |
| | Search for COTS software | | |
| | Gather COTS software information | | |
| | Analyse COTS software information | | |
| | Conduct COTS software demonstration | | |
| | Identify final candidate COTS software | | |
| | Prepare for evaluation | | |

| | | | |
|---|---|---|---|
| | Evaluate and qualify COTS software. | | |
| | Report evaluation | | |
| | Establish selection criteria | | |
| | Prepare for selection | | |
| | Apply selection algorithm | | |
| | Recommend a selection decision | | |
| | Validate selection decision. | | |
| | Select COTS software | | |
| | Identify imported COTS software requirements, | | |
| | Make decisions about COTS software-requirements compliance | | |
| | Define COTS software import method (if applicable) | | |
| | Prepare contract requirements | | |
| | Import COTS software | | |
| Requirements | Analyse problem | | |
| | Understand stakeholders needs | | |
| | Acquire, analyse and prioritise requirements | | |
| | Define and develop software requirements. | | |
| | Prioritise and integrate software requirements | | |
| | Identify functional requirements:<br>• Interface requirements<br>• Project organisation requirements<br>• Project outcome requirements<br>• Map acquisition requirements<br>• Derivation requirements<br>• Map visualisation requirements<br>• Database organisation requirements<br>• Business application requirements (i.e., domain). | | |
| | Non-functional requirements | | |
| | Define and refine system requirements. | | |
| Prototypes/pilot studies | Produce one or several prototypes | | |
| | Pilot Studies | | |

**Table 18: Research Protocol Question Base**

# ANALYSIS PLAN AND CASE STUDY REPORTS

## WITHIN-CASE ANALYSIS

- Constant Comparative Method
- Descriptive Information
- Application

## CROSS-CASE ANALYSIS

- Pattern Matching
- Clustering
- Data Counting
- Descriptive Information
- Application

# APPENDIX C

## PILOT STUDY QUESTIONS

## GENERAL QUESTIONS

1. What is the size of your organisation?
2. What is your role in the organisation?
3. What is your area of consulting (utilities, local government etc.)?
4. Describe the projects that you are/were involved with?
5. How large was/are the project team(s)?
6. What do you define as requirement's gathering/analysis?
7. What skills do you and your team needs for requirement's analysis?
8. What tools do you use for project development?
9. What programming languages and software do you use for project development?
10. What methodology are you using (JAD, RAD, OOA, SADM)?

## DETAILED QUESTIONS

### DEVELOPMENT LIFE CYCLE

11. Does the organisation have a specific lifecycle model that you and your team apply to a project?
12. ***Alternative*** - If no, why?
13. What are the steps in the lifecycle model that you use?
14. Are there any parts of the lifecycle model that you do differently, not do?
    a) And why?
    b) Which steps do you find most useful and important to requirements gathering?

## GEOGRAPHIC INFORMATION SYSTEMS

1. Describe the activities that are specific to gathering requirements related to a GIS for a project?
2. Where in the lifecycle model are these activities applied?
3. What activities are reviewed, and where are they repeated in the lifecycle model?

## COMMERCIAL OFF-THE-SHELF SOFTWARE

1. What products (software/components/modules) do you generally use in a development project?

2. Software Importation Activities
   a) Describe the activities that evaluate and acquire these products for a project?
   b) Explain if any of these activities are directly related to the GIS activities?
   c) Where in the lifecycle model are these activities applied?
   d) What activities are reviewed and where are they repeated in the lifecycle model?

## INFORMATION SYSTEMS/GENERAL

1. Requirements
   a) What are the activities that define software and system requirements of a project?
   b) Explain if any of these activities are directly related to the GIS, COTS or both?
   c) Do these activities apply to any particular part of the lifecycle?
   d) What activities are reviewed and where are they repeated in the lifecycle model?
2. Are there any other activities or issues that have not been discussed?

## ALTERNATIVE QUESTIONS

.....

.....

## FURTHER COMMENTS

.....

.....

## GLOSSARY

***Requirements gathering/analysis:*** all the activities devoted to identification of user requirements, analysis of the requirements to drive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against the actual user needs. [Saiedian, H., & Dale, R. (2000). Requirements Engineering: Making The Connection Between The Software Developer And Customer. Information And Software Technology, 2000(42), 419-428.]

***Commercial off-the-shelf Software:*** software defined by a market-driven need, commercially available, and whose fitness for use has been demonstrated by a broad spectrum of commercial users.
[IEEE. (1998). IEEE Recommended Practice For Software Acquisition (Std 1062-1998). New York: IEEE, Inc. (Original work published 1993)]

***Lifecycle model:*** the framework that maps a number of steps to follow (guidance) and the activities associated with those steps. These steps help the development team to meet the projects user requirements, budget and deadline constraints.
[IEEE. (1997). IEEE Standard For Developing Software Life Cycle Processes (Std 1074-1997). New York: IEEE, Inc. (Original work published 1995)]

# APPENDIX D

## CASE STUDY QUESTIONS

### GENERAL QUESTIONS

1. What is your role in the organisation you work for (e.g. Title)?

.....

.....

2. What is your area of consulting (e.g. utilities, local government etc.)?

.....

.....

3. How many people are employed in your organisation (approximately)?

.....

.....

4. How many people would be in a typical project team?

.....

.....

5. What skills would you or your team need for requirement's analysis for a GIS solution?

.....

.....

.....

.....

6. What tools do you use for developing the GIS solution?

.....

.....

.....

.....

7. What methodology do you use for developing a GIS solution (e.g. Joint Application Development (JAD), Rapid Application Development (RAD), Object Oriented Analysis and/or Design (OOA/OOD), Structure Analysis and Design Methodology (SADM), Rational Unified Process (RUP) etc.)?

.....

.....

.....

.....

.....

.....

8. To your knowledge, is there a methodology available specifically for developing GIS solutions?

Please circle: YES NO

*If YES, please name and describe briefly*.....

.....

.....

*If NO, why?*

.....

.....

## DEVELOPMENT QUESTIONS

9. Do you use a development life cycle model for developing GIS solutions?

Please circle: YES NO

*If YES, please name and describe*

*model*.....

.....

.....

.....

.....

*If NO, go to Question 13*

10. List the main steps in the lifecycle model that you use?

.....

.....

.....

.....

.....

.....

11. How many levels are there in the lifecycle model?

.....

.....

12. Are there any parts of the lifecycle model that you would do differently, and why?

.....

.....

.....

.....

13. If you do not use a life cycle model, what approach(es) do you take to develop GIS projects?

.....

.....

## GEOGRAPHIC INFORMATION SYSTEM (GIS) QUESTIONS

14. Would you perform the following activities (**please read the instructions below table**):

| No. | Activity | Order | Comments |
|---|---|---|---|
| G1 | Acquire requirements for features contained on maps. | | |
| G2 | Acquire requirements for deriving features on maps. | | |
| G3 | Acquire requirements on rendering maps for monitor's, paper, the internet (If other, please specify)..... | | |
| G4 | Acquire requirements for storing geographical features in a database. | | |
| G5 | Acquire requirements for a particular business context. | | |
| | | | |
| | | | |
| | | | |
| | | | |

- Specify the order in which these activities are applied according to your methodology (identify where activities may be repeated)
- Specify other activities.
- Include comments on the activities if you wish.

## COMMERCIAL OFF-THE-SHELF (COTS) QUESTIONS

15. What GIS products do you generally use in a project?

.....

.....

.....

.....

16. Would you perform the following activities (**please read the instructions below**):

| No. | Activity | Order | Related to GIS requirements (tick) | Comments |
|---|---|---|---|---|
| C1 | Establish a selection criteria | | | |
| C2 | Identify potential candidate software product(s). | | | |
| C3 | Identify requirements that satisfy with existing or acquired software. | | | |
| C4 | Prepare for the evaluation of a software product. | | | |
| C5 | Prepare for the selection of a software product. | | | |
| C6 | Evaluate the software. | | | |
| C7 | Report evaluation results. | | | |
| C8 | Apply a selection algorithm. | | | |
| C9 | Recommend a selection decision. | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

- Specify the order in which these activities are applied according to your methodology (identify where activities may be repeated)
- Identify whether these activities are related to GIS requirements gathering.
- Include comments on the activities if you wish.
- Specify other activities.

## INFORMATION SYSTEMS (IS) QUESTIONS

17. Would you perform the following activities (**please read the instructions below**):

| No | Activity | Order | GIS (tick) | COTS (tick) | Comments |
|---|---|---|---|---|---|
| I1 | Define/refine and develop software requirements using specified methodology. | | | | |
| I2 | Define the user, software and hardware interfaces. | | | | |
| I3 | Prioritise and integrate software requirements. | | | | |
| I4 | Manage changing requirements. | | | | |
| I5 | Prototypes | | | | |
| | | | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

- Specify the order in which these activities are applied according to your methodology (identify where activities may be repeated)
- Identify whether these activities are related to GIS and/or COTS requirements gathering.
- Include comments on the activities if you wish.
- Specify other activities.

18. List the main activities that specifically define the project's functional and non-functional requirements?

.....

.....

.....

.....

19. List the main activities that define the project's system requirements (e.g. sub-systems)?

.....

.....

.....

.....

20. Which activities do you find most useful and important to general requirements gathering (refer to the reference numbers in the table above if necessary)?

.....

.....

.....

.....

## FURTHER COMMENTS

21. Are there any other activities or issues that have not been discussed?

.....

.....

.....

.....

## GLOSSARY

***requirements gathering/analysis:*** all the activities devoted to identification of user requirements, analysis of the requirements to drive additional requirements, documentation of the requirements as a specification, and validation of the documented requirements against the actual user needs.
[Saiedian, H., & Dale, R. (2000). Requirements Engineering: Making The Connection Between The Software Developer And Customer. Information And Software Technology, 2000(42), 419-428.]

***commercial off-the-shelf software:*** software defined by a market-driven need, commercially available, and whose fitness for use has been demonstrated by a broad spectrum of commercial users.
[IEEE. (1998). IEEE Recommended Practice For Software Acquisition (Std 1062-1998). New York: IEEE, Inc. (Original work published 1993)]

***life cycle model:*** the framework that maps a number of steps to follow (guidance) and the activities associated with those steps. These steps help the development team to meet the projects user requirements, budget and deadline constraints.
[IEEE. (1997). IEEE Standard For Developing Software Life Cycle Processes (Std 1074-1997). New York: IEEE, Inc. (Original work published 1995)]

***project:*** the project refers to the development of a GIS application as a solution for a client.