

**Determining and Developing Appropriate Methods
for Requirements Verification and Modelling of
Telecentre Operational Monitoring in a Developing
Country**

By

Jeebodh Pancham

Submitted in fulfilment of the requirements of the Master of Information
and Communications Technology Degree

In the

FACULTY OF ACCOUNTING AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

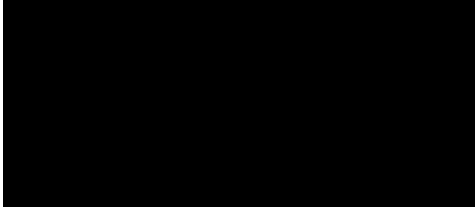
DURBAN UNIVERSITY OF TECHNOLOGY

DURBAN, SOUTH AFRICA

December 2016

Declaration

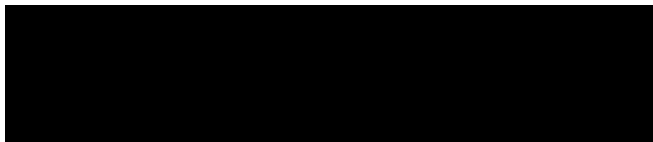
I, Jeebodh Pancham, declare that this thesis is a representation of my own work in both conception and execution.



12 December 2016

Date

Approval for examination



Supervisor

Prof R C Millham (PhD)

29 March 2017_____

Date

ABSTRACT

Telecentres are a means of allowing members of disadvantaged communities access to information and communication technologies (ICTs) so that they are included in the digital world. Thorough literature searches, along with communication with the Universal Service Access Agency of South Africa (USAASA) indicated that there was no common operational monitoring model for Telecentres. The lack of such a model resulted in a lack of real time user and usage profile information to provide strategic business insights for managers. To obtain the requirements for this model, different stakeholders of South African Telecentres were consulted, and these consultations were supplemented by research studies based on international Telecentres.

After a detailed evaluation of the different research methodologies, positivism and reductionism were selected as the most appropriate conceptual frameworks for the research. The research design included both quantitative and qualitative research methods. Requirements engineering was used to provide a number of different methods for verification and modelling. The UML methodology was used to represent the TeleMun monitoring model. A specific UML diagram, the activity diagram, was used to validate the phase consistency of the TeleMun model using the semiformal tool of VeriScene. The choice of methods depended on several factors, for example, the problem domain, and the nature of the solution required, amongst others. Design science methodology was selected as an overarching methodology to encompass the full process from requirements to the final design and reporting phases. This methodology was used both in the design of the model and in the design of VeriScene. (The literature review had revealed that there was a gap concerning appropriate phase consistency tools to ensure consistency between the requirements and design phases. To address this gap, a tool 'VeriScene' was developed to provide this consistency). In order to analyse these requirements, a combination of different appropriate methods was selected, providing the design strength associated with triangulation. These requirements engineering methods were applied to derive the TeleMun model.

Thus the monitoring model, TeleMun, was developed, verified and partially validated using several requirements engineering methods. The model is designed at a high level and therefore can be modified to suit other local and international Telecentre operations.

ACKNOWLEDGEMENTS

I would first like to thank my thesis supervisor Professor Richard C. Millham of the Department of Information Technology at Durban University of Technology for his guidance and support during this study. Professor Millham has made this journey an interesting one by making it practical with relevance to industry, and in steering me consistently in the right the direction.

I would also like to thank Prof Thiruthlal Nepal and Mr Mandla Sithole from USAASA for initiating discussions and for providing the necessary requirements on which this design is based.

My special thanks to Dr Jane P. Skinner for editing this thesis and to Ms Sara Mitha for ensuring that all references are correctly documented.

Finally, I must express my very profound gratitude to my wife (Nasha) and children (Rivajh, Shivash, Shreeya) for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of Contents

ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
List of Publications and Artefacts	ix
Abbreviations	xi
List of Figures	xii
List of Tables.....	xiv
CHAPTER 1 – Introduction.....	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Objectives.....	4
1.4 Significance of Study	4
1.5 Limitations	4
1.6 Chapter Summaries	5
1.7 Conclusion	7
CHAPTER 2 – Literature Review.....	8
2.1 Introduction	8
2.2 Digital Divide.....	8
2.3 Telecentres to Bridge the Digital Divide	8
2.3.1 Definitions of Telecentres	9
2.3.2 The Impact of Telecentres.....	9
2.3.3 South African Context of Telecentres.....	10
2.4 Need and Benefits of Monitoring of Telecentres	11
2.4.1 Research Indicating the Need for Telecentre Monitoring.....	11
2.4.2 Common Business Processes	12
2.4.3 Common Set of Attributes	12

2.5	Challenges in Data Collection.....	13
2.6	More Responsive Model Needed.....	16
2.7	Research Approaches for Development of an Electronic Model.....	18
2.8	The Development Process for a Validated Monitoring Model	24
2.9	Requirement Engineering.....	25
2.9.1	Requirements Elicitation.....	25
2.9.2	Requirements Analysis.....	30
2.9.3	Requirements Description and Modelling	31
2.9.4	UML Activity Diagrams	36
2.9.5	Requirements Evaluation and Validation	37
2.9.5.1	Validation and Verification.....	37
2.9.5.2	Existing Tools for Validation and Verification.....	38
2.10	Conclusion	40
	CHAPTER 3 – Methodology	42
3.1	Introduction	42
3.2	Research Approach	42
3.2.1	Research Philosophy.....	42
3.2.2	Research Design	43
3.2.3	Research Method	44
3.3	System Development Methodologies	45
3.4	Application of Design Science Methodology	48
3.4.1	Requirements Elicitation.....	50
3.4.2	Analysis	52
3.4.2.1	Stage 0 – Reduction of Attributes.....	52
3.4.2.2	Stage 1 - Drafting and Consolidation of Scenarios.....	52
3.4.2.2.1	Identification of Test Data for Scenarios and Consolidation of Requirements.....	54

3.4.2.4	Stage 2 - Confirmation of Scenarios.....	56
3.4.3	High Level Design	57
3.4.4	Verification	58
3.4.4.1	Application of Chosen Research Design for VeriScene Phase Consistency Tool	62
3.4.4.2	Requirements.....	63
3.4.4.3	Analysis	64
3.4.4.4	Design.....	65
3.4.4.5	Implementation	71
3.5	Summary	72
CHAPTER 4 – Results.....		73
4.1	Introduction	73
4.2	TeleMun Model.....	74
4.2.1	Requirements Elicitation.....	74
4.2.2	Analysis	75
4.2.2.1	Stage 0 – Reduction of Attributes.....	75
4.2.2.2	Stage 1 – Drafting and Consolidation of Scenarios.....	75
4.2.2.3	Identification of Test Data for Scenarios and Consolidation of Requirements.....	78
4.2.2.4	Stage 2 – Confirmation of Scenarios	80
4.2.3	Design	92
4.2.4	Final Verification and Validation.....	94
4.3	VeriScene	98
4.3.1	Requirements Elicitation.....	98
4.3.2	Analysis.....	98
4.3.3	Design	99
4.3.4	Evaluation	104
4.3.5	Communication.....	108

4.4	Final Results of TeleMun.....	108
4.5	Summary	108
	CHAPTER 5 – Recommendations and Conclusion.....	110
5.1	Introduction	110
5.2	Overview	110
5.3	Research Objectives Met.....	111
5.4	Recommendations	113
5.5	Future Work	114
5.6	Summary	114
	Appendix A – Researcher’s Attributes	116
	Appendix B - Activity Diagrams	118
	Appendix C – Testing VeriScene with Different Domains	122
	Appendix D – Initial Models.....	124
	Appendix E – Use Case Diagrams	127
	Appendix F – Database Structure	131
	Appendix G – Letter from Media Platform.....	135
	References	136

List of Publications and Artefacts

1. Paper 1

Pancham, J., Millham, R., Singh, P., ‘A Validated Model for Operational Monitoring of Telecentres’ Activities in a Developing Country’ in *Proceedings of the 7th International Development Informatics Association Conference*, IDIA International Development Informatics Association, Bangkok, Thailand, 2013.

2. Paper 2

Pancham, J. and Millham, R., 2015, June. ‘Design Phase Consistency: A Tool for Reverse Engineering of UML Activity Diagrams to Their Original Scenarios in the Specification Phase’. In *International Conference on Computational Science and Its Applications* (pp. 655-670). Springer International Publishing.

Note: Many of the phrases used to express concepts and the description of the tool from the paper were incorporated in this thesis. Hence, the possibility of phrase matches from Springer.

3. Project Funding

After a successful competitive proposal submission, the researchers received funding from the Technology Innovation Agency (TIA), a government department that sponsors innovative projects that impact on citizens’ lives. In order to be awarded funding, submitted projects must have a good scientific basis and must be implementable. As a result of this TIA funding, an electronic monitoring software application is in the process of being developed. The design model was verified against the user requirements, partially using this dissertation’s own phase consistency tool “VeriScene”, to ensure that user requirements would be fully incorporated in the design. Pilot Telecentre sites have been identified and this application, once fully developed, will be deployed at the selected sites to implement the concepts in the dissertation. Furthermore, this application will monitor and report on Telecentre user profiles and usage to Telecentre managers at the local level. The application will pass this information on to a centralised database at national headquarters. Based on the data from this centralised database, reports that are more sophisticated can be generated to provide better business

insights into Telecentre operations for higher levels of management and for sponsors.

4. Artefact

A phase consistency tool, VeriScene, was developed to traverse a derived activity diagram and produce a set of scenarios. These generated scenarios have been compared with the original scenarios, from which the activity diagram was derived, to ensure consistency and completeness between the specification and design phase. This tool was demonstrated to several local software development firms who expressed a written desire to utilise it to improve software quality and reduce errors (see Appendix G). Funding for further development of this VeriScene tool has been made via a TIA application.

Abbreviations

Abbreviation	Full Description
AGG	Attributed Graph Grammar
AOSD	Aspect-Oriented Software Development
AOSE	Aspect Oriented Software Engineering
ATM	Automated Teller Machine
BI	Business Intelligence
DFD	Data Flow Diagram
DFS	Depth First Search
ICT	Information and Communication Technologies
ITU	International Telecommunication Union
KZN	KwaZulu Natal
OCL	Object Constraint Language
OOA	Object-Oriented Analysis
OOP	Object Oriented Programming
RE	Universal Service Access Agency of South Africa
SDLC	Software Development Life Cycle
SOP	Standard Operating Procedure
TeleMun	Telecentre Monitoring Model
TIA	Technology Innovation Agency
UCEd	Use case Editor
UML	Unified Modelling Language
UNISA	University of South Africa
USAASA	Universal Service Access Agency of South Africa
VeriScene	Verification of Scenarios

List of Figures

Figure	Title	Page
2.1	Nominal Design Science Process	23
2.2	Methods applied in RE process - Adapted from (Xuping 2008)	26
2.3	Use case Diagram Key	36
3.1	Phase, Model, and Artefact	45
3.2	Design Science for TeleMun	46
3.3	Methodology to build and verify TeleMun	50
3.4	Reduction of attributes	52
3.5	Drafting and consolidation of Scenarios	53
3.6	Confirmation of scenarios	57
3.7	High level design process	58
3.8	DS Verification Process for Phase Consistency	60
3.9	Design verification and validation	61
3.10	Verification of VeriScene	66
4.1	Initial use case – Use Service	77
4.2	Services	79
4.3	Age Category	79
4.4	Equipment	80
4.5	Use case – Request Service	84
4.6	Use case – Allocate Service	85
4.7	Use case – Complete usage of service	86
4.8	Use case – Bill usage	88
4.9	Use case – Make Cash Payment	88
4.10	Use case – Bill Account Payment	89
4.11	Create user profile	90
4.12	Allocate Service	91
4.13	Graph of service usage over a 6 month period	91
4.14	User and usage Profile	95
4.15	TeleMun	97

Figure	Title	Page
4.16	Walkthrough of TeleMun with checklist	107
B1	Start of day routine	118
B2	Billing and payment	118
B3	Final use service process	119
B4	Refund fee	120
B5	End of day routine	121
C1	Trouble Ticket	122
C2	Order processing	123
D1	Call box and internet café business model	124
D2	Telecentre business model	125
D3	Monitoring model	126
E1	Telecentre monitoring and reporting system	127
E2	Acquire equipment	128
E3	Start Application	128
E4	Terminate Application	128
E5	Surf Internet	129
E6	Internet connection fault	129
E7	Internet connection restored	129
E8	PC switched on	130
E9	PC switched off	130
F1	TeleMun ERD	

List of Tables

Table	Title	Page
2.1	Four Worldviews (Creswell 2013)	19
3.1	Use case Definition	54
3.2	Scenario definition	54
3.3	Formalization, by action and action link rules, of the Telecentre activity diagram	69
4.1	Data for scenario	78
4.2	Sample Attributes	78
4.3	Use cases	82
4.4	Use case and scenario Request Service	83
4.5	Use case and scenario – Allocate a service	85
4.6	Use case and scenario – Complete usage of service	86
4.7	Use case and scenario – Bill Usage	87
4.8	Use case and scenario – Make Payment	88
4.9	Activity 2: Telecentre operation, Scenario 1: Successful usage = Yes	100
4.10	Activity 2: Telecentre operation, Scenario 2: Reuse Service = Yes	100
4.11	Activity 2: Telecentre operation, Scenario 3: Reuse Service = No	101
4.12	Activity 2: Telecentre operation, Scenario 4: Continue Wait = Yes	101
4.13	Activity 2: Telecentre operation, Scenario 5: Continue Wait = No	102
4.14	Scenario 1, Successful usage = Yes	102
4.15	Scenario 2, Reuse Service = Yes	103
4.16	Scenario 3, Reuse Service = No	103
4.17	Scenario 4, Continue Wait = Yes	104
4.18	Scenario 5, Continue Wait = No	104
A1	Researchers on Telecentres	116
A2	Attributes on Telecentres	117
F1	Databases Attributes	131

CHAPTER 1 – Introduction

1.1 Background

The International Telecommunication Union (ITU) Information and Communication Technologies (ICT) 2016 report states that only 47% of the world's population uses the internet. The reason for this low figure is the skew in internet access between developing and developed countries which stood at 41% and 84% respectively at the time of this research (Sanou 2016). These figures highlight the existence of the digital divide and indicate the lack of connectivity and of access to information by people in the developing world. The 'digital divide' refers to the difference in levels of access to ICTs, as well as the divergent quality of access, between communities in the developed and developing worlds (Mossberger, Tolbert and Hamilton 2012). One of the ways to bridge this digital divide has been found in the opening of ICT facilities in the form of 'Telecentres'. These provide public access to the internet and to other ICT facilities.

There has been growing utilization of ICTs for interventions in developing countries including initiatives at community level in social and economic development (Walsham and Sahay 2006) and in community mediation and violence prevention (Bailey and Ngwenyama 2010). Telecentres overall have been established as a means to enhance people's quality of life by providing them with greatly improved access to information, including useful commercial and employment information on, for instance, markets and vacancies, along with more basic facilities such as the ability to create, fax and email documents.

However, despite the promise that Telecentres offer to their users, the support processes for managing these centres has been identified by (McConnell *et al.* 2001; Sey and Fellows 2009; Pather and Gomez 2010) as a weakness, and in particular an automated monitoring system is needed for Telecentres' operational activities.

Currently, Telecentre usage data is collected using ad-hoc traditional, manual, practices (Gomez, Pather and Dosono 2012; Rajapakse 2012). These methods are subject to variation and inconsistency in personnel employed, information requested, and opinions obtained. One of Universal Service Access Agency of South Africa's (USAASA's) mandates involves recognizing these limitations and the need for a continuous monitoring tool (USAASA 2011). The most effective and the cheapest method of continuous monitoring is recognised to be an electronic model. However, such a system requires a verified design model before deployment (Liu and Yang 2005). This model, in turn, requires the use of a selected set of reliable, industry-accepted, practices to first develop and then verify it (Khan *et al.* 2011).

The methods, tools and processes needed to develop this model belong to the Requirements Engineering (RE) domain. Requirements engineering is a phase of the systems development process that focuses on gathering and documenting system requirements from stakeholders to serve as a basis for further system development such as draft design, verification and validation (Pohl 2010). RE can also be used to verify the derived requirements and validate the subsequent draft model in the design phase (Hull, Jackson and Dick 2010). In addition RE can encompass the viewpoints of different stakeholders (Pandey, Suman and Ramani 2010).

Triangulation of appropriate methods is sometimes needed to guarantee accurate results, as the deficiencies of any one method can be circumvented by a combination of methods, thus capitalizing on their individual strengths (Yeasmin and Rahman 2012). While investigating the verification of requirements for the Telecentre Monitoring Model (TeleMun), a new method of verification was developed because the existing methods were deemed inadequate.

1.2 Problem Statement

As indicated in the literature, in order to make sustainable business decisions for Telecentres, there is a need for continuous and systematic collection of operational

Telecentre data. An operational monitoring model for Telecentres is needed to accomplish this.

The first step in developing a common monitoring model would be to investigate the commonality of the processes and attributes required. Once these are identified, the existence of a model needs to be determined. If no appropriate model exists, then specific requirements and appropriate methods for developing, verifying, and partially validating such an operational model need to be identified. If available verification methods are not suitable, a set of scoping parameters needs to be investigated before a method of filling this gap can be developed.

In this study, feasible RE models had to be established to capture and verify requirements and to validate the TeleMun. It was recognised that if such a model did not exist, a new model to meet the stakeholders' needs would need to be developed. Such a model would use industry-accepted methods where these existed, or, where a gap in the required existing methods was identified, new methods would be developed.

Questions

The following questions were identified as pertinent to the research:

1. Do suitable electronic Telecentre operational monitoring models exist which are able to monitor user and usage profiles along with internet and power failure data, or does a model need to be developed to fill this need?
2. If a model needs to be developed, are there feasible RE methods to capture and verify the requirements for, and to partially validate, the ensuing model?
3. If there are gaps in existing feasible requirement engineering methods, can these gaps be addressed through the development of appropriate methods and tools to check that the model meets:
 - a. consistency of the requirements from the design to the specification software phase, and
 - b. Full path coverage?

1.3 Objectives

The purposes of this research are to:

1. Determine whether there are feasible models for electronic operational Telecentre monitoring that are able to gather user and usage profiles as well as monitor internet and power failures.
2. If no appropriate model exists, to determine appropriate RE methods to capture and verify the requirements for, and to partially validate, the ensuing model.
3. Investigate existing methods and, if necessary, to develop new feasible methods to verify and partially validate iteratively, the requirements for this draft model, and to ensure that the model meets the requirements of:
 - 3.1 Consistency from the design to the specification software phase,
 - 3.2 Path coverage.

1.4 Significance of Study

Research indicates that there is a need for continuous operational monitoring of Telecentres so that timely and accurate information for their activities is available for better decision making. (Harris 2007) identifies the need for robust monitoring and appraisal systems to capture operational data systematically, and to facilitate future management decisions based on this real time data.

This study develops a common verified, and partially validated, model with the flexibility needed to be able to monitor Telecentres at different locations in South Africa and in other countries. This model enables the implementation of a monitoring tool as required by USAASA in order to make better sustainability and other decisions.

Suitable RE methods were used to investigate, develop, verify and partially validate the TeleMun. The investigations identified a gap in these methods, and a tool was subsequently developed to fulfil these requirements.

1.5 Limitations

The Telecentre operational monitoring model was generated from a set of requirements acquired from Telecentre managers in KwaZulu Natal (KZN), a province in South

Africa. Although these requirements regarding monitoring are representative of a Telecentre operation in South Africa, some variation will exist in the implementation detail. As a result, the Telecentre operational monitoring model design, and its validation, remained at a high level. This adds flexibility in its implementation in the different Telecentres, however.

Because the research relied on the Telecentre managers from the KZN area for requirements elicitation, there is a possibility that some requirements were omitted.

Consequently:

1. The model might not cater for universal implementation.
2. There is a possibility that some requirements might have gone undetected.
3. Community viewpoints on Telecentres and their operations were not taken into account.
4. It is possible that a comprehensive set of reporting requirements was not obtained.
5. The requirements, in the form of natural languages, could be translated into formal languages, but these are difficult for stakeholders to review for errors. Therefore, because of the non-technical background of the stakeholders, certain RE methods were not feasible. The initial requirements could be translated into a formal notation, but this translation brings the risk that the stakeholders might not understand these formally notated requirements.

In addition, as access to stakeholders was limited, all the reporting requirements were not known and therefore could not be fully accommodated in the design. The standard processes for the collection of Telecentre data, as expressed in the literature and stakeholder elicitation, were taken into account in this model, which could however still have excluded data from some different processes.

1.6 Chapter Summaries

Chapter 1 presents the digital divide and the introduction of Telecentres as an intervention to assist in bridging this divide. The need for an intervention involving

continuous operational monitoring of Telecentres to enable better decision making was then discussed, leading to the need for a common monitoring model to be developed and verified through various RE methods. This chapter outlines the background of the Telecentre environment and provides the problem statement relating to Telecentre monitoring. It also provides the research questions, the objectives to be achieved, the importance of the study, and its limitations.

By reviewing the literature, Chapter 2 provides a critical analysis of the issues outlined in Chapter 1. It discusses the digital divide and the need for Telecentres and their monitoring. It indicates the need for monitoring to have a validated common model that in turn requires the use of existing RE methods. It highlights the need for RE methods to verify requirements, as well as methods to validate the draft model. It identifies a gap in RE methods, which validates the need for a model in this domain.

Chapter 3 discusses the methodology that was followed to obtain and verify the requirements, to develop a draft model, and to partially validate the model. It discusses the gap which was discovered in the RE methods, and the resulting necessity for verification. It explains the new feasible RE methods which were developed to address this gap.

Chapter 4 presents the results obtained from the requirements, and verification of these requirements. A draft model from these verified requirements is presented along with a prototype based on this model. This is used to verify and partially validate the model. Stakeholder feedback after demonstration of the prototype is presented. The results of the verification of the model, using RE methods, including newly developed methods, are detailed.

Chapter 5 concludes the discussion of the research project into the design of a validated model for continuous monitoring of Telecentre activities. Recommendations for further research in this area are provided.

1.7 Conclusion

This chapter provides a synopsis of the research undertaken. The concepts of the digital divide, Telecentres, the need for real time monitoring, the need for a verified and partially validated Telecentre model, and RE methods to verify and partially validate this model are discussed. Later chapters will discuss how the model was designed, verified, and partially validated. The problem statement, objectives and the importance of the study are given in order to indicate the relevance of this research. This study has limitations due to the broad nature of RE and its implementation – and these are explained. The structure of the study provides a description of the chapters that includes a literature review, the methodology employed, analysis of the results of the research, the conclusions drawn, and recommendations for future research.

CHAPTER 2 – Literature Review

2.1 Introduction

This chapter begins with the definitions and concepts relevant to the digital divide, which is defined as a gap between those with access to ICTs, with their global means to connect to the world, and those who lack this access. The role of Telecentres as a community-centred solution, or bridge, to help address this digital divide is presented. Telecentres have been researched widely using traditional methods, such as surveys, which have disadvantages in terms of speed and accuracy. In order to make sound business decisions and to ensure the sustainability of Telecentres, the need for timely and accurate monitoring information is indicated in the literature. It is shown that this can be provided through a common TeleMun. In addition, to develop this model, several different paradigms, methodologies, and methods are presented and evaluated as to their suitability for developing the model.

2.2 Digital Divide

The various definitions of the digital divide focus on different issues such as discrepancies in usage and skills (Min 2010); (in)equality in the use of ICT's (Gomez 2012); poor broadband access in rural areas (Townsend *et al.* 2013) and level of education and economic status, along with the significance of rural as opposed to urban contexts (Mossberger, Tolbert and Hamilton 2012). This research subscribes to the broad definition of this gap as one existing between those who have access to IT and those who do not and in particular, it localises the gap to varying opportunities for people to use computers and the internet (Van Dijk, 2005).

2.3 Telecentres to Bridge the Digital Divide

As explained above, one of the most promising ways to help bridge the digital divide is understood to lie in the implementation of Telecentres which are physical spaces that provide and enable public usage of ICT facilities and internet access involving online communication tools and providing for online learning (Razak, Hassan and Din 2010).

2.3.1 Definitions of Telecentres

Although there are different definitions of Telecentres (Pather and Gomez 2010; Gomez and Baron-Porras 2011; Gomez, Pather and Dosono 2012; Seman *et al.* 2013) this research uses the most common definition of Telecentres by the (Telecommunication Regulatory Authority. Sultanate of Oman 2012). This definition defines a Telecentre by its purpose of allowing people to access IT services in a public place and in terms of the telecommunication and IT services offered (Telecommunication Regulatory Authority. Sultanate of Oman 2012). There are a number of common IT services offered, most commonly internet access and word processing facilities (Colle 2005; Jacobs and Herselman 2006).

2.3.2 The Impact of Telecentres

Although challenges including sustainability, financing, monitoring and reporting of activities, have been identified by many researchers including (Hunt 2001) and (Benjamin 2009) there are also notable positive impacts of Telecentres worldwide. These include increased ICT literacy levels achieved by centres which offer extensive training to their staff and users in Malaysia (Rajapakse 2012); assistance with homework and transcription (Bayo, Barba and Gomez 2012); access to training, internet and office applications (Razak, Hassan and Din 2010); business growth resulting from online collaboration between customers and vendors in Nigeria (Achimugu *et al.* 2009) and extending e-governance services in India (Naik 2011). Furthermore, (Baron and Gomez 2012) identify some of the positive social impacts of usage of public access computing including clients having access to more sources of information, and to a greater volume of information, than would otherwise be possible, stronger interpersonal relationships resulting from a feeling of being included in a global community, and being given opportunities for improved learning.

Telecentres therefore serve as important facilities that provide access to technologies in areas that have limited or no access to ICT's. These examples highlight the many diverse benefits of access to ICT for the population, especially within developing countries.

2.3.3 South African Context of Telecentres

An initial study of Telecentres in South Africa by (Benjamin 2001) identified several challenges frequently faced by South African Telecentres including minimal financial support, equipment failure and poor technical support, all of which can threaten their sustainability. In addition there is a lack of any monitoring model to capture user and usage profiles or to record equipment and internet connection failure (Benjamin 2001). South Africa therefore does not have a sustainable model for Telecentres which can be rolled out on a large scale (Benjamin 2009). Coupled with this, there is no electronic monitoring model that can be used to monitor these Telecentres and provide the information required to analyse user and usage profiles in real time. Consequently, the impact that Telecentres have had so far on rural communities in South Africa is largely unknown. It has yet to be investigated and interpreted, as noted by Benjamin as long ago as 2001 (Benjamin, 2001).

The strategic plan of USAASA has listed the need for monitoring and reporting on these activities and for the impact of the Telecentres in South Africa to be documented (USAASA 2011). Monitoring will be the first step in providing the data necessary in determining access numbers at Telecentres. As a partial solution to this monitoring issue, the development of a monitoring model, along with the data it gathers, could be a valuable resource for future research in measuring the impact on rural communities. One of the strategic objectives of USAASA is to monitor and evaluate the degree to which universal access has been accomplished in South Africa. In particular, one of the key performance goals is to develop a monitoring and evaluation tool (USAASA 2011).

According to (Gomez 2012) attempts by the South African government to improve access to ICTs have been slow and this can be attributed partly to the problems experienced by Telecentres which often do not function adequately due to equipment failure and / or internet downtime. He notes also that new strategies focusing on current policy have led to outsourcing Telecentre operations to private organisations (Sitole 2014). Now USAASA will need hard real-time monitoring data for these outsourced

organizations to make informed decisions on the functioning of their Telecentres, and on the best use of their resources as well as their investment viability.

2.4 Need and Benefits of Monitoring of Telecentres

In this section the need for a common model to produce timely, consistent and accurate Telecentre user and usage data are highlighted. A common set of business process and data attributes will enable the development of a typical monitoring model.

2.4.1 Research Indicating the Need for Telecentre Monitoring

According to (Rahmat *et al.* 2013) quality decision making on issues relating to Telecentres is dependent on a Business Intelligent (BI) approach. BI can be referred to as a process of turning data into information and then into knowledge that can be used for good decision making (Ahmad and Shiratuddin 2010). This accurate, real time information needs to be gathered automatically rather than manually from the source if it is to provide timely BI at all levels. Such information will inform important decisions on sustainability of Telecentres, allocation of assets, and most needed locations. Once this type of real time data is available it can also be used to market the Telecentres to potential advertisers (Sitole 2014).

Monitoring will provide the raw data that can be turned into the information needed to provide insights into the current and future needs of Telecentres. BI, which is able to discover patterns from real time accurate data, is however beyond the scope of this research. Current ad-hoc queries based on data collected via questionnaires and interviews have a reliability challenge as this data relies on human participant input, which could entail bias. Ideally, more timely and fuller sets of data, employing BI, could be used in appealing to potential private sponsors and advertisers, as discussed by (Sitole 2014).

As indicated above, data from Telecentres is generally gathered on an ad-hoc basis and this process lacks uniformity in terms of times and data collected. In addition, because

there is no systematic gathering or methodological process, it becomes difficult time consuming and costly to report on user and usage profiles. A literature search could identify no formal standard operating procedure (SOP) or work flow, which had been documented to address this monitoring gap. This lack of a standardised process and indicators is highlighted by (Gomez, Pather and Dosono 2012) in their discussion of Telecentres in South Africa where they found it hard to discover vital information from Telecentre operators sufficient to allow USAASA to monitor their progresses efficiently. Moreover, USAASA has never developed standards for key terms that would permit it to plan effectively (Parkinson 2005).

In some cases, irregular monitoring was done whilst in other cases monitoring was omitted from the management plan altogether (Harris 2007). The issue of irregular monitoring is further emphasised by (Gomez, Pather and Dosono 2012) where they conclude that there is little proof of any comprehensive understanding of the impact of Telecentres, but only snippets of evidence from a few selected Telecentres as recorded in specific studies. These note that some resources became unusable within the first year, or that resources were barely used, and that there were long queues at Belhar and Bonteuewel Telecentres in Cape Town, South Africa. The consistent monitoring of activities, on the other hand, would enable an understanding of equipment usage and such problem cases could be identified at an early stage.

2.4.2 Common Business Processes

Business processes are defined as a combination of associated activities within an enterprise that delivers a service or product (Rodríguez *et al.* 2011). The work by (Veeraraghavan *et al.* 2006) on the commonality of Telecentre business processes demonstrates the potential for a universal monitoring model.

2.4.3 Common Set of Attributes

As outlined in section 2.4.2 these Telecentre business processes generate data on which decisions are based. As highlighted by (Gomez, Pather and Dosono 2012) a common

design model would provide real time Telecentre data through uniform processes to enable sound business intelligence on Telecentre operations to be gathered. This research indicates that a common design model that gathers data through common processes would be a solution able to provide for the needs of real time data collection.

After reviewing the types of data collected by Telecentre researchers, a common set of data attributes emerged, including gender, service usage, age, occupation, and number of visitors (Alasow, Udomsade and Niyamangkoon 2010; Cheang and Lee 2010; Lashgarara, Karimi and Mirdamadi 2012; Rajapakse 2012). Other researchers collected other attributes including qualification and distance of user to Telecentre (Cheuk, Atang and Lo 2012; Gomez 2012). However, because data on these attributes were collected manually, the process faced many challenges such as inconsistency (Burnard 1991), intensive resource usage, and high cost (Gomez, Pather and Dosono 2012).

Initially there was limited functionality and an emphasis on automating existing processes (data-driven) of existing data. In the late nineties and early 2000s the focus shifted to Business Process Reengineering (BPR) that modified existing processes for better efficiency (Stoilov and Stoilova 2006). This business reengineering signified the move from manual to automation, which included the transformation of data, and the automation of business processes to integrated electronic systems. It would be in keeping with BPR, for Telecentres' business processes to be standardised and incorporated into a common model (Xiaodong 2007).

2.5 Challenges in Data Collection

At present conventional manual data collection methods, as outlined below, are used to gather data from Telecentre users, operators and managers worldwide, for example (Gomez 2012) used semi-structured interviews, questionnaires with open and closed questions, interviews with experts and operators, visits to sites, surveys of users, literature reviews, and focus group interviews to collect data from 250 000 centres across 25 countries. In another study (Rajapakse 2012) face-to-face, semi-structured, and in-depth interviews were conducted in seventeen Telecentres across Sri Lanka. These data

collection methods clearly pose serious challenges especially as regards the enormous amount of time and cost associated with them.

Again, (Bailey 2009) used interviews together with thematic content analysis to study concerns which appeared to impact the social viability of Telecentres in developing contexts. His study categorised and discussed the different issues that affect sustainability. In another study, in-depth, semi-structured interviews were used by (Cheuk, Atang and Lo 2012) to study community perspectives regarding the Telecentre in Bario Borneo, Malaysia. This interview process included two interviewers per interviewee, one to focus on the questions whilst the other documented all responses. This indicates the costly and time-consuming aspects of using multiple resources to achieve a common aim. Furthermore, distilling common themes from qualitative answers is difficult, and non-uniform, due to the richness these responses (Burnard 1991).

In his work Gomez (2012) examines the user profiles and services rendered in twenty-five developing countries, in particular one study completed and analysed 799 user surveys in South Africa. This again indicates the large amount of work needed to collect, analyse and manage the data. Furthermore, these manually based research methods require staff training on data collection methods to assist the researchers (Gomez et. al., 1999).

As indicated by Gomez, each survey requires enormous resources. Another such research project conducted by (Pather and Gomez 2010) studied the success levels of public access to ICT programs in 25 countries. This international research also highlights the extensive resources required by traditional methods of data collection in the absence of an effective electronic monitoring model. A further problem with these surveys is that they only provide snapshots of the status quo at the time of the survey (Gable 1994). The time and cost of assimilating such survey results to produce conclusions also tends to negate the purpose of determining the success of Telecentres as the conclusions are inevitably based on outdated data. Consequently, a more responsive model that collects data on a real-time basis and produces timeous and accurate data is strongly indicated.

(Sopazi and Andrew 2008) in their qualitative study of Telecentres, used face to face interviews for a full day and thereafter observations and interviews over a period of four months in their South African case studies. They also noted the high cost of interviews and of non-participatory observations, and possible biases in the answers given. One important finding of their research was that internet access was only available two years after the opening of a Telecentre. This fact was only discovered after the interviews were completed. This finding illustrates a high delay factor in this method of collecting data and reaching conclusions. Findings from this study, along with the burden of work, and time spent, when using manually based methods, all go to prove the high time and cost factors involved in traditional data collection practices.

Another issue noted by (Hudson 2001) is that the sources of information on Telecentres contain stories and anecdotes that provide useful insights and lessons learned on usage, user profile and services offered. Anecdotes may provide insightful information, but are difficult to collect. These anecdotes could also be open to subjective interpretations as well as being multifaceted which makes synthesis difficult (Moore and Stilgoe 2009). In addition, brainstorming and focus groups need a well-trained moderator, as well as exhibiting the limitations experienced when participants shy away from contributing for various reasons. Interviews and questionnaires may include bias from respondents as well as researchers. Questions can be ambiguous and questionnaires and interviews are time consuming and costly. As a result of the unsatisfactory nature of some of this data poor, and sometimes incorrect, decisions can be made.

These are only some of many such manual surveys carried out by many researchers over the decades to gather information on users and usage of Telecentres. This qualitative data can be used to alert management and other stakeholders to problems, but they will need reliable quantitative data in order to produce accurate and usable information in the future. Therefore it can be concluded that traditional methods of data collection have major disadvantages (Lethbridge, Sim and Singer 2005) including cost, effort, record keeping, effort required to analyse the data, and resulting problems with the accuracy and reliability of reports.

If a more responsive model were used the results would have been obtained more quickly, more accurately and more timeously. An electronic monitoring model is such a responsive model, where the researcher is not on site, and knowledge of internet usage would be immediately available for management to respond quickly to problems.

2.6 More Responsive Model Needed

Support processes for management of Telecentres is identified by (McConnell *et al.* 2001; Sey and Fellows 2009; Pather and Gomez 2010) as one of the areas that is lacking in the Telecentre operational process. (Harris 2007) concluded that even programs that were meant to prove the potential of ICTs through pilot projects have often lacked strong monitoring and appraisal systems, negatively impacting on effective implementation. Gomez and Reilly's (2002) research on the needs and expectations of Telecentre operators in Latin America and the Caribbean revealed that monitoring did not begin immediately after implementation, but rather at the end of a long-term project, resulting in incomplete data.

(Veeraraghavan *et al.* 2006) recognized traditional means of data collection as extremely time consuming and resource intensive, requiring the continuous active participation of the researcher. In reporting on their research they emphasized the need to use a software based logging tool to understand the software usage of kiosks. Such a logging tool was subsequently implemented to gather precise usage statistics for Kiosks in Maharashtra and Uttar Pradesh, India. This tool collected information on sites visited, hardware and software configurations and applications used. Jacobs and Hersleman (2006) proposed that such systems could support community centre staff and enable management to provide and improve service to users.

Razak's (2009) paper discusses the aspects that contribute to the sustainability of Telecentres in Malaysia and highlights the importance of improved methodologies for continuous monitoring and evaluation of Telecentres.

Evidence of non-usage and of demand for services only becomes known when researchers happen to conduct research on such facilities – and the findings of this research are often delayed. This lack of real time information makes decision making on sustainability very difficult, while at the same time it is becoming ever more important due to the changing needs of users and the introduction of the latest technologies.

The urgent need for an efficient process that will perform continuous monitoring. Dependent upon an electronic, validated model that will take into account the common processes and data identified above, is therefore evident. Therefore this lack of common business processes, despite the use of common attributes for analysis, created an impetus to formulate a model which would incorporate these attributes and processes, that was then verified and partially validated by (Pancham, Millham and Singh 2013) for use in Telecentres.

(Veeraraghavan *et al.* 2006) tested an electronic software tool which collected information on sites visited, hardware and software configurations, and the applications used. By querying the data gathered, information on number of users, applications used, time spent, and sites visited and internet searches could be obtained.

Results from this research show that for certain types of questions the tool appeared to gather more reliable data than did surveys. For example, the survey reported an increase in customer traffic on particular days whereas the tool did not show this consistent bias; 53% of the kiosks surveyed reported higher customer traffic than did the tool; in 69% of the cases the survey reports overstate usage and in the remaining 31% of cases the survey understates traffic by between half an hour and four hours per day. Thus, there is an inconsistency in application usage between the survey and the tool results. Electronic monitoring, through software, appears here to produce more accurate results.

However, there were weaknesses in implementation, and collection of data from the system collected by the tool was combined using a memory stick. This was not only

labor intensive, but also it did not operate in real time and was error prone. Another drawback of this tool was that the user login was based on the continuous idle time exceeding a certain value, based on a heuristic. Furthermore, this design did not collect data on user profiles and therefore cannot be used to compare usage with user profiles. However the similarities of the model used by (Veeraraghavan *et al.* 2006) and the design of the model by Pancham (2015) in which the attributes and processes are the same, indicates that there is a common need and applicability of the model.

The research by (Veeraraghavan *et al.* 2006) highlights the issues identified in the earlier collection of data by researchers and the attempts to design electronic data collection tools in this regard. Most of the attributes used are common to those identified by authors in section 2.3.9. The verified model (Pancham, Millham and Singh 2013) is an attempt to present a universal model using the common attributes used in their electronic tool as well as the common attributes identified by the authors in section 2.3.9. The implementation of such a model will result in the continuous collection of data that is independent of the researcher.

2.7 Research Approaches for Development of an Electronic Model

The research approach to the development of an electronic operational monitoring model must include a combination of the philosophical worldview of the researcher, research designs and research methods. (Creswell 2013) sees worldview as a general philosophical perspective regarding the world and the essence of the researchers' contribution to the study, whilst other authors call these perspectives 'paradigms' or differences in epistemology or ontology. Creswell goes on to explain that these worldviews can be categorised as falling within postpositivism, constructivism, transformative approaches, or pragmatism as indicated in Table 2.1.

Table 2.1 Four Worldviews (Creswell 2013)

Postpositivism	Constructivism
<ul style="list-style-type: none"> • Determination • Reductionism 	<ul style="list-style-type: none"> • Understanding • Multiple participant meanings

<ul style="list-style-type: none"> • Empirical observation and measurement • Theory verification 	<ul style="list-style-type: none"> • Social and historical construction • Theory generation
Transformative	Pragmatism
<ul style="list-style-type: none"> • Political • Power and justice oriented • Collaborative • Change-oriented 	<ul style="list-style-type: none"> • Pluralistic combination of two or more world views to address a real world problem
Positivism	
<ul style="list-style-type: none"> • Experimental • Reliance on existing methods and theories • Reliance on establishing quality through established methods 	

(Guba and Lincoln 1994) suggest three questions that need to be addressed in defining a paradigm:

- 1) What is the nature of reality that is addressed (ontology)?
- 2) What is the nature of knowledge (epistemology)?
- 3) What is the best approach to obtaining the desired knowledge and understanding (methodology)?

(Creswell 2013) definition of positivism includes the need to identify and assess the causes, which are identified through experimentation, that influence outcomes. Positivism is based on objectivity that uses established methods to explore phenomena, to generalise reality in order to build a model, and establish the quality of the derived product (Lincoln, Lynham and Guba 2011). Positivism relies on using established methods or developing new methods using existing methods (Creswell, 2013). This definition of positivism also includes a reductionist approach in that the goal is to reduce concepts into a distinct set of ideas in order to evaluate the variables that constitute hypotheses and research questions. Postpositivism is similar to positivism but it assumes an underlying theory which can be supplanted by a new derived theory. If no theory exists to be verified, this worldview is not applicable. A transformative worldview involves research related to political policy and change (Mertens 2015). If research is not directly related to political change, this worldview is not applicable. Another worldview, constructivism is dependent on the multiple varying viewpoints of

the community of participants. Research that is not reliant on participants' viewpoints, but which relies on purely objective data, is not suitable for the constructivism worldview.

The methodology may be defined as a system of methods within the paradigm and the method refers to the systematic rules or tools used to collect and analyse data for research (Mackenzie, 2006). It is important to note and differentiate research method as procedures, schemes and algorithms used in research whilst research methodology as a systematic way to solve a problem (Rajapakse 2012).

Systems development as a research methodology is a multi-faceted approach that incorporates further exploration of a problem, development of better concrete solutions, and the evaluation of solutions through empirical means (Nunamaker Jr and Chen 1990). From a research methodology point of view, the steps of systems development processes can be considered as follows: build a system, develop theories and principles from observing behaviour, and incorporate expertise in software tools for increased availability. These tools in turn will be utilised to assist in the development of new systems (Kim 2013). This process is similar to the design science methodology where the entry point is a design and development centred approach.

Within the positivist paradigm, a specific software development methodology is chosen that allows the researcher to incorporate their research and further develop it into a software model. Some of the traditional software development methodologies include the Software Development Life Cycle (SDLC), waterfall model and rapid prototyping. In addition, incremental (a base for non-traditional agile software development) methodology is discussed due to its increasing prominence.

The most widely used methodology used to develop information systems is the SDLC, which comprises a sequence of well-defined linear tasks such, are requirements elicitation, analysis, and detailed design. The limitations of the SDLC methodology – for

instance, the traditional approach using a Data Flow Diagram (DFD) – prompted several new approaches that include data oriented, prototyping, object oriented, and strategic methodologies (Avison and Fitzgerald 2003).

According to (Royce 1970) the waterfall methodology is characterised by distinct phases of requirements elicitation, analysis, program design, coding, testing and operations. During each of these phases, detailed documentation of each phase is required by the team for communication, operations and maintenance purposes. Each successive phase requires completion of its predecessor phase, with documentation. This methodology is often used and justified on large business critical systems. However, this methodology has required documentation that is time consuming to compile and maintain as the system requirements change. Other drawbacks of this methodology include new requirements arising after certain phases have been concluded, with no chance of their incorporation into the project, no flexibility in partitioning the project into stages, and difficulty in estimating time and budget for each stage. This methodology is only well suited to cases where the requirements are well understood, non-ambiguous, clear and final, and where product definition is final (Stoica, Mircea and Ghilic-Micu 2013).

A rapid prototyping methodology produces a dynamic model that is functionally equivalent to a selected part of the product (Schach 2008). This prototype is produced quickly in order to obtain feedback and clarify requirements from the client and users. However this prototype must not be expanded to the final product as it often lacks full functionality, among other things (Azeem and Gondal 2011).

Agile methodologies are based on adaptive software development methods, tailored to client requirements, while traditional methodologies are based on a predictive approach to these requirements (Stoica, Mircea and Ghilic-Micu 2013). These agile methodologies possess a number of advantages such as adaptation to dynamic changes, close interaction with the client, and minimal documentation. This methodology is suited for

small and medium size projects. However, its non-requirement to produce detailed documentation makes it difficult for larger teams and for future software maintenance (Stoica, Mircea and Ghilic-Micu 2013).

The design science methodology of software development itself provides a model for the design of a model. Empirical research can be used in two ways: firstly, the validation of a designed artefact before it has been implemented, and, secondly, assessment of the performance of a design that has been implemented (Wieringa 2010) and (Hevner *et al.* 2004). Although both validation and verification can be housed in the Design Science process model evaluation phase, it can be used for verification and partial validation for software design before implementation. Design Science according to (Peppers *et al.* 2006) is a notable process model for performing research. The process includes six steps: problem definition and motivation, objectives for a solution, design and development, evaluation and communication (as shown in Figure 2.1). This process aims to develop and provide instructions for action that allow the design and operation of IS and innovative concepts within IS which result in artefacts, models, methods, and instantiations (Österle, H., 2011) and (Hevner *et al.* 2004). Although design science is used in different disciplines, one common domain is for development of software artefacts such as monitoring systems (Baskerville *et al.* 2011).

Within design science methodology, the object oriented design approach is adopted enabling analysts to decompose a complex system into smaller, more workable modules, develop modules independently, and integrate the modules to constitute an information system (Dennis, Wixom and Tegarden 2015).

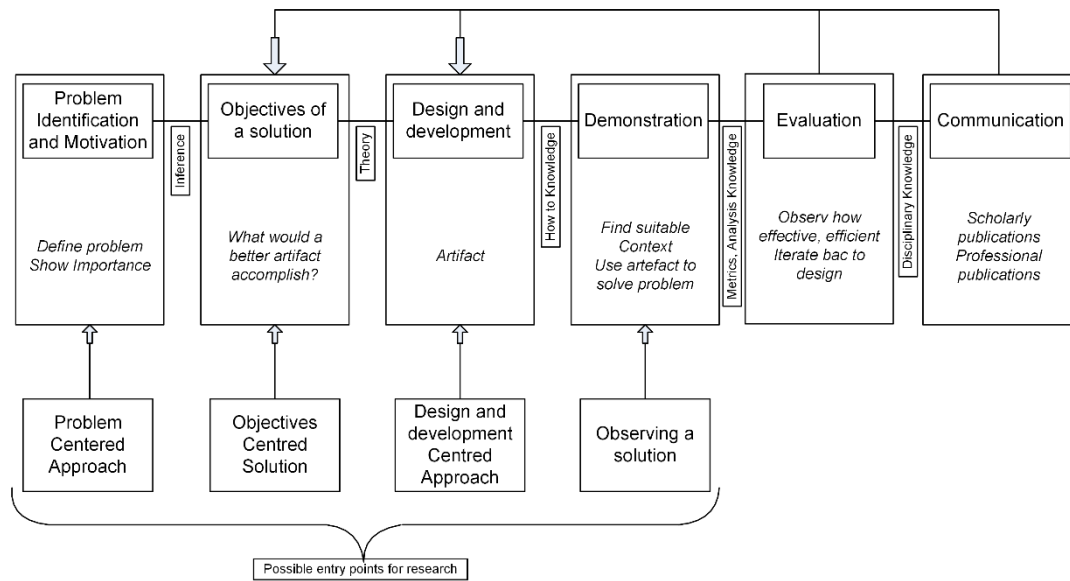


Figure 2.1: Nominal Design Science Process

The three main types of research design involve qualitative, quantitative or mixed methods. Quantitative research design (collection, analysis and generalization of numerical data) differs from qualitative approaches (understanding human experience and meaning within a given context) (Petty, Thomson and Stew 2012). Within the chosen methodology, different research methods may be chosen. The different types of research methods can fall into two basic approaches to research viz. qualitative and quantitative (Kothari, 2011). The aim of qualitative research methods is to understand problems by investigating the viewpoint and behaviour of the people in these settings and the context within which they operate, resulting in a rich understanding of the meaning and context of the phenomena studied in real life (Kaplan and Maxwell, 2005). These qualitative methods are often based on opinions, case studies or complex phenomena and are responsive to local situations, specific conditions and stakeholder’s needs. They have the disadvantage of difficulty in generalising and of making predictions and are also very time consuming. However, qualitative research methods can be used to gain rich insights on narrow topics. Quantitative research methods rely primarily on the collection of quantitative data, determining variables and constants, and studying the relationships between them (Johnson and Christensen, 2004). Using quantitative research methods one is able to generalize and predict based on the precise data gathered. This type of research is useful for large samples and will mitigate

researchers' bias. However, it is difficult to explore in-depth issues with quantitative research.

Within the positivist paradigm and its appropriate subset methodology, information systems researchers traditionally have employed a number of different research methods that can, at one level, be broadly categorized again into two: quantitative and qualitative. Further, triangulation of different methods ensures that the requirements are validated accurately as the deficiencies of any one method can be mitigated by combining methods and thus exploiting their individual advantages (Holtzhausen 2001; Yeasmin and Rahman 2012), thus increasing validity (Hussein 2009). Interviews are a common method of data collection in qualitative research. Interviews may be conducted in structured, semi-structured or unstructured form (Robson 2011).

2.8 The Development Process for a Validated Monitoring Model

The lack of a model highlighted in the above research discussion, strongly indicates the need to design an electronic monitoring model for common Telecentre processes. The model will need to be developed using an appropriate methodology. This model will also need to be verified to ensure that it meets the users' needs as well as being validated to ensure that it is built correctly. Using an appropriate paradigm, methodology, and approach, the first step in developing a monitoring model is to gather the requirements of the stakeholders who will require such information. The RE process for designing a system includes the following five main activities - requirements extraction, analysis, documentation, requirements validation and management (Attarha and Modiri 2011).

The recognised and "standardised" RE methods of these activities are depicted in Figure 2.2 which is adapted from (Xuping 2008). The most commonly used and appropriate method / methods were evaluated and selected for this research.

2.9 Requirement Engineering

2.9.1 Requirements Elicitation

A number of methods, involving both methods and tools, are used to gather requirements. RE methods include interviews, scenarios and prototyping (Attarha and Modiri 2011). Examples of these and other methods are discussed below. Requirements gathering methods considered for use include traditional methods (interviews, focus groups), prototyping, model driven technologies (use cases, scenarios), and other methods (goal oriented software engineering, aspect oriented software engineering, extreme user centred design, and quality models and goals). These recognised and “standardised” RE methods and tools are depicted in Figure 2.2 which is adapted from (Xuping 2008). This outline was chosen to include a broad range of commonly used methods from traditional to modern.

Requirements elicitation will normally begin with the functional aspects indicating client requirements. The procedure for determining the client’s requirements is called requirements elicitation (or requirements capture) (Schach 2008). Once the original set of requirements has been formulated, the process of clarifying and expanding them is termed ‘requirements analysis’.

Most of the methods listed in Xuping’s diagram were investigated for suitability for this research. Although the diagram outlined a number of methods categorised in different software development phases (Wilson, Rosenberg and Hyatt 1997), some of these methods, such as Automated Requirements Measurement, were not included in this research as investigation indicated that these methods were too complex, or awkward to use, and required well-defined requirements as their starting point

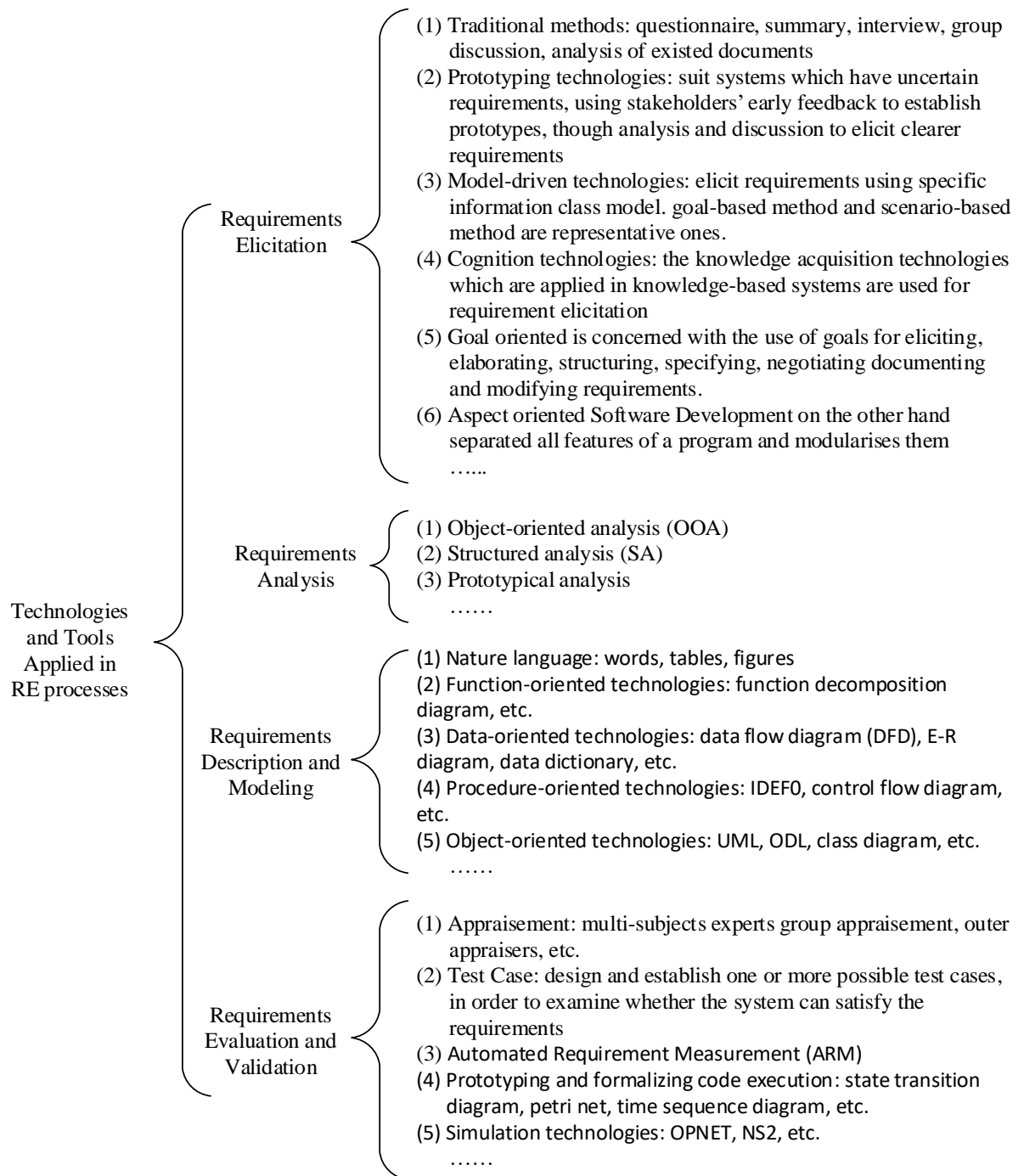


Figure 2.2 Methods applied in RE process - Adapted from (Xuping 2008)

Several RE methods have been proposed by researchers in order to reduce requirements ambiguity and to improve requirements clarity (Bee Bee, Bernardo and Verner 2010). A common method proposed by (Bee Bee, Bernardo and Verner 2010) is a

combination of face-to-face RE methods such as stakeholder and focus groups interviews. (Face-to-face interviews are possible if the user community is accessible and has the time to dedicate to such requirement gathering sessions. These respondents also need to have expertise in the system to be designed, among other things). This coincides with (Xuping 2008) traditional method of requirements elicitation as indicated in Figure 2.2.

However, if there are a large number of users, and face-to-face individual interviews are not possible, focus groups are widely used as a qualitative method to ensure that the requirements are coordinated and the process is efficient. This method encourages group interaction and will generally stimulate richer responses than individual interviews with reduced cost and time (Debus 1988; Kontio, Lehtola and Bragge 2004). However, researchers find that focus groups work well only if led by an experienced facilitator, otherwise there is a risk that they will be dominated by only a few members.

Focus groups are group interviews that are based on participants' communication (Kontio, Lehtola and Bragge 2004). Consequently, the researcher must acknowledge both the direct answers to interview questions and the communication between participants. Members of the team will be encouraged to deliberate each other's contributions. This deliberation is useful when exploring participant knowledge and experiences. It is used to research not only what participants think, but also their way of thinking and its context (Martakis and Daneva 2013). This method aids in understanding the different business contexts and processes and thereby helps to develop a good set of requirements. Focus groups are excellent data collection tools to use when one is new to a domain and seeking ideas for further exploration as well as identifying what is important (Lethbridge, Sim and Singer 2005).

Another data collection method is the analysis of documents (Xuping, 2008). This method is helpful for understanding business processes but there is a risk that documents may be outdated (Lethbridge, Singer and Forward 2003) and they are often

incomplete. Consequently, document analysis will normally need to be complemented by other methods of requirements elicitation.

If documents are poor or unavailable, another method is to use prototyping to help to ascertain which requirements users want, rather than helping reduce the complexity of business and technology requirements (Bee Bee, Bernardo and Verner 2010). Prototyping allows the developer to demonstrate (or walkthrough) a system or part thereof quickly, allowing early visibility of the prototype and giving the user an idea of what the final system will look like and how it functions (Azeem and Gondal 2011). Further it gives the client and end-user intense exposure and hands-on involvement early in the life cycle so that problems can be identified and addressed (Moscove 2011). Prototyping is a good communication tool that allows the developer to explore ideas between the analyst and the client as well as to exchange feedback (Eker 2014). This communication is an important step in preparing to develop better software that is fit for purpose, does what it needs to do, and does it well (Arnowitz, Arent and Berger 2010). Although the prototype is not used by the client directly, it prepares the analyst in providing the best possible solution. A prototype is used to obtain tacit information from the user after the user has been shown the prototype and after some interaction with him/her (Rantapuska and Millham 2010). In addition, concepts of complex systems can be demonstrated using a working model where requirements are further elicited and clarified.

Although prototyping promises positive results there are disadvantages such as the time and cost involved in delivering a prototype, too many iterations of a prototype resulting in dissatisfaction and impatience amongst users, and reducing time for documentation and testing, which could result in maintenance problems (Moscove 2011). Prototypes are usually constructed quickly mainly to obtain feedback, demonstrate particular features, or prove a concept. They are generally discarded after use thereby costing the developer time and effort. The prototype in fact, should not be extended to a live system because it lacks detailed design within its composition and often does not have important related aspects such as security. Although a prototype is indicative of the potential features of the live system it is not recommended in a

production environment, as it is not constructed from a detailed design. If this is rolled out to a live environment it could lead to implementation of incomplete systems and project management difficulties such as unreasonable expectations of completion times (Azeem and Gondal 2011).

Aspect Oriented Software Engineering (AOSE) is aimed to address previously overlooked issues of modularizing crosscutting concerns (Mohamed, Hegazy and Dawood 2010) by addressing the relationships between both functional and non-functional requirements and aspects of a system. However, within this method, there is a lack of agreed processes in separating concerns and there is no universal model that can be followed to translate requirements to aspects. Aspect-Oriented Software Development (AOSD) which follows AOSE takes imperfectly-defined aspects and encapsulates them into objects, structurally similar to an object-oriented system (Filman and Friedman 2000). One problem with AOSE and AOSD is its focus on non-functional requirements which makes elicitation of functional requirements from stakeholders secondary and/or unavailable in this model. Furthermore, not all scenarios are suitable for aspect extraction and AOSD expects developers to be able to analyse and extract concerns, which requires training and practice, with no universal agreement on the method (Besa 2011). While AOSE may be useful in modularisation of non-functional requirements, its drawbacks made it unsuitable for this research.

Another RE method that is used to elicit requirements from stakeholders in the modelling and the development of a system is Goal Oriented Software Engineering (GORE). According to (Van Lamsweerde 2001) GORE is concerned with the use of goals for eliciting, elaborating, structuring, specifying, negotiating documenting and modifying requirements. Goals are prescriptive statements of purpose whose fulfillment requires the collaboration of agents (or active components) in the software and its environment (Van Lamsweerde 2004). This mode of RE is useful where the stakeholders pay attention to the achievement of goals that are clear and known such as servicing more passengers for a transportation system or retaining cards after three wrong passwords for an Automated Teller Machine (ATM) system. However, there is no standard method of defining goals across systems. In practice, customers for a

system often find it difficult to translate their goals into measurable requirements (Sommerville 2011).

Most business analysts and software developers interviewed by Bee Bee et al (Bee Bee, Bernardo and Verner 2010) preferred to use interviews alone for RE, and only chose multiple methods if the single method did not result in clear requirements. However, choosing multiple methods ensures that requirements are clear and reinforces the system requirements. This also minimises the shortcomings of one method whilst maximising the benefits of other methods used (Yeasmin and Rahman 2012).

2.9.2 Requirements Analysis

Structured analysis methods evolved in the early 1980s to help clarify requirements for a computer system before developers designed the programs. This method helps the developer define what the system needs to do (the processing requirements), what data the system needs to store and use (data requirements), what inputs and outputs are needed, and how the functions work together to accomplish tasks. The traditional key graphical model of the system requirements that are used with structured analysis is called the DFD that depicts the flow of data between processes. According to (Braude and Bernstein 2011) nodes shown as circles or rectangles represent processing units, arrows between nodes denote flow of data, and data stores are denoted by a pair of horizontal lines. Data and processes are considered separately in a traditional approach.

The combination of data and processes led to the new object oriented approach adopted by (Booch, Rumbaugh and Jacobson 1999). In an object-oriented approach, information systems are viewed as collections of interacting objects consisting of encapsulated data and operations that manipulate them. During Object-Oriented Analysis (OOA) the objects and use cases are analysed and documented (Satzinger, Jackson and Burd 2011). Most current software development uses object oriented languages as well as OOA with objects and with use cases at an early stage of software development where use cases depict user tasks (Dennis, Wixom and Tegarden 2015).

The implementation of OOA and Object Oriented Programming (OOP) can be extended to different domains.

2.9.3 Requirements Description and Modelling

There are many challenges in gathering and representing requirements. One of these is highlighted in the (Raatikainen *et al.* 2011: 257) case study which indicates that use of natural language in requirements definition entails that “natural language sentences provide a relatively one dimensional and fragmented view of requirements”. A description of requirements using natural language is context dependent and therefore natural language cannot clearly define the processes. Various problems can arise when natural language is used to write user and system requirements: difficulty in using specifying requirements in a precise and unambiguous way resulting in a lack of clarity, and difficulties in distinguishing functional and non-functional requirements, and amalgamation of requirements (Sommerville 2011). Most of the system requirements are written in a natural language. However, is not easy for the system development team to understand unambiguously a document which is written in a natural language, without domain specific knowledge and furthermore, it is difficult to check the accuracy of these requirements (Hon, Gayen and Ehrich 2008). Using natural language for specifications is also prone to be culturally dependent and therefore to result in ambiguous or unclear meanings (Yang *et al.* 2011).

In order to mitigate this ambiguity, formal specifications, rather than specifications in a natural language, are proposed as they are unambiguous, analysable and facilitate rigorous testing procedures. However, there are disadvantages of using formal methods for specification and latter workflows, as they are highly dependent on stable and strictly defined requirements. If their requirements are not well defined, as is often the case, formal specification is impractical. In addition, formal methods are not practical where the target implementation language is not formally defined and, consequently, there cannot be a crossover from formal requirements to implementation phases (Gibson and Méry 1998).

Most software developers are unwilling to use existing formal methods because they require a huge learning curve as well as enormous effort. This time and effort spent to implement a formal method may not be worth the advantage provided by the method. In some cases, this effort could be better spent implementing an alternative method, such as simulation (Heitmeyer 1998). These formal specifications are very labour intensive and hence have an increased cost in the process. For such processes to be used personnel must be highly skilled in the methods to be used and as they are tedious and expensive they are seen as suitable only for critical systems and hard problems, where traditional methods are ineffective (Rushby 1997).

Development methods such as structured analysis and structured design for traditional languages emerged in 1970s and became widespread in the 1980s. However, these methods did not have uniformity. Unified Modelling Language (UML) consolidated diverse structuring modelling notations including the emerging object oriented notations that achieved penetration into the large system area. Both the structured and emerging object oriented modelling methodologies had their own concepts, definitions, notations, terminologies and processes. Once the object oriented language became prominent a number of authors produced a range of books on object-oriented methodology, each with its own concepts, definitions, notations and terminologies (Watson 2008). (Britton and Doake 2004) note different notations to denote process and data flow within the same system. Each of the authors used and standardised the best methods of structured analysis and design which maintained both the static and dynamic views of the program. In 1995 Grady Booch, James Rumbaugh and Ivar Jacobson combined concepts to form UML (Booch, Rumbaugh and Jacobson 1999). The final collaborative effort by these three together with many others resulted in the final version of UML, now a de-facto standard in RE (Swain, Panthi and Behera). UML has become the industry standard for requirements specification that are used by analysts. The uniformity in UML has led to the development of tools that provide multiple perspectives for analysis. It is easier to integrate the various phases using these UML processes and tools.

An object-oriented process modeling approach provides a more holistic view of business operations as it models the mechanistic processes of business along with its human interactions (Kosalge and Chatterjee 2011). This allows objects to be easily understood by client stakeholders. This object-oriented paradigm regards both attributes and operations to be equally important and looks at an object as a unified software artifact that includes both the attributes and the operations performed on the attributes (Schach 2008). The object-oriented approach to software development has a decided advantage over the traditional approach in coping with complexity (Munassar and Govardhan 2011).

Object oriented processes illustrated through UML diagrams are easily understood by client stakeholders. Several different UML tools can be used during the analysis and design phases. In this section the use of scenarios and use cases are explained. UML has multiple perspectives – for example, scenarios and use cases are used for end users whilst activity diagrams and sequence diagrams are used for developers. Use cases are a requirements discovery method that were first introduced in the Objectory method (Jacobson et al., 1993). Objectory is an object-oriented methodology that uses design method called ‘design with building blocks’. With this building block design in mind, UML was designed to be used with object oriented design and development as the dominant modelling standard. Industry experience and research validates that the UML reflects some of the best modelling practices and that it includes notations that have been recognized as useful in practice. Yet, basic UML has the disadvantage of lack of modelling precision (Evans *et al.* 2014).

In order to address the lack of modelling precision, (Chanda *et al.* 2009) proposed a formal model for UML activity diagrams which includes correctness, traceability and consistency rules for activity diagrams and for inter-diagram. This model checks for correctness, traceability and consistency between use case events, activity events (analysis phase) and class events (design phase). It checks for the above criteria by using predefined rules and depends on the definition of the different UML diagrams in a formal way. It does not check for any variance between the initial defined requirements and those translated to UML constructs. (Bhattacharjee and

Shyamasundar 2009) explored the specification of operational semantics for the activity diagrams of UML 2.0 for simulation and code generation. However, the resulting model was not verified. Consequently the subsequent code generation and simulation components could not be used. This highlights the need to verify a design model before implementation becomes feasible.

A use case is a coherent unit of functionality expressed as a transaction between the software product itself and the users of the software product (actors) (Booch, Rumbaugh and Jacobson 1999; Schach 2008; Sommerville 2011; Satzinger, Jackson and Burd 2012). Use cases are extensively used to document user requirements and to drive the software development process (Juan Zheng, Liu and Liu 2010) (Simmons 2005). A use case generally describes several scenarios that will allow an actor (usually a system user) to benefit from the services offered by that use case (El-Attar 2011). Further (Bee Bee, Bernardo and Verner 2010) and (Kof *et al.* 2010) also recommend employing use cases to determine requirements as they are easily understood by users. Modelling requirements employing use cases therefore become a viable option. There is no hard and fast rule in defining scenarios and use cases. Some people consider that each use case is a single scenario; others, as suggested by Stevens and Pooley (2006), encapsulate a set of scenarios in a single use case. According to Pooley, each scenario is a single thread through the use case, thereby consolidating the common scenario into a single use case. This has the advantage of guiding the number of use cases in a system to the core requirements. Any variation of the core feature will be encapsulated within a use case. A scenario may be used to illustrate an interaction or the execution of a use case instance (Booch, Rumbaugh and Jacobson 1999) and (Sommerville 2011). A scenario is a specific instantiation of a use case, just as an object is an instantiation of a class (Schach 2008). Thus, a scenario is a unique set of internal activities within a use case and represents a unique path through the use case. A fully developed use case increases the probability of a developer thoroughly understanding a business process and the ways in which the system must support them (Satzinger, Jackson and Burd 2012).

Scenarios provide partial rigour in specification, in comparison with natural language specifications that are understandable by the end user (Somé 2005). Scenarios' rigour stems from pre-defined structures and rules that are understandable by users. Use cases provide a framework for grouping and organising related scenarios (Somé 2005). One disadvantage of scenarios is that if each scenario is required to cover all the exceptional cases, the requirements specification will suffer from scenario explosion and redundancy. Consequently, software developers, unless in certain domains (such as critical systems) may not detail exceptional circumstances as this would make specifications unwieldy.

Another modelling method is the use of formal methods. Many different modelling notations support the precise formal description of requirements. These formal methods support the reasoning which helps achieve completeness and consistency in the specified requirements (Kof *et al.* 2010). However, formal languages have not found general acceptance as the level of expertise required is fairly high (Krishnan 2003). In order to use formal languages, the stakeholders who approve requirements must be familiarised with this language, which often entails the involvement of formal language experts (Woodcock *et al.* 2009). A formal verification of a system will need to be repeated every time requirements change which is not economically viable (McDermid *et al.* 1998). Inability to scale up to larger systems, specificity to a particular technical environment, and lack of corresponding formal constructs in modern programming languages due to their complexity or constraints, are some of the disadvantages noted by (Kneuper 1997). Furthermore, most formal languages do not allow seamless transition from one phase to the next: for example, Z is used for specification but it is not continued in the later phases (Crow and Di Vito 1998).

One method within formal methods is the implementation of petri-nets to represent a program which requires that all possible states of the program be explored (Peterson 1977). Although these methods guarantee consistency at a particular phase in a life cycle, they are time consuming as they require exhaustive formal analysis and hence become very costly. In addition it is difficult to model every possible state (Heitmeyer 1998). This will result in a combinatorial explosion that will become difficult to

manage, or to check, and hard for end users to understand. Formal approaches, such as the use of petri-nets, are therefore applicable in limited and specific domains and their corresponding tools are constructed for research purposes and have limited usability. Defining every possible state for non-trivial programs is difficult and time consuming and exploring all possible states is time consuming. Object Constraint Language (OCL), a semiformal notation for UML diagrams is also tedious and time consuming to document. Furthermore, the normalization of post conditions in OCL needs preferred patterns while modelling operations. Moreover, many transformations are required before test data can be generated and complex types of redundancy or inconsistency have to be identified manually (Aggarwal and Sabharwal 2012).

2.9.4 UML Activity Diagrams

An activity diagram is one of the nine diagrams in the UML, which shows the dynamic aspects of systems (UML 2005). Activity diagrams emphasize the flow of sequential or concurrent control from activity to activity.

The following UML use case diagram is used to identify the actor, use case and their interaction. The actor can be a human or an external system (Bruegge and Dutoit 1999). The use cases are illustrated using the symbols in Figure 2.3 where the actors include persons and / or subsystems interacting with the system.

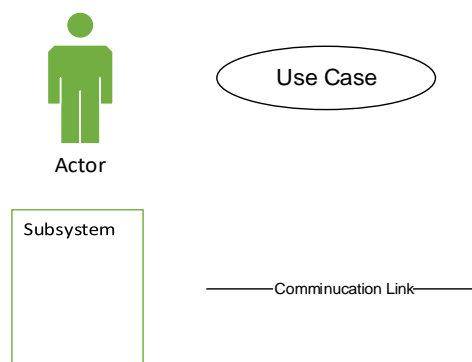


Figure 2.3 use case Diagram Key

Although Xuping categorises various methods under RE, there is often an overlap between the requirements phase and the early stages of the design phase. Some of the tools mentioned above can also be used during various software phases to achieve different objectives. For example, prototyping may be used to clarify requirements as well as demonstrating the feasibility of a design (Azeem and Gondal 2011).

2.9.5 Requirements Evaluation and Validation

2.9.5.1 Validation and Verification

There are varying definitions given by different authors of verification and validation. According to (Boehm 1984) verification refers to the process of establishing the truth of the correspondence between a software product and its specification. Static methods such as expert opinion, walk-throughs, inspections and reviews are used to establish and document whether items, processes, services or documents conform to specific requirements and whether the products of a given development phase satisfy the conditions imposed at the start of the phase (ESA Board for Software Standardisation and Control (BSSC) 1994; Sommerville 2011). Expert opinion is also used to resolve any ambiguity in specifications (Bakhouya *et al.* 2012). Requirements modelled as use cases, activity diagrams, and interaction diagrams are used in the process. Walkthroughs with experts having the requisite domain knowledge is a valuable method when the stakeholders giving the requirements have the time and skill to communicate these clearly. Inspections and walkthroughs are two types of reviews that serve as important processes by a team of experts in order to identify errors as early as possible in the software development life cycle (Schach 2008). Each of these methods will have its own disadvantages and advantages. However, the advantages experienced when combining the appropriate methods given above will outweigh the individual disadvantages of each of the methods.

Validation refers to the process of establishing the fitness or worth of a software product for its operational mission (Boehm 1984). This will entail evaluating a system at the end of the development process to determine whether it satisfies specified require-

ments (ESA Board for Software Standardisation and Control (BSSC) 1994). Static validation has limitations and therefore this disadvantage is addressed through dynamic methods of validation (Ling, Jing and Xiaoshan 2009). Because developers and testers consider verification difficult for software engineers, dynamic analysis (such as rapid prototyping) is regarded as an effective alternative method for model validation. Verification of specification does not imply validation, consequently methods such as dynamic analysis must be used for validation (Bakhouya *et al.* 2012). Dynamic analysis has the ability to check properties which is very difficult and costly using static analysis (Colcombet and Fradet 2000).

Systems for modelling and simulation are developed to model physical processes or situations, which include many separate, interacting objects. Modelling and simulation are often computationally intensive and require high-performance parallel systems for execution (Sommerville 2011). Simulation is best suited for systems that are to be used in a life threatening or dangerous environment. Simulation is usually carried out in a closed environment where the variables are known and processes have been verified. Simulation sometimes becomes very costly and complex to set up and this cost will need to be justifiable prior to its implementation. The domain of the system determines the level of accuracy in the evaluation and validation required and therefore any simulation must be representative of the actual scenario. Furthermore, variations and responses encountered in the simulation are restricted to the designer of such a system. Critical systems with well-known scenarios that are subject to costly failures require verification and validation through simulation (Ouyang 2014) and formal modelling (Ostroff 1992).

2.9.5.2 Existing Tools for Validation and Verification

There are only a limited number of formal verification tools available to verify incomplete requirements that justifies the need to develop a verification tool. In most cases, tools are not available and, even where they are; they are inflexible and have limited support for various languages and methodologies. (Kneuper 1997).

Automatic verification and validation which includes checks for consistency, completeness and dependability requirements of activity diagrams, was conducted using graph transformation by (Rafe *et al.* 2009). These researchers used a graph transformation approach to check automatically for aspects of verification and validation. However, this transformation involves manual intervention at times, and is very time-consuming. While, for example, informal analysis and requirements and design reviews are possible, the lack of precise semantics for object oriented modelling makes it difficult to develop rigorous, tool-based validation and verification procedures (Evans *et al.* 2014).

A number of researchers investigated how activity diagrams could be used in the validation and verification process in a Software Development Life Cycle (SDLC). Linzhang et al's research checked for inconsistency between the implementation and the design phases by using the test cases generated from a formalised UML activity diagram through a gray-box method. Linzhang's research restricted fork nodes to two exit edges only. Furthermore, concurrent activity states can not access the same object and concurrent activity states can only execute asynchronously. These limitations will place restrictions on the way in which business processes can be modelled using activity diagrams. Again, a Depth First Search (DFS) method can be used to traverse a tree structure. However, a tree structure is not suitable for business process activity diagrams on account of their loops and concurrent structures. In order to traverse an activity diagram (Linzhang *et al.* 2004) stipulated the constraint that the decision path be executed at most once and that all action states and transitions be covered in order to get all basic paths – but the constraint of traversing a decision at most once negates the free flow of paths through an activity diagram. Although a DFS graph search may be used to traverse certain graphs, this type of search is not suitable for business process activity graphs due to its cyclic nature

A review of tools such as Use Case Editor (UCed) and Attributed Graph Grammar (AGG) that transfer specifications to the design phase, indicates that these are domain specific, costly to utilise, and have a steep learning curve. The AGG is a development environment supporting an algebraic approach to graph transformation (Taentzer 2003)

while a UCed tool produces validated requirements in the form of use cases and scenarios (Somé 2007). These tools have the disadvantages of being implementation specific and in addition, they are mainly developed for research purposes. The UCed supports use case elicitation, clarification, composition and simulation. The approach is rooted in the Unified Modelling Language (UML) (Maiden 1998). Attempts to obtain and utilise this tool to perform any validation were unsuccessful.

The AGG tool environment consists of a graphical user interface supporting several kinds of validations which comprise graph parsing, consistency checking of graphs, applicability checking of rules sequences, and conflict and dependency detection by critical pair analysis of graph rules (Runge, Ermel and Taentzer 2011). This model will require a complete, detailed design from which the rules will translate into graphs. It is therefore useful for validation during the later stages of implementation. Furthermore, the lack of a construct to match and transform collections of similar subgraphs makes graph transformation complex, or even impractical in a number of transformation cases (Grønmo, Krogdahl and Møller-Pedersen 2013).

2.10 Conclusion

This chapter presented a literature review of the digital divide, Telecentres and RE. The research highlighted the need for a common model, TeleMun, to gather the data needed for sound business decisions, including sustainability. Although sustainability of Telecentres is discussed by many researchers, the lack of systems to perform electronic monitoring and reporting is also highlighted - indicating the necessity of ensuring that accurate and timely data is available.

Several RE models were presented in order to develop arguments for this monitoring model. It was found that the use of scenarios and use cases for RE is commonly accepted practice in industry. The different methods and tools have their advantages and disadvantages. By using a combination of different methods, the advantages can

be maximised and the disadvantages minimised. Therefore, a combination of methods is recommended to reduce, if not eliminate, ambiguity and to ensure completeness.

The literature on RE indicates a gap in going from scenarios to use cases as well as to activity diagrams. Manual processes used can miss some of the scenarios and therefore there is a need for a tool to perform checking between the scenarios obtained during the requirements and the design specifications. Against this background, there is a demonstrable need to develop a TeleMun, which can effectively monitor Telecentres and distribute this information to relevant stakeholders, researchers and sponsors.

CHAPTER 3 – Methodology

3.1 Introduction

The literature review identified the importance of Telecentres and the need for a monitoring model to collect their operational data. Methods of developing this model were researched and the most appropriate methods were identified. However, the research also identified a gap within these methods in terms of an appropriate phase consistency tool to ensure consistency between the requirements and design phases.

After the most suitable methods were identified, including design science methodology with the object oriented analysis and design approach, the requirements were obtained and documented using a variety of methods in UML. The requirements were analysed, consolidated and verified using these appropriate methods. The set of requirements and the ensuing draft design model were then iteratively refined until all requirements and the model were finalised. Using design science methodology, a tool (VeriScene) was also developed to fill the gap identified in the literature with respect to phase consistency between requirements and design stages, and this tool was tested in different domains for correctness. Once verified, it was applied against the model for requirements verification.

3.2 Research Approach

The approach required to conduct research involves the intersection of philosophy, research designs and specific methods (Creswell 2013). The selection of the research philosophy, design strategy and methods will depend on the discipline, beliefs and experience of the researcher and this will form the basis of the research methodology and design.

3.2.1 Research Philosophy

After considering the paradigms outlined in the literature and evaluating them against the research environment, it was decided that positivism was best suited for the research. Rather than using the research based upon the viewpoints of participants in

the research environment [constructivism], validating an existing theory [post-positivism], or advocating change on behalf of these participants [transformativism], this research environment was constrained to collecting empirical monitoring data from machines which best fits the positivism worldview. Because this research fits the positivism worldview well, there is no need to consider other worldviews [including pragmatism].

Positivism was considered the most appropriate worldview as it incorporated mathematical formalism, reductionism, and reliance on existing methods and theories for both development and quality. The reductionist approach described in the literature review was used to reduce the natural language requirements identified by the Telecentre managers to defined scenarios that were further consolidated into use cases. Through the literature review and stakeholder input, ill-defined processes were iteratively refined (through reductionism and design science) until they were definite and clear enough to be translated into UML. Possible scenarios provided by Telecentres were reduced, through commonality, to a set of scenarios and similarly with a set of use cases. Reductionism was also utilised to reduce the set of attributes gathered by Telecentres to a common set of attributes to be incorporated in a model. Thus, using reductionism, a common set of data attributes to be measured was identified. Positivism is the most suitable conceptual framework for this methodology. TeleMun was further enhanced and verified using semi-formal action rules via the VeriScene tool, this use of mathematical formalism also indicates a positivist approach, which is often based on mathematical formalism in order to develop an empirical model. A positive along with an empiricist view [that phenomena can be sensed and evaluated] (Deshpande, 1983) was therefore adopted and this was then followed by system development methodologies.

3.2.2 Research Design

This Telecentre research warranted both the use of quantitative and qualitative research. One of the goals was to streamline and standardise the Telecentre processes into a common consistent yet flexible workflow. The small number of participants

available and accessible for the research entailed the use of some qualitative methods. The problems were unknown at the beginning and needed joint exploration between the participants and the researcher.

The researcher therefore used qualitative methods to explore the problem, and quantitative methods to analyse and formulate a solution. The qualitative methods helped to understand and explore ill-defined processes. These qualitative explanations of scenarios were then transformed by quantitative methods into structured and consistent requirements. These requirements were then used in the RE workflows to produce a uniform set of processes that would yield a consistent monitoring data set. A combination of different methods was used to validate the requirements and the derived model. These methods, housed within design science, included walkthrough [requirements and prototype with Telecentre managers], expert opinion [confirming and verifying requirements by Telecentre managers], and prototyping [demonstration of prototype screens to verify processes]. The process involved combining methods in such a way that the advantages of the methods were capitalised (Yeasmin and Rahman 2012).

3.2.3 Research Method

A suitable research method had to be selected. One method, a strategic approach, involves BPR to align the process to the business objectives. This approach is used at a higher level as compared to the process and model design level that this research considers. In the case of this Telecentre research, the processes were not well defined and so had to be first defined and then formalised into a model. Because the initial processes of Telecentres were not well defined, applying reengineering to these processes would only result in a further ill-defined mode.

A method that allowed iterative refinement of these processes, such as Design Science, was therefore needed. Design Science was combined with the well-known and utilised Software Development Life Cycle (SDLC) to encapsulate the requirements and to

design a validated and partially verified TeleMun. These methods are described in detail in the following sections.

Figure 3.1 illustrates the progression of the process followed, together with the relationship between the tool and the phases, models, and artefacts of the SDLC. The phases of SDLC are followed according to the requirements needed to design for the TeleMun and to the requirements of tool construction for VeriScene. The equivalent models of these phases are textual description, UML and Design Science. Design Science was used to construct the TeleMun model and VeriScene at the tool prototype phase. The research results include the equivalent user scenarios, TeleMun activity diagrams, and lastly the scenarios generated by the phase consistency tool. The generated scenarios were compared with the user scenarios to ensure that all scenarios matched.

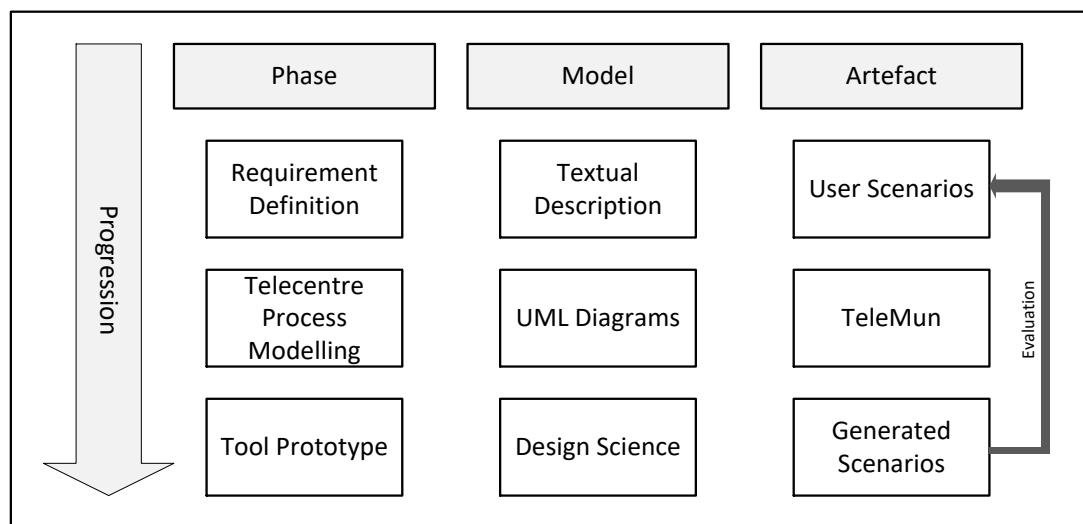


Figure 3.1 Phase, Model, and Artefact

3.3 System Development Methodologies

Given the previous choices of the components of research design approaches, the most appropriate high-level methodology for the systems development approach needed to be selected. Given different methodologies outlined in the literature review, design science was chosen as the most appropriate due to its iterative nature. Several iterations between the requirements and design phases were used because requirements were not

completely specified, or were ambiguous. For software development to be successful there must be several iterations between the development and evaluation phases as all requirements are not identified or completely specified at the beginning of the development phase.

Design science is a well-established software engineering practice with a good success record in industry as it encompasses an iterative element that most information systems modelling the systems development require. The Design Science process consists of the following phases: problem identification and motivation, defining the objectives of the solution, design and development of the solution, and demonstration, evaluation and communication of the solution. It also provides for feedback loops and iterations that were required for TeleMun as the requirements, as explained above, were not well defined and therefore needed several opportunities for confirmation and reworking/redefinition. This is illustrated in Figure 3.2. This approach suited the development of a validated TeleMun.

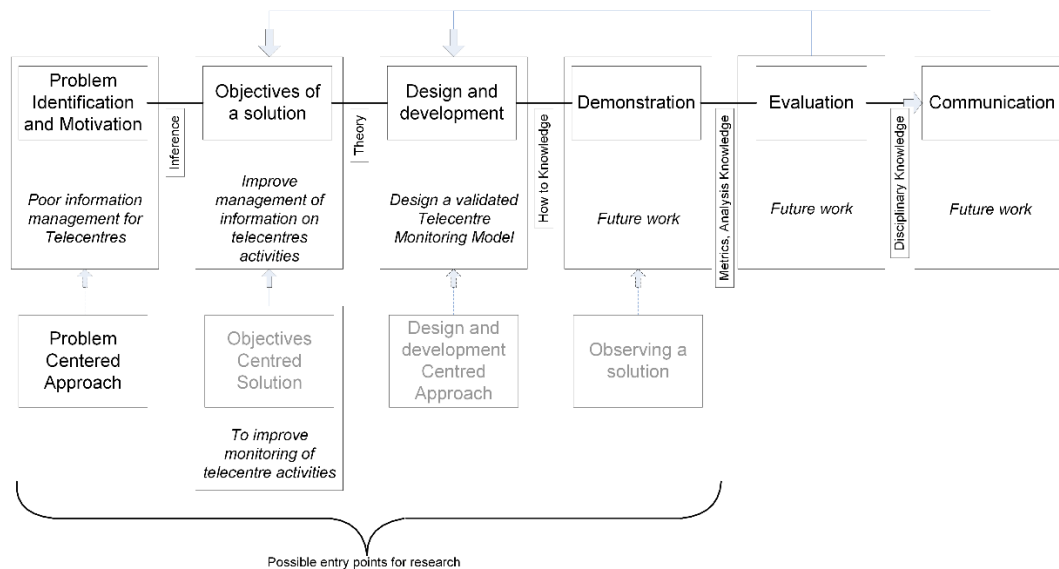


Figure 3.2 Design Science for TeleMun

Various languages / models can be used to document requirements, analysis and design artefacts including formal methods, BPR, and UML. The reasons that formal methods and BPR were not considered are highlighted in the literature and these reasons include complexity of understanding in formal methods and BPR being suitable for large

projects only. Given the size of the system, UML was chosen over BPR as UML offers multiple perspectives of the system for various groups of stakeholders. The emergence and advantages of UML are discussed in the literature review by reference to (Booch, Rumbaugh and Jacobson 1999). UML activity diagrams were used to model the Telecentre activities with specific emphasis on monitoring. UML has features for all workflows of the systems development life cycle. Therefore, models from one workflow can easily be migrated into the next workflow. For example, the Telecentre use cases, formulated during requirements elicitation, can be migrated to the activity diagrams and then later to the system sequence diagrams used in detailed design. UML is a standardized, object-oriented, visual language for modelling software intensive systems (Milicev 2008) which offers multiple perspectives of TeleMun. These multi perspectives help in that the scenarios, as well as activity diagrams, are easily understood by the end users (Telecentre managers) whilst the developers will use the use cases to proceed to a detailed design and then implementation.

Scenarios described by the Telecentre managers are grouped into use cases that can be easily modelled. The use cases are documented using standardised templates described in section 3.4.2. One or more use cases are modelled in activity diagrams to represent the process flow. Each use case is allocated a unique identification number for easy identification, which is later used to associate it with the scenarios. The use case is also related to a specific requirement obtained from the semi-structured interviews and each use case achieves a specified goal. UML includes constructs such as scenarios, use cases and activity diagrams to describe requirements. These give multiple views of the requirements at different levels as well as adding detail to the requirement – for example, the activity diagram shows the process flow from one activity to the next, whilst the scenario indicates a single thread through a use case. In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role (known in the UML as an actor) and a system, to achieve a goal.

Although UML offers many types of diagrams that provide different views of the system, the ones that are relevant to TeleMun, and that are relevant to the stakeholders,

were chosen. These relevant diagrams are activity diagrams, scenarios and use case. Other types of diagram such as state charts and collaboration diagrams required more formalisation. Consequently they are often used to model real time systems which require precision and the ability to handle reactive and exceptional behaviour (Köhler *et al.* 2000). These diagrams lend themselves to object oriented design and development (Satzinger, Jackson and Burd 2012). Use case diagrams and descriptions, together with activity diagrams were used, as these were more understandable to the Telecentre managers who were not familiar with the details of software engineering.

3.4 Application of Design Science Methodology

Because the business processes were not well-defined in this case system, the traditional method of using a series of semi-structured interviews from stakeholders was used to discover initial system requirements (Xuping 2008). The requirements for Telecentre monitoring were not known or, if known, they were not well defined; therefore, an iterative model was used to ensure that the experts were consulted and their inputs were fed back into the draft model for further refinement. Although experts knew the domain, they were not equipped to express this domain knowledge as well defined requirements for the development of an information system. The iterative model is well suited for a domain where requirements are unclear, documentation non-existent and processes ill defined. The defined processes and a common data set were not formalised for this environment as would be required to proceed to the design phase.

As indicated in Figure 3.2 the requirements (or specification) phase in design science produces a textual description of the requirements that are then translated into UML user scenarios. After the specification phase, the business process construction (or design) phase produces a business model that is translated into a UML activity diagrams

Due to the unclear requirements and consequent ill-defined processes and data set, various strong verification and validation methods were required to ensure that the

requirements captured were indeed a true reflection of what was needed. This required the selection of a triangulation of different methods to ensure a verified and validated TeleMun at various stages of the development process. The process followed to develop TeleMun is illustrated in Figure 3.3. This process began with requirements elicitation followed by analysis.

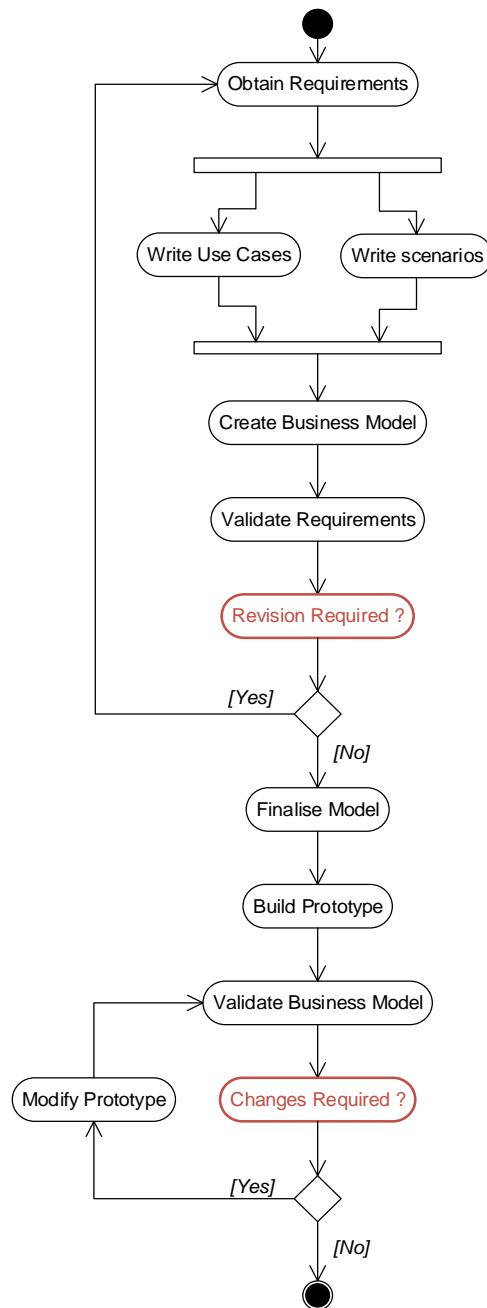


Figure 3.3 Methodology to build and verify TeleMun

3.4.1 Requirements Elicitation

The design and development phase of the design science model began with requirements elicitation. The principal means of requirements elicitation, as explained above, involved qualitative interviews with Telecentre managers. To begin the

requirements gathering process, a series of semi structured qualitative interviews with key stakeholders (consisting of the area manager and Telecentre managers) were used to obtain information on preliminary requirements. Qualitative interviews were used, as respondents are known to be comfortable and familiar with this method. The respondents enjoyed answering questions related to their work, and researchers were able to probe unexpected responses to obtain further or hidden requirements (Lethbridge, Sim and Singer 2005). Because the number of stakeholders was small (a narrow group of Telecentre managers) it was deemed unnecessary to set up focus group interviews. Consequently, a set of requirements consisting of scenarios was drafted which was clarified at subsequent interviews / walkthroughs. This clarification produced the basis for subsequent stages.

The callbox and Internet café business model shown in Figure D1 and appendix D were investigated during the initial stages of the research. One of the dangers of relying on user requirements only, without the various stages of analysis, design and verification along with the use of scenarios and use cases, is illustrated in Appendix D. Appendix D shows the model derived from users' requirements and business processes only, without the above-mentioned stages. Consequently, the flow is awkward and does not allow for flexibility and / or a comprehensive model.

In addition to the interviews with Telecentre managers, requirements were also based on related research, from which a set of commonly used attributes were derived, being those required for monitoring activities at most Telecentres throughout the world. (Pancham and Millham 2015) developed a model for operational monitoring of Telecentre activities based on data attributes that were identified by other researchers and a common set of processes for Telecentres as defined by (Veeraraghavan *et al.* 2006).

This group of attributes was reviewed and verified by domain experts from USAASA. The expert opinions from the area manager and Telecentre managers also formed an integral part of the modelling, within the requirements definition process, including, for example, services offered, services used, and payment before or after usage. These

managers were the most knowledgeable about the domain as a result of their daily involvement in the Telecentres.

The TeleMun is a model designed from these requirements that represents the activities, and their interactions with external entities, that are performed at a Telecentre. These activities and interactions with entities form the basis of scenarios and use cases. The data generated from these are captured for monitoring purposes. This includes the actions that are performed prior and after the service is used. These requirements were then analysed and drafted into scenarios that were reduced to use cases. These use cases were used to formulate the draft activity diagrams that encompassed the corresponding activities of the Telecentre.

3.4.2 Analysis

3.4.2.1 Stage 0 – Reduction of Attributes

The inputs consisting of attributes and processes from the literature in conjunction with attributes and processes identified during the interviews with Telecentre managers, were consolidated and reduced using the researcher’s expertise. This is shown in Figure 3.4.



Figure 3.4 Reduction of attributes

3.4.2.2 Stage 1 - Drafting and Consolidation of Scenarios

Requirements obtained from the interview processes that needed to be collated and organised were consolidated into coherent use cases and scenarios. Initial requirements were unstructured and vague and therefore needed structuring and clarification. Due to limited user stakeholder interaction, initial requirements needed to be organized by the

researcher so that further analysis could be performed. This was the application of the reductionist paradigm. These were documented in a suitable structured way using the UML standard formal definitions of use cases and scenarios as detailed in tables 3.1 and 3.2.

The scenarios formulated from the interviews were consolidated into use cases. The analysis of the scenarios would indicate that the respective use case is analysed implicitly. Similarly, the test data gathered for the scenarios would be applicable to the corresponding use case. Researcher opinion was used for verification and perhaps initial validation ensuring scenarios and use cases were valid. This researcher opinion ensured that requirements from interviews were well structured into scenarios and into use cases that had a logical flow. Using walkthroughs with Telecentre managers together with the researcher’s expertise, these diagrams were checked for consistency and logical flow of events. This process formed part of Stage 1 of analysis as depicted in Figure 3.5, where the initial set of requirements from literature and stakeholders’ input formed the input, and the researcher’s expertise and walkthroughs formed the methodology. The possible set of scenarios and use cases formed the output.

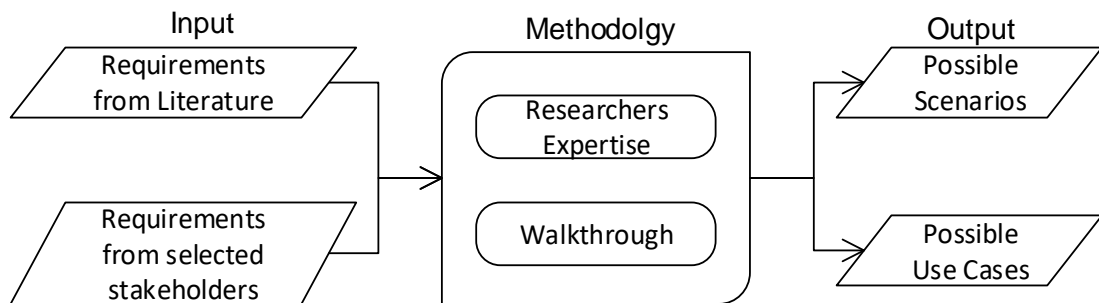


Figure 3.5 Drafting and consolidation of Scenarios

As per the analysis, during Stage 1 the use case template in Table 3.1 was used to describe each of the use cases. The UML notation in Figure 3.2 was used to design the use cases, to indicate their actors and their related functionality. The processes modelled in the TeleMun, and detailed scenarios defined using criteria in table 3.2, were used to create the use cases.

Table 3.1. Use Case Definition

Category	Description
Use Case No	Use Case Identification
Related Requirements	Indicate the requirements this use case partially or completely fulfils.
Goal in Context	The use case's place within the system and why this use case is important

Table 3.2: Scenario definition

Category	Description
No	Scenario Identification
Name	Scenario Name
Preconditions	What needs to happen before the use case can be executed?
Successful End Condition	What should the system's condition be if the use case executes successfully?
Failed End Condition	What should the system's condition be if the use case fails to execute successfully?
Primary Actors	The main actors that participate in the use case. Often includes the actor that triggers or directly receives information from a use case's execution.
Secondary Actors	Actors that participates but are not the main players in a use case's execution.
Trigger	The event triggered by an actor that causes the use case to execute.
Main Flow Action Steps	The place to describe each of the important steps in a scenario's normal execution.
Extension Branching Steps	A description of any alternative steps from the ones described in the main flow.

3.4.2.2.1 Identification of Test Data for Scenarios and Consolidation of Requirements

The Telecentre managers were consulted again and shown the draft scenarios and use cases to obtain possible data for each scenario. (Ogata and Matsuura 2010) agree that prototypes can be used together with concrete test data to validate requirements to the

satisfaction of test data for the user profiles). Scenario usage data was generated to cover all possible combinations of users and service usage. A total of one thousand combinations of usages of the different services was created to provide a wide range of data including exceptional circumstances. This data was used in walkthroughs in various stages of the methodology:

- 1) Analysis Stage 1 – to ensure that there was a logical flow of the draft scenarios and their derived use cases
- 2) Analysis Stage 2 – to ensure logical flow within scenarios for formalisation and reduction (see section 3.4.2.4)
- 3) High level Design – to ensure that the use cases encapsulate the requirements in a complete and logical fashion (see section 3.4.3)
- 4) Verification – to ensure that all finalised scenarios are contained within the TeleMun model and that a logical and consistent flow of activities exists within the design.

Prototyping also utilised this test data to show a realistic functioning proto-system in stage 2 of analysis in order to elicit further requirements and acquire feedback regarding functionality. Prototyping, based on the draft TeleMun, was also used with test data during the verification phase to ensure that the actual outputs produced by the prototype matched the expected outputs. The matching of these outputs corresponds to dynamic analysis of the system, resulting in partial validation of the model.

The drafting of the use cases was an iterative process and each iteration further refined the use cases, the scenarios and the TeleMun. This refinement also follows the reductionist principle as redundancies identified were reduced in scenarios and then in use cases. As an example, the following facilities were offered: internet, fax, and word processing along with other services. These facilities or services were reduced to “Service Offering”. The loop in the first half of Figure 3.3 illustrates the iterative nature of the design science methodology. After analysing the scenario in light of the common attributes achieved, many of the scenarios were condensed allowing for flexibility of the model.

3.4.2.4 Stage 2 - Confirmation of Scenarios

Reverting to the requirement elicitation phase, another set of interviews, showing the consolidated scenarios and attributes to the Telecentre managers, was conducted.

Scenarios have been advocated as an effective means of acquiring and verifying requirements as they capture examples and real world experiences that users can understand (Stevens and Pooley 2006). The scenario template in table 3.2 is used to describe each scenario as it is executed in the live situation. Although theoretically possible, documenting all scenarios and then condensing them is often not done because it is too time consuming and therefore not all scenarios are listed in their full detail. However, consolidation of scenarios occurred during the drafting of use cases. Similar scenarios were grouped as per their common actors and processes in order to form a use case.

Once the possible set of reduced scenarios and use cases was formalised using the researcher's experience and walkthroughs, further interviews were conducted with Telecentre managers to obtain their expert opinion on the findings. Their confirmation of the newly derived diagrams during this Stage 3 of the analysis workflow resulted in a verified set of scenarios and use cases. Furthermore, these possible scenarios, use cases and initial processes were incorporated into a prototype that was demonstrated to the users. Feedback from the users served to clarify some ambiguous requirements and to confirm other requirements. These inputs, methods and outputs are illustrated in Figure 3.6.

Using prototypes, clients gain an earlier and much clearer understanding of a proposed system via an intuitive mock-up (Robertson and Robertson 2012). In this research the requirements from Telecentre managers were not well defined, thus the prototyping methodology presented possible functionalities, screens, and workflows that led to opportunities for discussions about requirements, and for clarification and confirmation.

Once the requirements were confirmed using the prototype they were used in the analysis and design workflows. An object-oriented approach incorporates data and actions within a structured set of processes that are well suited to modern programming. Attributes and processes were identified, with documentation, which can be used for object-oriented analysis for encapsulation into a class design. These attributes and processes led to the formation of use cases. (Schach 2008) states that identification of use cases and actors from the initial analysis workflow which form potential class activities/methods in the detailed design, aids in a smooth transition from analysis to the detailed design workflow in an object-oriented environment

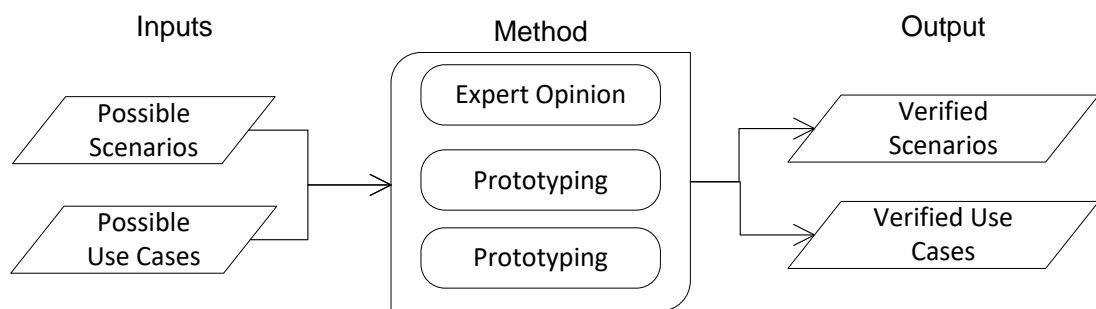


Figure 3.6 Confirmation of scenarios

After user cases and scenarios were finalised, a suite of test data for each finalised scenario, as indicated section 3.4.2.2.1, was obtained. Once Stage 3 resulted in finalised use cases and scenarios, the development proceeded to the high-level design phase.

3.4.3 High Level Design

During the first phase of the design, a draft TeleMun was developed from the valid use cases, scenarios and business processes from analysis Stage 3. Once the requirements in the form of scenarios and use case diagrams were confirmed, these were used to draft activity diagrams. The tasks / sub-tasks of the use cases form the activities within the activity diagrams; the sequencing of activities and paths within the activity diagrams are derived from the paths of the scenarios. The methodology to verify the model included walk through and expert opinion as illustrated in Figure 3.7.

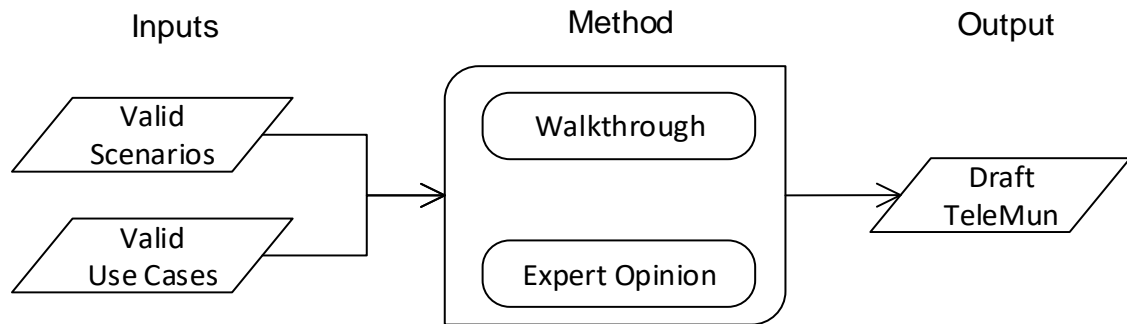


Figure 3.7 High-level design process

This involved a walkthrough of the scenarios with test data for their corresponding use cases to ensure that the use cases were representative of conditions and processes within the requirements. Using expert opinion and walkthroughs, these requirements, in the form of UML diagrams, were verified. Once verified, draft activity diagrams were formulated into a model (TeleMun) to represent the design phase of the software development cycle.

To verify this draft design, multiple traditional methods were used in order to ensure that the requirements were verified and draft activity diagrams were preliminarily validated as the deficiencies of any one method could be overcome by combining methods and by capitalizing on their individual method strengths (see Figure 3.6) (Yeasmin and Rahman 2012). As explained above, expert opinion, in the form of managers who were experts in their domain, was used to ensure that all business scenarios reflected the business processes and events in the draft diagrams. Walkthroughs, with typical business test data, were conducted to ensure that all paths of the activity diagrams were traversed and the expected outputs obtained.

3.4.4 Verification

During the second stage, the draft TeleMun was verified using a combination of expert opinion, VeriScene, prototyping, and walkthrough of the scenarios. These methodologies are illustrated in Figure 3.8. Using a checklist of scenarios and business rules in conjunction with a walkthrough using a full set of test data, it was ensured that all scenarios and all possible paths of the draft TeleMun were covered and that they

were representative of all TeleMun workflows. Expert opinion verified first scenarios as legitimate and then verified the design as encompassing their processes and capturing their data. Prototyping, representing the core feature set of functionality, used dynamic analysis with this test data to match expected outcomes with actual outcomes, in order to ensure that the design functionality was correct thus partially validating the TeleMun. VeriScene was used to ensure phase consistency between the requirements and the design phases [see section 3.4.4.1].

These various methods used in the analysis phase, clarified the originally ill-defined requirements from specification composed in a natural language, partially verified them, and structured them using relevant UML diagrams. The phases transformed the ambiguous natural specifications into well-defined and structured requirements that met the requirements of formal methods for well-defined requirements. The various methods and stages of the methodology clarified ill-defined requirements and ensured their consistency. Unlike the Telecentre stakeholders, the researcher had some familiarity with formal languages, but in order to avoid the steep learning curve of formal methods, a lightweight formal method for verification was selected. Consequently, given that one of the requirements for formal methods is that it requires well-defined requirements; these multi-phased methods enabled the translations that transformed requirements into a formal notation for further verification.

The strength of formal notations within UML diagrams was used to develop a tool to fill the gap, which had been identified in the literature concerning phase consistency tools. The tool provided phase consistency between the requirements and design phases involving model scenarios. Using design science, a tool (VeriScene) was developed to traverse activity diagrams based on scenarios, in order to generate scenarios based on the particular path(s) and activity processes traversed. These generated scenarios, notated in terms of semi-formal actions and action link rules, were compared to the original scenarios to ensure that the original scenarios' flow steps were the same as those generated by the tool. In so doing, this comparison ensured that all of the scenarios derived from the specifications, using traditional methods, had been incorporated into the activity diagram. Consequently, VeriScene ensured that all

scenarios (in the specification phase) were brought into the activity diagrams (the design phase). This design led to the identification and formulation of action rules and an algorithm. Figure 3.8 illustrates the process of design science applied in developing VeriScene.

In order for the project to proceed beyond the design phase:

- the design needed to be validated to ensure that the requirements had no inconsistencies
- the design had to be correct in order to ensure that the system was developed correctly and that
- it satisfied the user needs during the first stage.

A grant was awarded for the full development of the TeleMun and a third party was contracted to perform the development. Consequently, there was a need for a well-designed system that minimized the need for changes and minimized the cost.

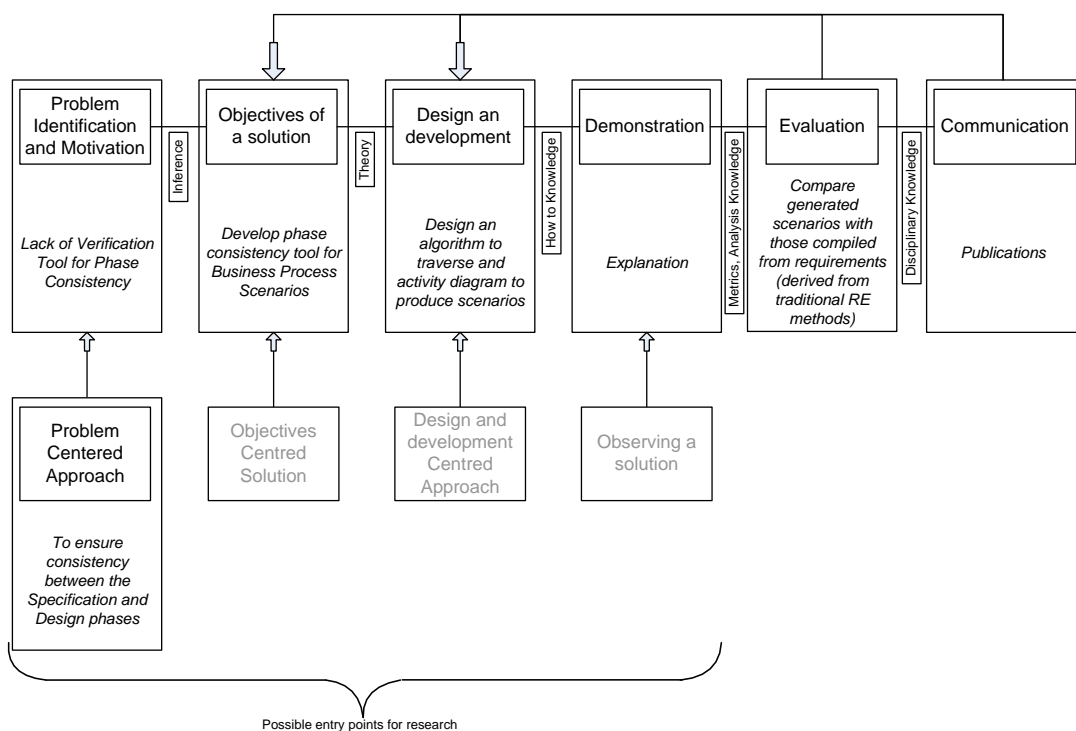


Figure 3.8 DS Verification Process for Phase Consistency

The design science approach caters for an iterative workflow that allows for multiple revisions of specifications and design to ensure correctness. There was an iterative cycle between the design and demonstration phases of the design science to ensure the requirements were accurately captured and verified. The evaluation phase of design science included testing through comparison of the generated scenarios against the consolidated scenarios elicited during the requirements phase. This process ensured that all logic errors were identified and resolved. The success criteria consisted of a correspondence between processes in the model and those in the requirements. Logic errors identified during prototyping and presentation to the user were iterated back to the objectives, design or development phases depending on the results that needed to be obtained. The inputs consisting of scenarios, use cases and the draft TeleMun were verified and partially validated through a combination of methods of expert opinion, walkthroughs, VeriScene and dynamic analysis to produce a completed checklist of use cases from the requirements, verified scenarios and use cases as illustrated in Figure 3.9. These four methods formed a triangulation of methods for TeleMun verification and validation.

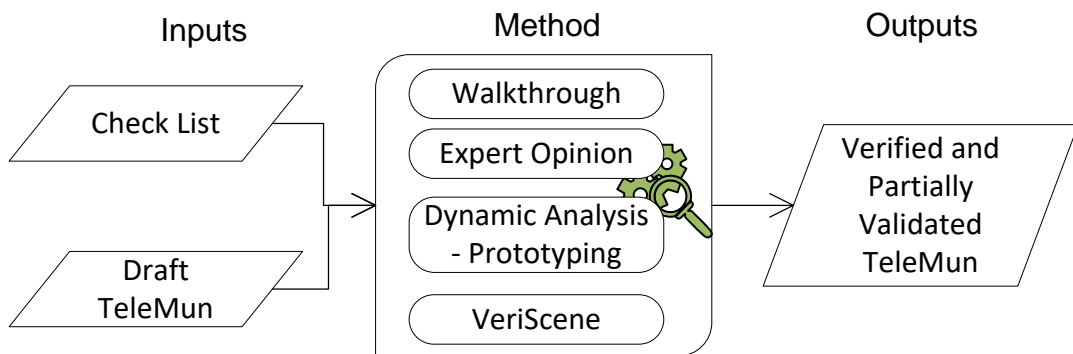


Figure 3.9 Design verification and validation

Using activity diagrams as the basis of development, a prototype was built to provide both dynamic analysis verification and validation (Rantapuska and Millham 2010). In order to simulate the Telecentre operations, test data derived from stage 1 (see section 3.4.2.2) was fed into the prototype and the output produced for each test case was

compared with the corresponding expected outcome. If all matches occurred, this would indicate a partial validation of TeleMun through dynamic analysis. As explained above, expert opinion used was a very accurate method of ensuring coverage of all possible activities and of as far as possible, of all contingencies. Expert opinion was used to complete a checklist confirming that all possible scenarios were covered in the activity diagram, a walkthrough was also conducted with the test data to ensure proper outputs were produced and that all paths were followed. VeriScene ensured phase consistency in that the verified scenarios from the specification phase were incorporated in the activity diagrams.

Figure 3.9 illustrates the validation process of the TeleMun where scenarios, prototype and the TeleMun formed the inputs. The prototype, scenarios and TeleMun inputs were fed into the walkthrough, expert opinion, VeriScene, and dynamic analysis processes to successfully produce a verified and partially validated model.

The need for verification was important to provide developers with a verified set of requirements so that a pilot of implemented TeleMun could be deployed by the developer who had been awarded the grant.

3.4.4.1 Application of Chosen Research Design for VeriScene Phase Consistency Tool

Once the activity diagrams were verified in the TeleMun model, there was a need to ensure that all of the specifications from the earlier phase were brought down and incorporated into the activity diagram. The lack of any appropriate tool that could be identified from the literature led to the second implementation of design science in order to develop such a tool that was needed to ensure phase consistency between the requirements and the design phases. To ensure this, another method, involving a tool to reverse engineer activity diagrams back to their original scenarios, was needed and this tool was consequently developed. This led to the second phase that involved

building VeriScene to verify the scenarios created in the requirements phase, indeed to complete this, and to be consistent.

There was an identified need to perform a formal verification of the design during design stage 2 of TeleMun. Once the inputs were well defined using the scenarios and use cases explained above, the activity diagrams were developed in the TeleMun. In order to ensure correctness and broadness of applicability, two complex activity diagrams from different domains (stock inventory control and trouble ticket), in addition to the TeleMun activity diagram, were selected for the validation of the tool.

Design science was used to develop TeleMun up to the design phase, and design science was used again to develop VeriScene, but in different way. The development of TeleMun followed utilized distinct stages within phases of design science, whereas in VeriScene these stages were blurred. The reason for this lay in the nature of what was being developed. TeleMun needed to be carefully analysed, designed and verified at most stages. The development of TeleMun stopped at the design phase and because it did not lead to implementation, the output could not be used for validation. On the other hand, VeriScene was a full prototyping tool, which produced outputs that could be evaluated against the expected outcomes. Any mismatches would indicate errors, which would iteratively lead to redesigning and redevelopment. Full prototyping by its nature involves rapid development with immediate output for its evaluation. As a result, there is less need to break down development processes into distinct phases and to verify and validate them. Due to the nature of prototyping there is more emphasis on the implementation and evaluation aspect with frequent iterations back to the requirements and design phases (Kordon 2002). Design science by its flexibility allowed both types of artefacts to be developed following its methodology.

3.4.4.2 Requirements

The initial requirements for VeriScene were obtained from the previous phases of the TeleMun development cycle along with various similar tools identified in the literature review, which could be used for verification. Inconsistency between the requirements

from end users and the design derived from this set of requirements could be identified. The requirement of the VeriScene was to ensure that there was consistency between the requirements obtained during the requirements phase, as represented by UML scenarios, and the design phase as represented by UML activity diagrams. The requirements evolved through the iterations within the design science methodology.

Tools reviewed in the literature had identified a gap in existing phase consistency tools between the requirements and design phases (see section 2.9.5.2). It was apparent that existing tools used semi-formal methods and notations. An initial set of requirements was therefore drawn from the literature and these were supplemented from researcher's industry experience.

The requirement was that the tool use existing programming structural notation, particularly UML, rather than proprietary formal structures. In addition, the tool should be easy and simple to use with a short learning curve, in order to mitigate some of the disadvantages of formal methods.

3.4.4.3 Analysis

The first stage in the analysis was to establish suitably feasible and rigorous methodologies to develop VeriScene. Hence, suitable notations and rules that encompassed rigour to formalise UML activity diagrams were investigated. Once these were established the next process was to ensure that the activity diagram could be traversed to obtain the different scenarios.

Based on the definition of a scenario as a single path of a use case (Stevens and Pooley 2006) the requirements of VeriScene were refined to traverse the activity diagram with full path coverage. The activity diagram needs to be traversed so that full path and node coverage are accomplished. Based on the definition of scenario, the use case needed to be traversed in a certain way to produce the list of scenarios from which the use case was derived. In order to traverse the activity diagrams correctly, different methods were identified and considered – for example breadth first or depth first. Due to the

design science methodology, a clear distinction was often lacking between the analysis and design phases. However, there were analysis type tasks such as reducing and clarifying requirements and incorporating new requirements.

The second stage used a walkthrough, check list, and researcher experience to further refine the requirements in order to accommodate advanced UML constructs such as 'fork' and 'merge'. A checklist was used to ensure all nodes were visited. The accuracy of this list was based on research experience.

3.4.4.4 Design

That the model needed to be verified to add rigour was determined during the first application of design science to the model design. This process allowed for iterative development and testing until the desired results were achieved.

The problem of phase consistency between the specification and design phases in the software development process was identified from the literature review. This allowed for the development of an algorithm to traverse an activity diagram in order to produce a set of scenarios. Action and action link rules from another researcher were incorporated within this algorithm to provide rigor and consistency, and the tool was evaluated by comparing its generated scenarios from the activity diagram to those original scenarios from which it was derived. According to the design science evaluation process, the generated scenarios (results) were reviewed and if any discrepancies with the manually created scenarios were identified, these discrepancies were resolved by modifying the algorithm or by determining if there were any unforeseen legitimate scenarios omitted from the original activity diagram. This iterative process is in line with design science iterations of the different phases.

In this way, all possible path traversals and the consistency of the activity diagrams are both ensured. The scenarios were defined manually using action and action link rules as outlined in Table 3.4 (Maiden 1998). The draft activity diagram was also defined using an action to denote a scenario step, and an action link rule to connect the steps

into a consistent flow. The definition of scenarios and activity diagrams using action and action link rules provided some consistency and rigor to these diagrams. The scenarios, as defined by these rules, followed the action flow steps as outlined in the scenario description.

To ensure that the correct design was adopted a suitable verification method needed to be identified and deployed. As explained above, one combination of verification methods (a walkthrough using a checklist) ensured that all possible paths and activities for each scenario were covered and no orphaned activities remained in the activity diagram. Consequently, this checklist was adopted. The inputs to this checklist process were the original scenarios of the draft TeleMun and the generated scenarios from VeriScene. Each of the original scenarios were compared with those generated by VeriScene using a walkthrough and any differences were resolved through modification of the software. Figure 3.10 shows the input, methodologies used and the result of this process.



Figure 3.10 Verification of VeriScene

Action Rules

As indicated in the analysis phase, a suitable semiformal notation capable of encompassing an activity diagram was used. The action rules and action link rules used in the manual specification, the set of initial confirmed scenarios, and the coding of the different structures that are traversed in the activity diagram of scenarios, are as follows (Maiden 1998):

Action Link Rule Definition:

- Strict sequence (A then B): Defines sequential order of actions i.e. action B occurs after the completion of action A

- Alternative (A or B): Defines a choice i.e. action A or action B occurs. This is used in the case of a Branch – Merge condition.
- Concurrent (A and B): Defines a concurrent set of actions where action A and B occur concurrently. This is used in the case of a Fork and Join condition.
- Equal-end (A ends-with B): Define two actions A and B that end together.

Notating of Activity diagrams

The activity diagrams and initial confirmed scenarios were formally notated using the action rules and link rules defined previously. The following nodes, together with transitions, will be used in activity diagrams: Start, End, Branch, Merge, Fork Join, and Guard Condition. Each activity diagram will begin at a Start node and finish at the End node, and all nodes will be linked via a directed transition. The scenarios were formally notated in order to increase structure and rigor which enables easier comparisons to be made.

- Each activity diagram will have one Start node and one End node.
- Each of these actions will be linked to subsequent actions in their paths using the Strict Sequence rule.
- Branch Merge construct rules
 - A Branch–Merge construct will be used in the case of a decision so that a single path can be selected based on a guard condition.
 - An action (generally a question that results in a single guard condition) will link to a decision node using a Strict Sequence rule.
 - A decision will link to a branch using a Strict Sequence rule.
 - Each of the subsequent actions following a branch will be linked using the Alternative rule.
 - The Branch will also link to the subsequent actions using the Strict Sequence rule.
 - Each of these actions will be linked to subsequent actions in their paths using the Strict Sequence rule.

- The last action of each branched path will be linked to the merge using a Strict Sequence rule.
- Each of these actions will also be linked to each other using the Equal End rule.
- Fork – Join construct Rules
 - An action will link to a Fork node using a Strict Sequence rule.
 - The first set of actions of each path following the Fork will be linked using the Concurrent rule.
 - Each of these actions will also be linked to subsequent actions using the Strict Sequence rule
 - The last action of each forked path will be linked to the join using a Strict Sequence rule.
 - Each of these actions will also be linked to each other using the Equal End rule.

Table 3.3 provides a coded description of an activity diagram using the rules to formalize the activity diagram together with the action link rules (Maiden 1998). This table is used by the algorithm to walkthrough the activities, flows, and decision/merge/fork/join nodes of the given activity diagram to generate all possible scenarios. The algorithm takes into account decision and parallel activities during its walkthrough.

Table 3.3 Formalized notation, by action and action link rules, of the Telecentre activity diagram

ID	Action One Name	Rule Name	Action Two Name
30	Start	Strict Sequence	Start of Day
31	Start of Day	Strict Sequence	Request Service
32	Request Service	Strict Sequence	Log User Profile
33	Log User Profile	Strict Sequence	Service Available
34	Service Available	Strict Sequence	Branch-Service Available
35	Service Available-Yes	Alternative	Service Available-No
36	Branch-Service Available	Strict Sequence	Bill Usage
39	Bill Usage	Strict Sequence	Allocated Service
41	Allocated Service	Strict Sequence	Use Service
43	Use Service	Strict Sequence	Successful Usage
44	Successful Usage	Strict Sequence	Branch-Successful Usage
45	Successful Usage-Yes	Alternative	Successful Usage-No
46	Branch-Successful Usage	Strict Sequence	Rate Service
47	Rate Service	Strict Sequence	Merge-Successful Usage
49	Branch-Service Available	Strict Sequence	Join Queue
50	Join Queue	Strict Sequence	Continue Wait
52	Continue Wait	Strict Sequence	Branch-Continue Wait
53	Continue Wait-Yes	Alternative	Continue Wait-No
54	Branch-Continue Wait	Strict Sequence	Service Available
55	Branch-Continue Wait	Strict Sequence	Merge-Service Available
56	Branch-Successful Usage	Strict Sequence	Reuse Service
57	Reuse Service	Strict Sequence	Branch-Reuse Service
58	Reuse Service-Yes	Alternative	Reuse Service-No
59	Branch-Reuse Service	Strict Sequence	Service Available
60	Branch-Reuse Service	Strict Sequence	Refund Fee
61	Refund Fee	Strict Sequence	Merge-Successful Usage
62	Merge-Successful Usage	Strict Sequence	Merge-Service Available
63	Merge-Service Available	Strict Sequence	End of Day
66	End of Day	Strict Sequence	End

VeriScene Algorithm

After the diagrams were formally notated, a suitable algorithm needed to be developed to traverse the activity diagrams to generate scenarios as per the definition. The final VeriScene algorithm, iteratively developed for the scenario generation tool, is expressed in pseudo-code as follows:

```

Create List of Rules
Set Path Traversed
Count Paths Remaining
While there are paths remaining

```

```

Create Scenario Header for a new scenario
Set Available Paths = 1
While Scenario is not complete
  Get next Rule
  If ActionOne = 'Branch'
    IF ActionTwo = 'End' OR Loop Identified
      Set Scenario Complete
      Save Action as scenario action
    ELSE IF ActionTwo = 'Fork'
      IF AvailablePaths > 0
        Save BranchPaths traversed
        Call ProcProcess Concurrent Actions
      ELSE
        Find the next available Path
        IF another path is available
          Prepare to Get Next Rule
        END
      ELSE
        IF AvailablePaths > 0
          Save BranchPaths traversed
          Save Action as scenario action
          Prepare to Get Next Rule
        ELSE
          Find the next available Path
          IF another path is available
            Prepare to Get Next Rule
          END
        END
      END
    ELSE
      IF ActionTwo = 'End' OR Loop Identified
        Set Scenario Complete
        Save Action as scenario action
      ELSE IF ActionTwo = 'Fork'
        IF AvailablePaths > 0
          Call ProcProcess Concurrent Actions
        ELSE
          Find the next available Path
          IF another path is available
            Prepare to Get Next Rule
          END
        END
      ELSE IF ActionTwo = Branch and PathTraversed = 0
        Save Action as scenario action
        Update BranchPathsUsed
    END
  END
END

```

```

        Prepare to Get Next Rule
    END
ELSE
    IF AvailablePaths > 0
        Save Action as scenario action
        Prepare to Get Next Rule
    IF Loop Encountered
        Set Scenario Complete
    END
END
END
END
END

```

```

Proc ProcessConcurrentActions
Set PathsRemaining
While PathsRemaining > 0
    Get Next Rule
    Save Action as scenario action
    IF ActionOne = 'Fork'
        Decrease the available paths
    IF ActionTwo = 'Join'
        Get PathsRemaining
        IF PathsRemaining > 0
            Set Next action to Fork
            Save Action as scenario action
        END
    END
END

```

```

END
Proc UpdateBranchPaths
Get BranchPaths
Get NoOfBranches in Current Branch
While BranchPaths are available
    Get Next Branch
    Increase BranchPaths = BranchPaths + NoOfBranches -1
END

```

3.4.4.5 Implementation

As indicated in the design the algorithm was coded, implemented and evaluated for correctness. If any errors were found the algorithm was refined, re-implemented and re-evaluated. The algorithm was also tested against two different domains viz. ‘trouble ticket’ and ‘order processing’ as indicated in Figure C1 and Figure C2 in appendix C.

3.5 Summary

This chapter described both the different research methodologies that were evaluated and how positivism and reductionism were selected as the most appropriate conceptual frameworks for the research. The research design included both quantitative and qualitative research methods. Design science was selected as the overall methodology due to its iterative nature that suited the development, verification and partial validation of TeleMun with its new/clarified requirements. A combination of different methods included walkthrough, expert opinion and prototyping, was used during different stages to correct errors and refine / manage new requirements.

The model is limited to high-level design as implementation of the model is beyond the scope of this research. As this model is being developed by an outside party, the strength of the methods chosen for requirements elicitation, analysis, design, and verification is demonstrated by the fact that these requirements and design were presented to, and accepted by, the developers without any required changes (Ramjugath 2015). In addition, the methods used ensured the scenarios identified in the requirements elicitation were complete and consistent. A gap identified from the literature in available methods led to the development of a universal software tool, VeriScene, using the same methodology and using some of the same combinations of methods. This tool has also been used on other non-trivial examples to demonstrate its universality. The next chapter presents the results obtained from the application of the selected methods in the design and validation of the TeleMun.

CHAPTER 4 – Results

4.1 Introduction

The methodology chapter identified and justified an appropriate research worldview, approach, research methodology, and methods for the development of a Telecentre operational monitoring model and for the software development of a required phase consistency tool, VeriScene. The Design Science approach was chosen due to its ability to iteratively correct errors found at later stages of the software lifecycle and refine requirements. At different phases of the software lifecycle, various combinations of methods were used to clarify and verify/validate requirements to ensure a proper design. This chapter shows how the requirements were elicited and then, through methods at various stages, requirements/design were verified or errors discovered with iterations back to the appropriate phases for correction. Examples of errors/omissions found at various stages are given to show the usefulness of the multi-step methods in identifying these, and ensuring that the final requirements and design are well structured, consistent, and easily implemented.

The VeriScene tool was designed and developed to verify and partially validate the operational Telecentre model design. The chapter concludes with the results of the comparison of the initial scenarios and those generated by the VeriScene tool.

Scenarios, use cases and activity diagrams were drafted, consolidated, and refined to produce a draft TeleMun using the methods in the various stages of analysis and design outlined in the methodology. Through the methods of verification and validation this draft TeleMun was used to create a final TeleMun. As the requirements were refined, the draft TeleMun was altered to reflect these changes. After the requirements were verified, a final TeleMun was produced. Using the methods outlined in the methodology and shown in Figure 4.1 the final TeleMun was verified and partially-validated.

4.2 TeleMun Model

The TeleMun resulted from the combination of research, observations and close consultation with local Telecentre representatives. The scenarios noted were documented using UML diagrams and were later validated using the VeriScene tool that was designed and implemented by the researcher.

4.2.1 Requirements Elicitation

After a series of interviews with selected Telecentre stakeholders, their initial requirements were obtained. During these interviews, Telecentre managers presented the different scenarios that could be experienced at the Telecentres and the managers outlined the different business processes of the Telecentres. These initial requirements formed the basis of the inputs required for a more detailed analysis. An example of an initial requirement (described as a ‘scenario’) that required expansion was power loss at the Telecentre. When this loss occurs, a record should be kept which records when the loss occurred and when power was restored. Consequently, this requirement was expanded to require a device that will detect both power loss and its duration. This had to be installed in such a way that it could not be inadvertently disconnected by the Telecentre users or managers. This ensured that accurate records of power losses could be kept.

As new scenarios were discovered in subsequent stages, they iterated back to the requirements phase as per the Design Science model. Another example of a new scenario being identified and included was loss of internet connectivity. Since most of the users visit the Telecentres to browse the internet, it was important to establish when, and for how long, this connection becomes unusable.

As indicated by requirements elicitation, the attributes identified must be recorded when a user requests a service. The initial attributes identified through the interviews were supplemented by additional attributes identified from an analysis of the Telecentre monitoring literature. When users request a service that is not available, a record of the user profile and service is made. As part of the Telecentre process, the

user is requested to wait if the service is unavailable and this waiting time needs is established and recorded. This type of data will inform future needs of the Telecentre.

Initial data attributes from interviews identified for monitoring purposes included age, distance from Telecentre, employment status, specific application usage, browser usage and sites visited. Together with this data, the managers identified Telecentre usage scenarios such as fax, photocopier, as well as specific applications usage such as word, and excel. Common data attributes of Telecentre monitoring, identified through the literature review, complemented these initial data attributes. Together these attributes and scenarios contributed to a broad scope of Telecentre operations.

4.2.2 Analysis

Analysis and design are quite closely linked. As a result, the draft activity diagrams form part of the preliminary design phase. As the analysis progressed, the corresponding activity diagrams were refined.

4.2.2.1 Stage 0 – Reduction of Attributes

A total of twenty attributes were identified from the literature review as detailed in Table A1 of Appendix A. Common attributes such as age category, profession, distance from Telecentre, and gender were identified from the literature and from the interviews. During this Stage 0, processes were identified such as use of the internet, fax, photocopier, and preparation of a document, payments for services rendered, and recording of internet and power outages.

4.2.2.2 Stage 1 – Drafting and Consolidation of Scenarios

Once the initial requirements were obtained the researcher's expertise and the expert opinion of the Telecentre managers were used to formalise these requirements into UML scenarios and use cases with reduction and non-duplication. As part of this reduction, walkthroughs of scenarios were utilised in order to identify common and redundant processes and to ensure proper process flow for each draft scenario. During the refinement process, commonalities were looked for and categorised accordingly.

As an example, the Telecentre managers explained that users will visit the Telecentre to write CV's, browse the internet and send faxes, amongst other things, as indicated in Figure 4.1. These were consolidated into a single scenario 'Use Service'. This gives flexibility as fax is being replaced by scanning and email. Several iterations of this consolidation process were carried out in conjunction with expert opinion sessions to affirm that the use cases and scenarios were accurate and a complete representation of the requirements. The test data was reduced and modified to fit the consolidated scenarios. Further reduction and consolidation of events is indicated in Figure B3 where processes are executed once after a "Self Service" decision is made.

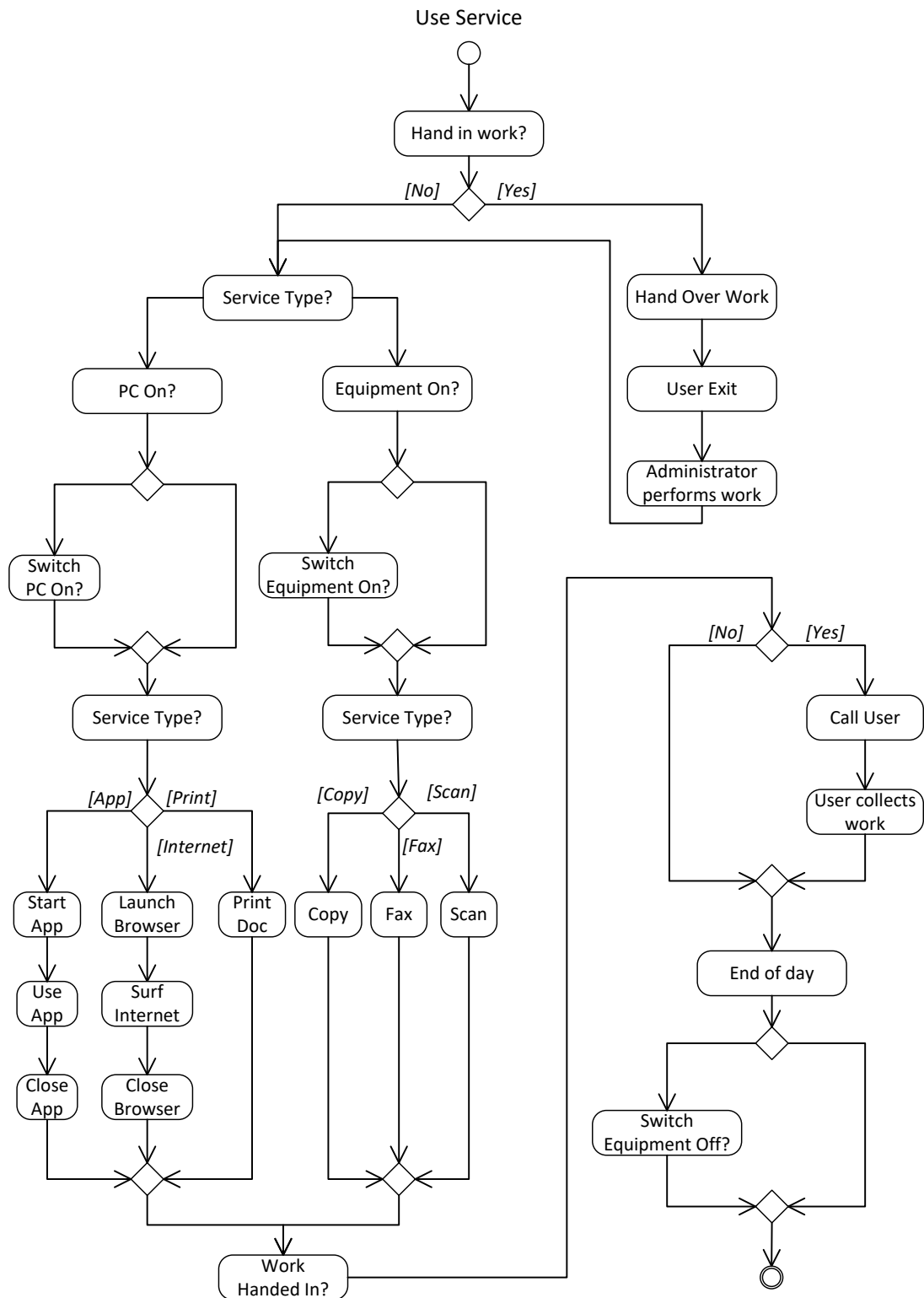


Figure 4.1 Initial use case – Use Service

4.2.2.3 Identification of Test Data for Scenarios and Consolidation of Requirements

Once the scenarios were consolidated each required test data for the walkthrough process to be conducted in subsequent phases. The following scenario shows the sample test data identified as being required for the walkthrough process. A user requests the use of a computer to surf the internet. From this scenario, the data that was required is tabulated as shown in table 4.1.

Table 4.1 Data for scenario

No	Attribute	Typical data
1.	Age	20
2.	Occupation	Unemployed
3.	Distance from Telecentre	5 Km
4.	Service requested	To surf the internet
5.	Payment type	Cash

To use the developed prototype, a suitable test suite was needed. Using the test data that was set up in Table 4.2, 1000 scenarios were generated for a six-month period covering all of them. This test data consisted of a combination of the different age categories, and possible equipment and occupation categories obtained from Telecentre stakeholders, in addition to different scenario paths that these different users might use, including service and service type allocation.

Table 4.2 Sample Attributes

Age Category	Application Name	Occupation Category	Services
Age 15 to 20	Word	Learner	Internet access
Age 21 to 30	Excel	Student	PC
Age 31 to 40	Power Point	Entrepreneur	Print
Age 41 +	Publisher	Unemployed	Fax
	Outlook		Copy
			Scan

The screen in Figure 4.2 below indicates the different services with sample codes that was used for testing.

Application		Reports	Setup	Logs	Services
Code	Name				
FAX	Fax Documents				
INT	Internet				
PCU	PC Usage				
PHC	Photocopy				
PRI	Print Documents				
SCA	Scan Documents				





-  Refresh
-  **New Service**
-  Edit Service
-  Delete Service

Figure 4.2 Services

The sample Figure 4.3 below indicates the age category that was used by the prototype to generate the test cases. These age categories are used to profile the users of the system and can be used later to generate statistics.

Application		Reports	Setup	Logs	Age Category
Code	Name				
A	Between 15 to 20				
B	Between 21 to 30				
C	Between 31 to 40				
D	Older than 40				





-  Refresh
-  **New Age Category**
-  Edit Age Category
-  Delete Age Category

Figure 4.3 Age Category

Once the profile is created the administrator checks to see if the equipment is available. The administrator will select this from a list and allocate the appropriate one to the user. Available equipment categories are listed in Figure 4.4.

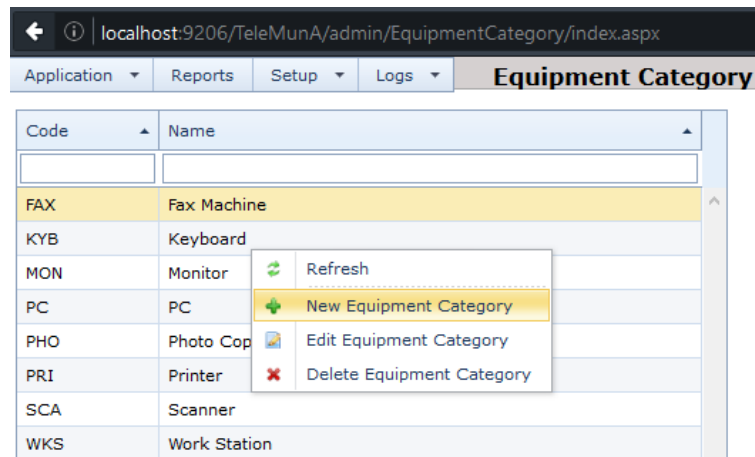


Figure 4.4 Equipment

The above set of test data, together with the features to capture them, formed a basis for the subsequent phases. The test data contained inputs that would be required by different methods at different stages to provide full test coverage of scenarios, use cases, and activity diagrams.

4.2.2.4 Stage 2 – Confirmation of Scenarios

The combined methods of researcher's opinion and walkthrough produced a consolidated set of scenarios with a logical flow but these scenarios may not have been representative of every one of the business processes of the Telecentre. Consequently, there was a need for stage 2, which involved prototyping and expert feedback from Telecentre managers. Although the scenarios were logically correct, they did not reflect the actual business processes executed within the Telecentre environment. The combined methods of stage 2 indicated that there were design errors and non-universality of UML diagrammed scenarios concerning the business process. For example, the billing process is executed at the beginning in some cases while at other Telecentres, the user is not billed at all, or the user is billed at the end of usage (as are, for instance, UNISA students). Therefore, when the prototype presenting the Telecentre scenarios and functionality was presented to the managers they were able to identify that these scenarios did not reflect the actual business process universally for all of their Telecentres. From the prototype demonstration and expert opinion feedback, this information was, however, able to be corrected, and then confirmed.

After consolidating the identified scenarios, they were condensed into logical use cases. As an example the two scenarios, UC09SC01 (Payment Cash for service) and UC09SC02 (Bill Account for service), as described in Table 4.8, were consolidated into a use case (make payment) indicated in Figure 4.9. These two scenarios and their corresponding use case formed the activity diagram Figure B2 in appendix B. This use case contains a decision with each decision path constituting a scenario (either UC09SC01 [Payment Cash for service] or UC09SC02 [Bill Account for service]).

Fifteen use cases, listed in table 4.3 covering all possible scenarios, was created after consolidation and verification. Five of the use cases that are related to user profile and service usage, which provide the most needed information required by the Telecentre managers and researchers, are explained and shown. In this chapter these five main use cases, together with one scenario for each of the use cases, is discussed. They are request a service; allocate service; complete usage of service; bill usage; and make payment. The remaining use cases and scenarios are listed in Appendix E. Each of the use cases were documented in Tables 4.4 to 4.8 using the template. These were complemented by use case diagrams in Figures 4.5 to 4.10. Besides use cases related to the service offering, additional use cases were identified to monitor power and internet connectivity, as listed in Table 4.3. These use cases were included due to the intermittent nature of the internet connection and power supply in the areas where the Telecentres are situated. (The use case diagrams are included in appendix E).

Table 4.3 provides a full list of use cases produced in the final iteration of analysis. These include scenarios that were discovered and / or refined during the design phase and (as explained above) led to the inclusion of power outage and internet loss cases.

Table 4.3 Use Cases

No	Name	Description
UC01	Acquire Equipment	Maintain an inventory of equipment that will be used at the Telecentre.
UC02	Dispose Equipment	Removal of unusable equipment from inventory
UC03	Request Service	The users request for a service from the administrator is logged. This service is executed either by the administrator (full service) or by the user.
UC04	Allocate Service	A User is allocated a service by the Administrator
UC05	Start Application	Log the start of an application
UC06	Terminate Application	Log the close of an application
UC07	Complete usage of service	Log the completion of usage of a service and completion of a survey.
UC08	Bill Service	Bill for all services rendered
UC09	Make payment	Record all payments received
UC10	Identify Internet connection fault	Record all internet connection failures
UC11	Restore Internet connection	Record all internet connection restorations
UC12	Power failure	Record all power failures
UC13	Power restored	Record when power is restored
UC14	PC is switched on	Record all instances when the PC is switched on
UC15	Equipment is switched off	Record all instances when the PC is switched off

The use case descriptions are described using the templates together with a use case diagram, explained in Chapter 3. The use cases and scenarios described here represent the most common functions performed at the Telecentres.

Figure 4.5 illustrates the “Request a service” use case “UC03” in standard UML version 2.0 format. The accompanying scenario(s) of a user who requests a service from the administrator, who will log the request in the service request log, are described in table 4.4. Figures 4.11 and 4.12 show a typical screen which would capture and display the service request and allocation of a service. The administrator logs the user profile, involving their age and occupation category, together with the services requested. This use case corresponds to common user profile attributes identified above. In this scenario, UC03SC03, the process can follow one of three possible paths as extension branching steps: viz. firstly the service is usable and the user can perform the task, secondly the task is handed over to the administrator to perform and, thirdly, service is not available. Note that this scenario is the scenario developed after several iterations beyond information obtained at the initial design phase. Hence, it contains information that was obtained at the design phase, such as the option for self-service or administrator-service, and fed back to the appropriate phase for remodeling and incorporation into scenarios.

Table 4.4 Request Service

Category	Description
use case No	UC03
Related Requirements	Request a service
Goal in Context	The user’s request for a service from the administrator is logged.
Scenario	
No	UC03SC03
Name	Request a PC to use an application
Trigger	User arrives at the Telecentre to use an application on a PC.
Main flow action steps	<ol style="list-style-type: none"> 1. A user requests to use a PC for an application. 2. The administrator enters the user profile where the following detail is captured: services requested, occupation, and age category.

	<ol style="list-style-type: none"> 3. The administrator saves the profile in the service request log. 4. The user joins the queue.
Extension branching Steps	<ol style="list-style-type: none"> 3.1 The user asks the administrator to perform the task. 3.2 The user exits. 1.1 The service is not available / unusable. 1.2 The administrator captures a reason for non-usage of service. 1.3 The user is removed from the queue. 1.4 The user leaves.

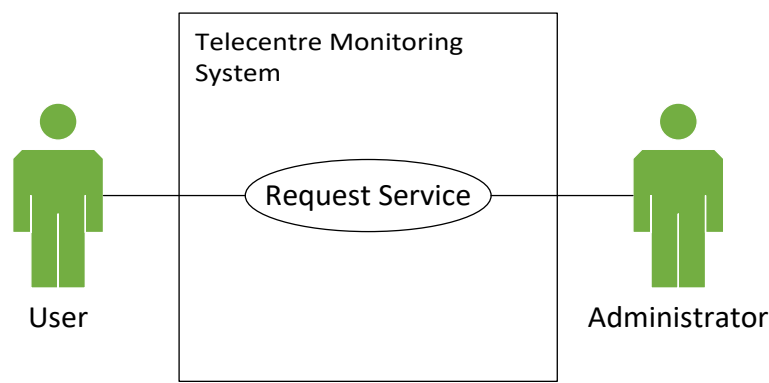


Figure 4.5 Use Case – Request Service

Figure 4.6 illustrates the “Allocate Service” use case “UC04” described in Table 4.5, of a user who is allocated a service by the administrator. This allocation of service to the user is logged by the administrator, together with the user profile, in the service request log. The use case diagram in Figure 4.4 illustrates that one of the actors is the “service request log” as this unit delivers part of the input whilst the remaining actors are the administrator and the user. The scenario “UC04SC02” allows the date and time of allocation, together with the equipment and services, to be logged together with the user profile. This data is used for operational and reporting purposes.

Table 4.5 use case and scenario – Allocate a service

Category	Description
use case No	UC04
Related Requirements	Allocate Service
Goal In Context	A User is allocated a service by the Administrator
Scenarios	
No	UC04SC02
Name	Allocate PC for application usage
Trigger	Availability of service and a user is in the queue for the service
Main flow action steps	<ol style="list-style-type: none"> 1. The User / Administrator is allocated a PC for Application usage. 2. The administrator captures the service allocated and Equipment ID against the user profile.

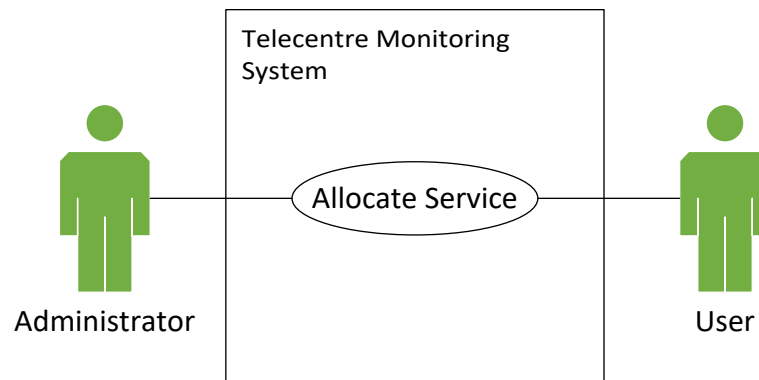


Figure 4.6 User Case – Allocate Service

Figure 4.7 illustrates the “Complete usage of service” use case “UC07” that is described in Table 4.6, of a user who informs the administrator the he has completed usage of a service. In the case of the user utilising a service on a PC, he will close all applications/services, log off and report to the administrator. The user has the option of completing a survey that will be initiated once the log-off process is triggered. If the user does not log off, the administrator will log off the session. If the administrator has performed the activities on behalf of the user, the administrator will notify the user that

the work has been completed. The user will be given the option of completing a survey, and will then hand the work to the administrator.

Table 4.6 use case and scenario – Complete usage of service

Category	Description
Use case no.	UC07
Related Requirements	1. Complete usage of service
Goal in Context	2. Log the completion of usage of a service and completion of a survey.
Scenarios	
No.	UC07SC01
Name	Complete usage of the service
Trigger	Administrator / User has completed usage of the service.
Main flow action steps	<ol style="list-style-type: none"> 1. The Administrator / user closes all applications. 2. The user reports to the Administrator. 3. The Administrator / user completes a survey. 4. The user exits
Extension Branching steps	<ol style="list-style-type: none"> 3.1 The Administrator notifies the user for the completed task. 3.2 The user collects the competed work. 3.3 The user exits.

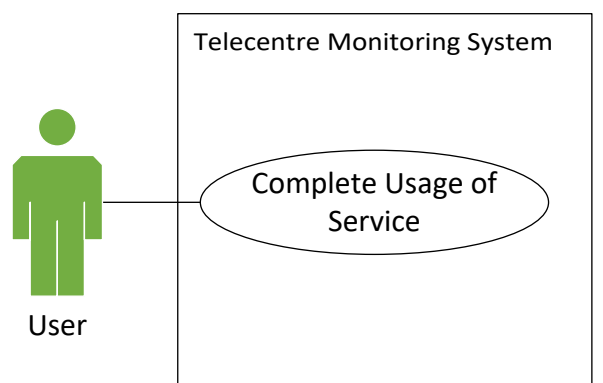


Figure 4.7 User Case – Complete usage of service

Figure 4.7 illustrates the “Bill usage” use case “UC08” that is described in table 4.8, of an administrator who checks the availability of the requested service and bills the user for the service. The cost is calculated, recorded and the receipt is printed. If the user has an account, then the account is debited, otherwise the user is billed for the service usage. The University of South Africa (UNISA), an example of an account customer, has an agreement with some of the Telecentres in South Africa for students to utilise the Telecentres upon which UNISA will settle the account. This feature can be used to accommodate any customer of this type.

Table 4.7 use case and scenario – Bill Usage

Category	Description
Use case no.	UC08
Related Requirements	Bill Service
Goal in Context	Bill for all services rendered
Scenarios	
No.	UC08SC01
Name	Bill usage
Trigger	User is in the queue and service is available.
Main flow action steps	The user requests the administrator for a service. The administrator checks to see if the service is usable. The administrator calculates the cost and records it. The receipt is printed. The user pays for the service requested.
Extension branching Steps	The user positively identifies himself. The cost is billed to the institution responsible for the payment as the account holder e.g. UNISA.

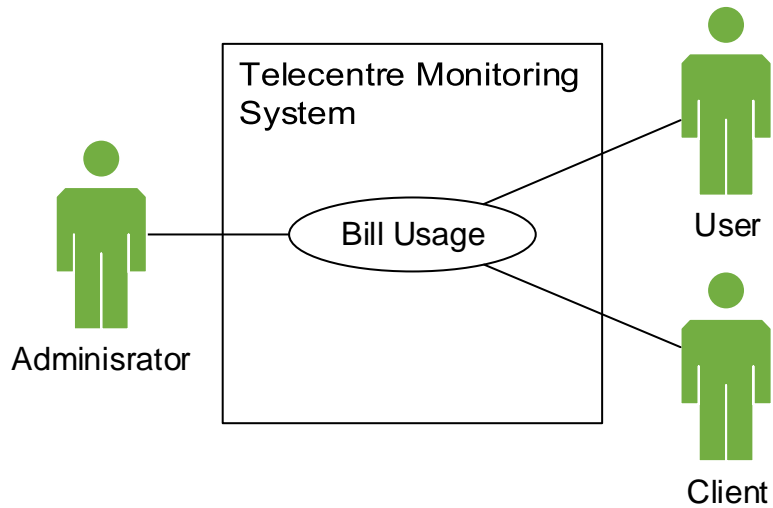


Figure 4.8 Use Case – Bill usage

Figure 4.9 illustrates the “Make Payment” use case “UC09” that is described in Table 4.8, of a user who makes payment to the administrator for the service usage.

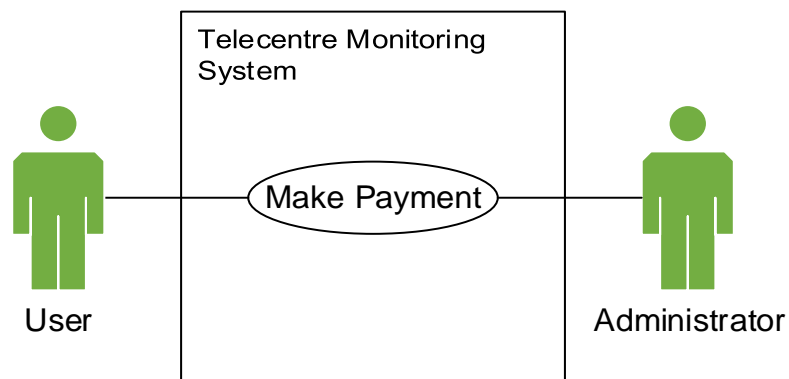


Figure 4.9 Use Case – Make Cash Payment

Table 4.8 use case and scenario – Make Payment

Category	Description
Use case no.	UC09
Related Requirements	Make payment
Goal in Context	Record all payments received

Scenarios	
No.	UC09SC01
Name	Payment Cash for service
Trigger	A user requests for a usable service.
Main flow action steps	The bill is presented to the user The user pays the administrator for the service.
Scenarios	
No.	UC09SC02
Name	Bill Account for service
Trigger	A user requests for a usable service.
Main flow action steps	The amount billed is charged to the customer's account

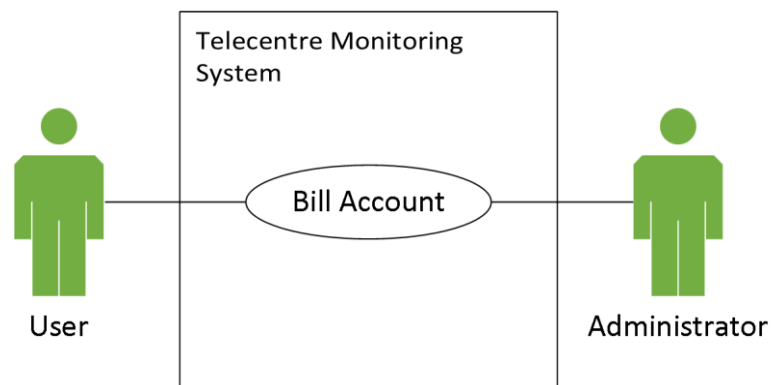


Figure 4.10 Use Case – Bill Account Payment

The use cases and scenarios in Figure 4.5 to Figure 4.10 were confirmed and verified using the expert opinion of the Telecentre managers. A prototype was created and data were captured to show the possible reporting potential, and to further clarify the scenarios and the use cases.

Figures 4.11 and 4.12 show the feature of the prototype that mimics the design functionality of creating a user profile when a user requests a service, and for allocating

a service respectively. This profile consists of a combination of attributes that were initially set up. In order to minimize errors and have consistent data the data capture is validated using combo boxes where applicable. The second half of the screen has a table that allows multiple services to be captured for a user. This functionality was demonstrated to the Telecentre managers in order to obtain feedback and to clarify their requirements.

Request No	Service	Amount Paid	Refund Amount	Date Out	User
------------	---------	-------------	---------------	----------	------

Figure 4.11 Create user profile

Once the services become available, the administrator searches for the users in the queue and allocates the respective equipment to them. This feature is shown in Figure 4.12. Only the available equipment is shown in the list.

The above data was generated so that it could be used by TeleMun in order to demonstrate the typical graphs that could be displayed and the information that could be extracted from this data. The administrator's capture of the user profile and service usage, highlighted in corresponding scenarios identified in the requirements phase, and incorporated into the tool, demonstrated the ease of use for the administrator.

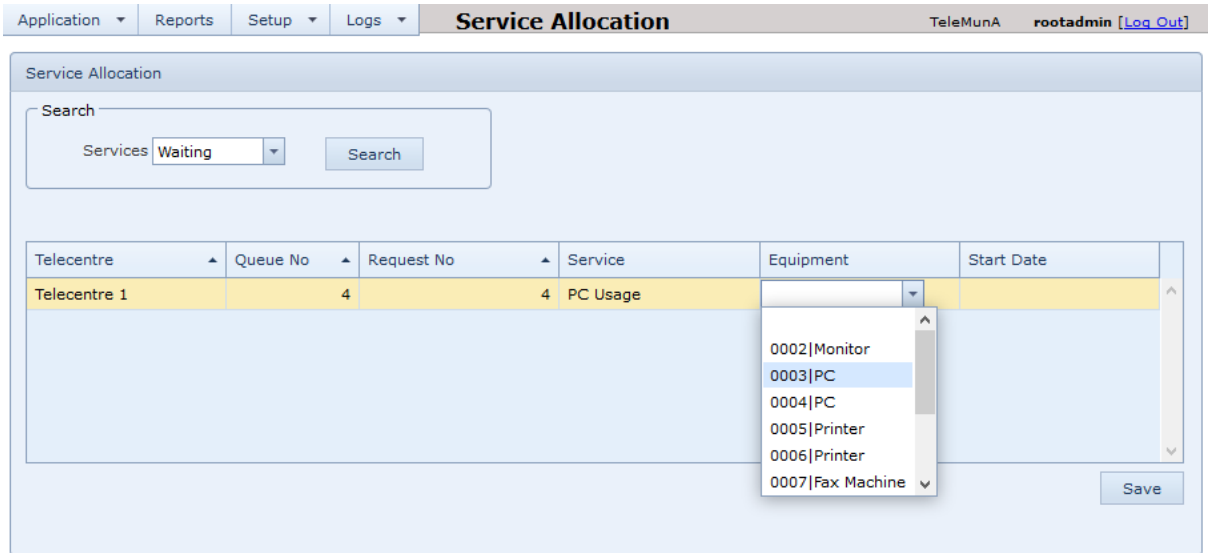


Figure 4.12 Allocate Service

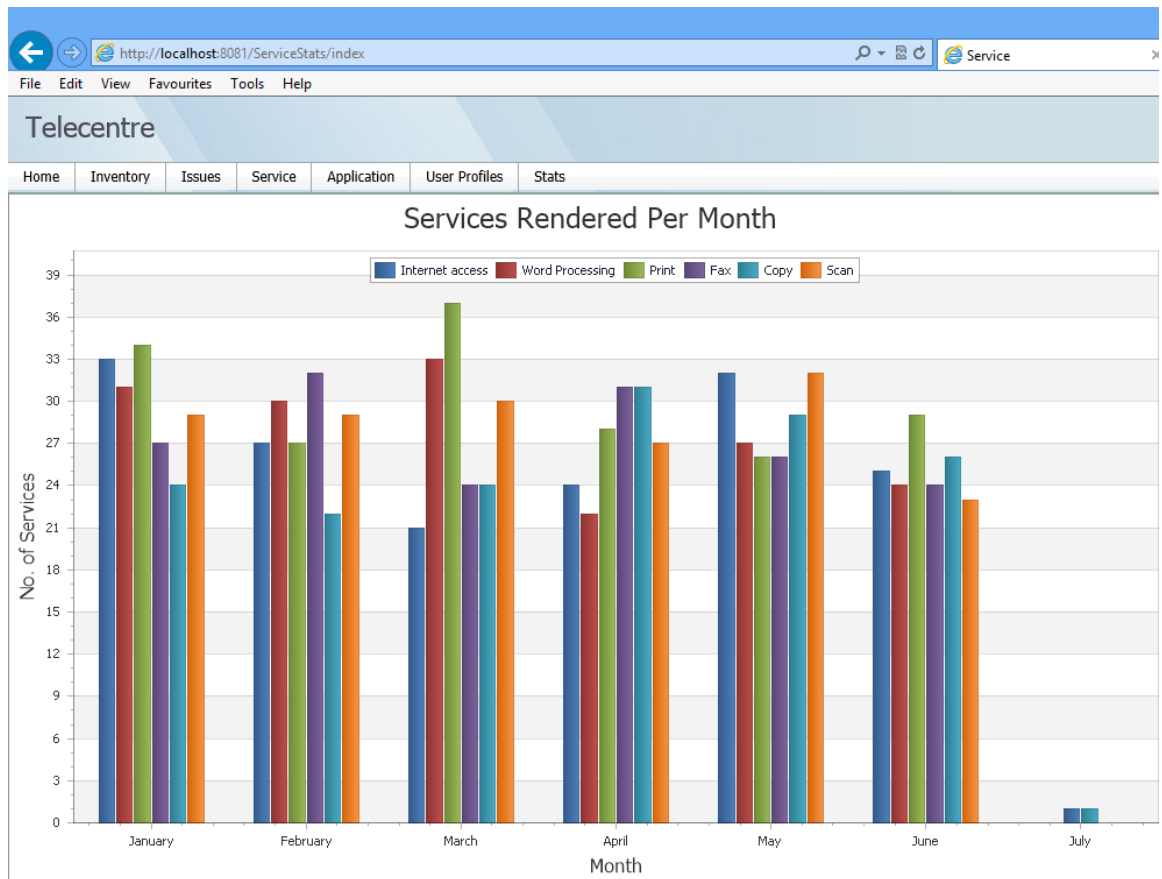


Figure 4.13 Graph of service usage over a 6-month period.

Figure 4.13 shows the different services used over a 6-month period. This demonstrates the reporting potential of the prototype's design functionality from the test data produced.

The area manager and Telecentre managers reported that they were not only impressed that the model met their monitoring requirements, but that they were also impressed with the potential reporting of the model after having captured the necessary user and usage profiles. They were very pleased with the potential outlined in Figure 4.13 which demonstrated the simulation of the usage of different services over a six-month period.

In particular, they said that they appreciated the value of the TeleMun's capability to aggregate data and produce graphs of service usage and total usage. The TeleMun's ability to change the granularity of the reporting data enables it to produce reporting for the different management levels, as well as providing different perspectives on the information. This feedback from the users, in the form of expert opinion, helped refine the model.

4.2.3 Design

Expert opinion and walkthroughs were used to refine the use cases and scenarios that were later transformed into activity diagrams. Initial analysis of the scenarios revealed that there were a number of different services offered by the Telecentres. These scenarios included using an application, using the browser, copying documents, scanning documents and faxing documents, as indicated in Figure 4.1. This figure also shows the initial repeated check for the type of service offered. Although "Use Service" was consolidated during stage 1 of the analysis, two checks for type of service were done: once at the beginning when the user is allocated a service, and then again at the end for billing purposes. The process flow was modified in design so that a single check for the type of service was executed once at the beginning. The services initially considered were consolidated for simplicity and extensibility. The common service is broken down by type of service later on, after monitoring its usage time, due to specific

monitoring needs of service requested. The sponsors of the monitoring system wanted to know the details of the applications used, the sites visited, equipment used and durations of usage of the equipment. This information required a separation at the services level for the monitoring process to deliver the correct detail. Hence, a single use case is used to consolidate services, as indicated by the use case “Allocate Service” as shown in Figure 4.6. This provides for simplicity in overall diagrams, but it can also be expanded to adapt to client needs as shown in Figure 4.14.

Further external scenarios were identified during the walkthrough with domain experts while traversing the process flows of scenarios that were incorporated in the activity diagrams. One of these scenarios missed in the initial analysis was that there was no prescribed method to record and manage unavailability of external services such a power failure. Another use case identified was the refund of usage, after allocation but before its completion, for an event occurring during usage such as internet loss. This resulted in further investigation to incorporate missing scenarios, which necessitated iterations back to earlier stages of the design science model, leading to further refinement to include these new scenarios and processes. Using a similar process, other business activities were also formalized and included in the model.

Another example of a missed scenario, this time at the design phase, was the discovery that some Telecentres offer services on a self-service and / or full-service model. In the case of a full-service option, the user elects to hand over their work to be completed by the Telecentre staff. The user will then be billed for the service. Once the task is completed, the administrator informs the user. Alternatively, the user selects the self-service option where they will request a service and complete it for themselves. Based on the initial requirements of only a self-service model (as documented in Figure 4.1) this new dual service requirement also had to be refined through design science iterations. This process now started with a decision as to self-service or full-service: either the user would personally use the equipment, or she/he would hand over the work to be done to the administrator, and the staff at the Telecentre would complete it. Once services are complete, another check establishes the service as self-service or full-service. The user is then notified.

Certain scenarios could not easily be incorporated into the TeleMun model due to their non-routine nature. Scenarios such as start of day process, status of each computer, acquiring and disposal of equipment were considered separate activities from the routine monitoring process.

The iterative nature of the design science model using expert opinion and walkthrough for verification therefore ensured that the design was complete and consistent. It allowed for multiple revisions until the desired result was obtained.

4.2.4 Final Verification and Validation

The final verification and validation of the TeleMun included a walkthrough with Telecentre managers, prototyping and the use of VeriScene and this combination of methods is shown in section 3.4.4.

Expert opinion in the review of TeleMun ensured that the scenarios in the model designed encapsulated the business context, processes and rules. A final walkthrough with test data for each scenario incorporated within TeleMun confirmed that TeleMun's activities and flow consistently and accurately matched the Telecentre business process flow.

The prototype, with its test data together with the TeleMun design, formed the basis of the dynamic analysis. Using the test data in section 3.4.2.2.1 as prototype inputs, the possible outputs were generated based on the final TeleMun. A sample output is shown in Figure 4.14. This includes the date requested, the date the service was allocated, and the date it was completed – also whether it was self-service or full-service, along with the occupation and age category of the user, the service required, the equipment used and the amount charged. These outputs were compared with the expected outputs to show dynamic verification and partial validation.

Telecentre										
Home	Inventry	Issues	Service	Application	User Profiles	Stats				
#	ID	User Profile	Service	Equipment	Requested	Allocated	Completed	Self Service	Amount	
Edit Delete	1021	9 - Unemployed - Age 21 to 30	Scan	Scanner 01	25-02-2013 09:20 AM	25-02-2013 12:40 PM	25-02-2013 09:33 AM	<input type="checkbox"/>	R 115,00	
Edit Delete	1022	3 - Retired - Age 11 to 20	Fax	Fax 01	21-02-2013 09:28 AM	21-02-2013 04:58 PM	21-02-2013 04:13 AM	<input checked="" type="checkbox"/>	R 10,00	
Edit Delete	1023	2 - Retired - Age 61 +	Scan	Scanner 01	31-03-2013 12:29 PM	31-03-2013 08:43 AM	31-03-2013 11:51 AM	<input type="checkbox"/>	R 135,00	
Edit Delete	1024	2 - Retired - Age 61 +	Copy	Copier 01	28-01-2013 12:29 PM	28-01-2013 06:06 AM	28-01-2013 03:30 PM	<input checked="" type="checkbox"/>	R 180,00	
Edit Delete	1025	6 - Retired - Age 21 to 30	Word Processing	PC 02	25-03-2013 08:37 AM	25-03-2013 09:56 AM	25-03-2013 01:59 PM	<input type="checkbox"/>	R 35,00	
Edit Delete	1026	9 - Unemployed - Age 21 to 30	Copy	Copier 01	12-06-2013 01:33 PM	12-06-2013 07:56 AM	12-06-2013 04:15 PM	<input type="checkbox"/>	R 145,00	
Edit Delete	1027	4 - Student - Age 11 to 20	Scan	Scanner 01	14-05-2013 02:27 PM	14-05-2013 12:10 PM	14-05-2013 09:20 AM	<input type="checkbox"/>	R 125,00	
Edit Delete	1028	10 - Employed - Age 41 to 50	Fax	Fax 01	16-03-2013 12:06 PM	16-03-2013 09:50 AM	16-03-2013 11:34 AM	<input checked="" type="checkbox"/>	R 10,00	
Edit Delete	1029	3 - Retired - Age 11 to 20	Word Processing	PC 02	25-05-2013 02:36 PM	25-05-2013 07:44 AM	25-05-2013 01:00 PM	<input checked="" type="checkbox"/>	R 130,00	
Edit Delete	1030	1 - Employed - Age 51 to 60	Print	PC 03	17-04-2013 10:48 AM	17-04-2013 11:10 AM	17-04-2013 02:11 PM	<input checked="" type="checkbox"/>	R 170,00	
Edit Delete	1031	5 - Employed - Age 61 +	Internet access	PC 01	06-05-2013 10:02 AM	06-05-2013 03:28 PM	06-05-2013 05:31 AM	<input checked="" type="checkbox"/>	R 25,00	
Edit Delete	1032	2 - Retired - Age 61 +	Copy	Copier 01	31-01-2013 11:28 AM	31-01-2013 03:40 PM	31-01-2013 03:06 AM	<input checked="" type="checkbox"/>	R 130,00	
Edit Delete	1033	10 - Employed - Age 41 to 50	Fax	Fax 01	12-02-2013 08:16 AM	12-02-2013 09:35 AM	12-02-2013 01:18 PM	<input checked="" type="checkbox"/>	R 200,00	
Edit Delete	1034	5 - Employed - Age 61 +	Word Processing	PC 02	08-05-2013 02:18 PM	08-05-2013 10:58 AM	08-05-2013 04:23 PM	<input checked="" type="checkbox"/>	R 65,00	
Edit Delete	1035	6 - Retired - Age 21 to 30	Internet access	PC 01	27-03-2013 08:07 AM	27-03-2013 02:36 PM	27-03-2013 06:06 AM	<input type="checkbox"/>	R 160,00	
Edit Delete	1036	4 - Student - Age 11 to 20	Scan	Scanner 01	05-02-2013 03:32 PM	05-02-2013 04:28 PM	05-02-2013 09:24 AM	<input checked="" type="checkbox"/>	R 50,00	
Edit Delete	1037	7 - Learner - Age 31 to 40	Scan	Scanner 01	21-04-2013 04:41 PM	21-04-2013 12:03 PM	21-04-2013 09:54 AM	<input type="checkbox"/>	R 20,00	
Edit Delete	1038	9 - Unemployed - Age 21 to 30	Copy	Copier 01	21-03-2013 01:26 PM	21-03-2013 03:19 PM	21-03-2013 07:21 AM	<input checked="" type="checkbox"/>	R 195,00	
Edit Delete	1039	6 - Retired - Age 21 to 30	Scan	Scanner 01	20-05-2013 04:09 PM	20-05-2013 08:01 AM	20-05-2013 03:16 PM	<input type="checkbox"/>	R 150,00	
Edit Delete	1040	1 - Employed - Age 51 to 60	Print	PC 03	29-01-2013 12:03 PM	29-01-2013 10:45 AM	29-01-2013 04:15 PM	<input checked="" type="checkbox"/>	R 25,00	

[New](#) [Refresh](#)

Page 1 of 51 (1003 items) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) ... [40](#) [50](#) [51](#) [»](#)

Figure 4.14 User and usage Profile

Using dynamic analysis with a prototype and user generated test data, it was determined that the actual outputs of the prototype met the expected outputs as determined by the sample input data. Using dynamic analysis with the prototype, it was determined that the functionalities produced by the prototype matched the business rules incorporated within the Telecentre business processes. Due to the matching of the actual and expected outputs produced from the selected input data, the dynamic analysis of the prototype verified and semi-validated the TeleMun. The TeleMun was semi-validated because the validation process ended at the design phase. In order for full validation the functionality at the implementation phase must match the required functionality at the specification phase. VeriScene was used to ensure design consistency, through a semi-formal notation, between the design phase modelled as activity diagrams and the requirements phase modelled by scenarios. This consistency was ensured by matching the initial scenarios coded using the semiformal notation with the generated scenarios of VeriScene [see section 4.3]. The scenarios generated by VeriScene, which are listed in section 4.3.3 below, matched the scenarios specified manually during the requirements elicitation and reduced during analysis. Using the same type of test data as the prototype, a walkthrough of the activity diagram was conducted to ensure full path coverage of paths and activities in the model

VeriScene indicated that all scenarios that were specified during the analysis were

indeed the complete set of scenarios incorporated within the given activity diagram and ensured that the activity diagrams were correct and consistent. The Telecentre experts, via expert opinion, affirmed that there was a correlation between the business processes of the Telecentre, modelled as activity diagrams, and their initial requirements modelled as scenarios. They further confirmed that there were no missing processes and scenarios ensuring completeness of the activity diagram which forms the TeleMun model.

The methods used in the various phases resulted in the final TeleMun. Prototyping using dynamic analysis with user-given test data semi-validated TeleMun. Expert opinion verified that the TeleMun design met their initial requirements. Walkthrough ensured that there was full path coverage for each of the scenarios. VeriScene demonstrated phase consistency between the design and specification phases. By triangulation of four different methods to verify and partially validate TeleMun, any disadvantage by one method was overcome by the advantages other methods used (Yeasmin and Rahman 2012).

After the verification and validation processes the resulting model is illustrated in Figure 4.15. This begins with a start of day during which all PCs are switched on ready for usage. Once a potential user requests to use one of the services of the Telecentre, the user profile will be logged. If the service is available, the user is allocated the resource to be used. If the service is not available, the user is asked to wait until the service becomes available. The TeleMun includes a flexible billing process allowing for pre- and post- usage as well as being able to handle both cash and account users. Furthermore, the TeleMun is adaptable to outages and / or managing queues. For example, should there be an interruption in service the user is reallocated the service or offered a refund where applicable. The user is asked to complete a survey once usage is completed – but this request is optional.

Due to the application of multiple methods at various stages, errors were corrected which resulted in a model that reflected the actual business processes as stipulated in the requirements. The model terminates with an end of day routine during which the

administrator will switch off all equipment.

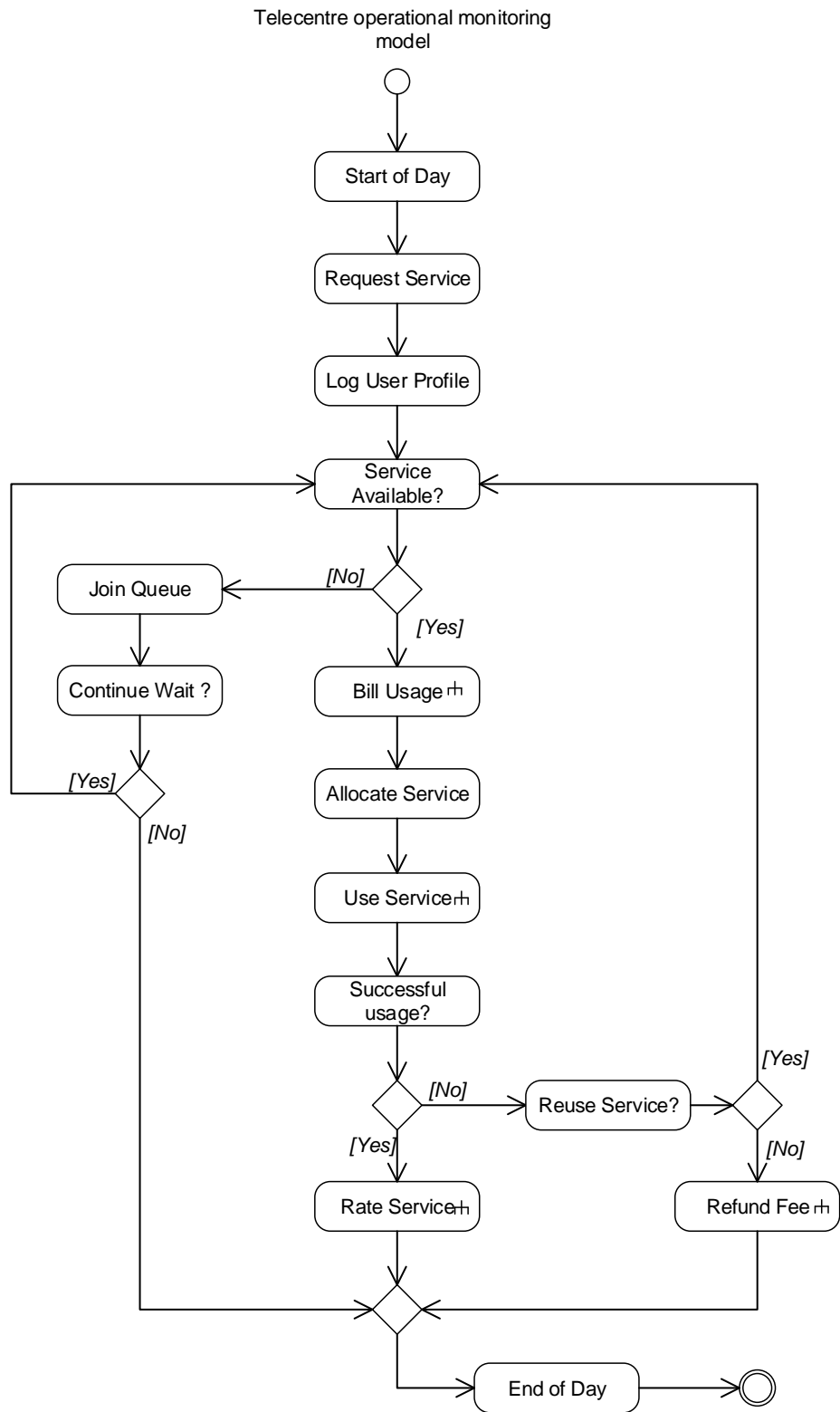


Figure 4.15 TeleMun

Requirements for TeleMun were identified both from the literature and from domain experts. These requirements were notated in scenarios that were then reduced to use cases. At each stage of TeleMun's analysis and design, a combination of methods was used to verify these requirements. An example of a common method used was walkthroughs, which used test data to walk through various diagrams. Using prototyping, dynamic analysis, walkthrough and VeriScene the Telecentre Model was verified and partially validated. As explained above, during the verification and validation phase, a gap in existing verification tools was identified, and subsequently a verification tool VeriScene was designed to address this gap.

4.3 VeriScene

The tool to bridge this gap was developed to ensure consistency between specification and design phases. The tool generated all possible scenarios from a formal notation of an activity diagram.

4.3.1 Requirements Elicitation

As indicated in the literature review, a gap was identified between the requirements and design phases. There was a need for a phase consistency tool (Muskens, Bril and Chaudron 2005) to ensure that the initial scenarios identified during the requirements phase matched those in the detailed design phase. There was a need to incorporate formal notation into the tool to add rigour. This notation was able to encode UML diagrams, both scenarios from the requirement phase, and activity diagrams from the design phase.

4.3.2 Analysis

The processes from the activity diagrams formed the input to VeriScene. The requirement of the tool was to generate the scenarios from the activity diagrams. These processes and paths were coded to enable the activity diagrams traversal. These coded activity diagrams are shown in tables 4.9 to 4.13. VeriScene was required to traverse all possible paths from the start node to the end node. The result of this traversal of all

possible paths was the generated scenarios. During the analysis, each unique path in the activity diagram translated to one use case. Scenarios were denoted in formal notation both for rigour and for easier comparison and generation.

4.3.3 Design

The design science methodology was used to iteratively refine the algorithm to traverse the paths. The depth first search was iteratively modified to accommodate constructs of activity diagrams such as loops, merges and forks, which other search algorithms do not accommodate. During the design phase, a path traversal method was used to do a depth first search. Several iterations were used to correct errors and incorrectly generated scenarios, as explained above.

Tables 4.9 to 4.13 represent the coded versions of the manually specified scenarios from the analysis phase. These scenarios needed to be coded so that they could be compared to the automatically generated scenarios in the evaluation phase. The sequence of actions determined the order in which the actions were coded and each action was paired with the next. Each scenario commenced with a “Start” action and ended with an “End” action. The individual scenarios are determined by the decisions made which identify a unique path – for example in Table 4.9 the decision made is “Successful usage = Yes”.

Table 4.9 Activity 2: Telecentre operation, Scenario 1: Successful usage = Yes

	Action One	Action Two
1	Start	Start of Day
2	Start of Day	Request Service
3	Request Service	Log User Profile
4	Log User Profile	Service Available
5	Service Available	Branch-Service Available
6	Branch-Service Available	Bill Usage
7	Bill Usage	Allocated Service
8	Allocated Service	Use Service
9	Use Service	Successful Usage
10	Successful Usage	Branch-Successful Usage
11	Branch-Successful Usage	Rate Service
12	Rate Service	Merge-Successful Usage
13	Merge-Successful Usage	Merge-Service Available
14	Merge-Service Available	End of Day
15	End of Day	End

Table 4.10 Activity 2: Telecentre operation, Scenario 2: Reuse Service = Yes

	Action One	Action Two
1	Start	Start of Day
2	Start of Day	Request Service
3	Request Service	Log User Profile
4	Log User Profile	Service Available
5	Service Available	Branch-Service Available
6	Branch-Service Available	Bill Usage
7	Bill Usage	Allocated Service
8	Allocated Service	Use Service
9	Use Service	Successful Usage
10	Successful Usage	Branch-Successful Usage
11	Branch-Successful Usage	Reuse Service
12	Reuse Service	Branch-Reuse Service
13	Branch-Reuse Service	Service Available

Table 4.11 Activity 2: Telecentre operation, Scenario 3: Reuse Service = No

	Action One	Action Two
1	Start	Start of Day
2	Start of Day	Request Service
3	Request Service	Log User Profile
4	Log User Profile	Service Available
5	Service Available	Branch-Service Available
6	Branch-Service Available	Bill Usage
7	Bill Usage	Allocated Service
8	Allocated Service	Use Service
9	Use Service	Successful Usage
10	Successful Usage	Branch-Successful Usage
11	Branch-Successful Usage	Reuse Service
12	Reuse Service	Branch-Reuse Service
13	Branch-Reuse Service	Refund Fee
14	Refund Fee	Merge-Successful Usage
15	Merge-Successful Usage	Merge-Service Available
16	Merge-Service Available	End of Day
17	End of Day	End

Table 4.12 Activity 2: Telecentre operation, Scenario 4: Continue Wait = Yes

	Action One	Action Two
1	Start	Start of Day
2	Start of Day	Request Service
3	Request Service	Log User Profile
4	Log User Profile	Service Available
5	Service Available	Branch-Service Available
6	Branch-Service Available	Join Queue
7	Join Queue	Continue Wait
8	Continue Wait	Branch-Continue Wait
9	Branch-Continue Wait	Service Available

Table 4.13 Activity 2: Telecentre operation, Scenario 5: Continue Wait = No

	Action One	Action Two
1	Start	Start of Day
2	Start of Day	Request Service
3	Request Service	Log User Profile
4	Log User Profile	Service Available
5	Service Available	Branch-Service Available
6	Branch-Service Available	Join Queue
7	Join Queue	Continue Wait
8	Continue Wait	Branch-Continue Wait
9	Branch-Continue Wait	Merge-Service Available
10	Merge-Service Available	End of Day
11	End of Day	End

The corresponding activity diagrams were coded using the set of actions and action link rules defined in the methodology. VeriScene was then used to generate all possible scenarios for the given activity diagram. These scenarios are listed in tables 4.14 to 4.18.

Table 4.14 Scenario 1, Successful usage = Yes

	A1	Action One	A2	Action Two
1	1	Start	19	Start of Day
2	19	Start of Day	20	Request Service
3	20	Request Service	21	Log User Profile
4	21	Log User Profile	22	Service Available
5	22	Service Available	3	Branch-Service Available
6	3	Branch-Service Available	23	Bill Usage
7	23	Bill Usage	24	Allocated Service
8	24	Allocated Service	25	Use Service
9	25	Use Service	26	Successful Usage
10	26	Successful Usage	33	Branch-Successful Usage
11	33	Branch-Successful Usage	27	R a t e S e r v i c e
12	27	Rate Service	34	Merge-Successful Usage
13	34	Merge-Successful Usage	4	Merge-Service-Available
14	4	Merge-Service - Available	28	End of Day
15	28	End of Day	2	End

Table 4.15 Scenario 2, Reuse Service = Yes

	A1	Action One	A2	Action Two
1	1	Start	19	Start of Day
2	19	Start of Day	20	Request Service
3	20	Request Service	21	Log User Profile
4	21	Log User Profile	22	Service Available
5	22	Service Available	3	Branch-Service Available
6	3	Branch-Service Available	23	Bill Usage
7	23	Bill Usage	24	Allocated Service
8	24	Allocated Service	25	Use Service
9	25	Use Service	26	Successful Usage
10	26	Successful Usage	33	Branch-Successful Usage
11	33	Branch-Successful Usage	31	Reuse Service
12	31	Reuse Service	41	Branch-Reuse Service
13	41	Branch-Reuse Service	22	Service Available

Table 4.16 Scenario 3, Reuse Service = No

	A1 ID	Action One	A2	Action Two
1	1	Start	19	Start of Day
2	19	Start of Day	20	Request Service
3	20	Request Service	21	Log User Profile
4	21	Log User Profile	22	Service Available
5	22	Service Available	3	Branch-Service Available
6	3	Branch-Service Available	23	Bill Usage
7	23	Bill Usage	24	Allocated Service
8	24	Allocated Service	25	Use Service
9	25	Use Service	26	Successful Usage
10	26	Successful Usage	33	Branch-Successful Usage
11	33	Branch-Successful Usage	31	Reuse Service
12	31	Reuse Service	41	Branch-Reuse Service
13	41	Branch-Reuse Service	32	Refund Fee
14	32	Refund Fee	34	Merge-Successful Usage
15	34	Merge-Successful Usage	4	Merge-Service Available
16	4	Merge-Service Available	28	End of Day
17	28	End of Day	2	End

Table 4.17 Scenario 4, Continue Wait = Yes

	A1	Action One	A2	Action Two
1	1	Start	19	Start of Day
2	19	Start of Day	20	Request Service
3	20	Request Service	21	Log User Profile
4	21	Log User Profile	22	Service Available
5	22	Service Available	3	Branch-Service Available
6	3	Branch-Service Available	29	Join Queue
7	29	Join Queue	30	Continue Wait
8	30	Continue Wait	37	Branch-Continue Wait
9	37	Branch-Continue Wait	22	Service Available

Table 4.18 Scenario 5, Continue Wait = No

	A1	Action One	A2	Action Two
1	1	Start	19	Start of Day
2	19	Start of Day	20	Request Service
3	20	Request Service	21	Log User Profile
4	21	Log User Profile	22	Service Available
5	22	Service Available	3	Branch-Service Available
6	3	Branch-Service Available	29	Join Queue
7	29	Join Queue	30	Continue Wait
8	30	Continue Wait	37	Branch-Continue Wait
9	37	Branch-Continue Wait	4	Merge-Service Available
10	4	Merge-Service Available	28	End of Day
11	28	End of Day	2	End

The designed algorithm was coded using T-SQL stored procedures. A relational database was created to house all the data in MSSql Server. The structure of the database is shown in Appendix F.

4.3.4 Evaluation

The purpose of the evaluation phase was to compare the output of VeriScene with the expected outcomes. In this case, the evaluation phase compared the original scenarios (converted using formal notation) to their corresponding VeriScene generated scenarios for an exact match.

Figure 4.16 shows the initial scenarios numbered 1 to 5 identified from the requirements phase. These five scenarios formed a checklist that was used to walk through the TeleMun. The numbering indicates the respective activities that are executed for the different scenarios. A scenario is a unique single path through activity diagram of TeleMun as defined by (Stevens and Pooley 2006). While traversing the activity diagram and encountering a decision, one scenario will follow one path of the decision whilst another scenario will follow the alternative path of the same decision. In so doing, a unique path through the activity diagram for each scenario is guaranteed. Using a checklist, each scenario is numbered to indicate its unique activities in the diagram. In doing so, the checklist ensures that all relevant activities in each of the scenarios are incorporated within it. Based on this criterion, traversing a path as per a given scenario will use all the relevant processes. By matching traversed activity numbers with relevant activities incorporated within scenarios, the checklist ensures that all activities of a given scenario will incorporate their specified activities

Initially all scenarios begin at a start node with a common path and then, later on, are separated by decisions. Each path of a decision contains one or more scenarios of the initial group and nested decisions further separate the scenarios of the parent decision path. Merges of decision path join the scenario contained with these paths. Certain paths could lead to a previous process forming a loop. Once all processes within a scenario are completed, the scenario terminates at the End Node referred to in Figure 4.16.

Scenario 1 – Successful first time

The following is a sequence of activities during the successful first time usage of a service at the Telecentre. Start > Start of Day > Request Service > Log User Profile > Service Available = Yes > Bill usage > Allocate Service > Use Service > Successful usage = Yes > Rate Service > End of Day > Stop. After a user requests a service that is available, the following steps occur: the user profile is logged; the user is billed; the service is allocated; and the user uses the service. The user successfully completes usage of the service and then rates the service and exits.

Scenario 2 – Service is unavailable and user waits

The following is a sequence of activities that follows when a service is unavailable at the Telecentre. Start > Start of Day > Request Service > Log User Profile > Service Available = No > Join Queue > Continue Wait = Yes. After a user requests a service that is unavailable, the user profile is logged and the user joins a queue. The user is advised when a service becomes available.

Scenario 3 – Service is unavailable and user does not wait

The following is a sequence of activities that follows when a service is unavailable at the Telecentre. Start > Start of Day > Request Service > Log User Profile > Service Available = No > Join Queue > Continue Wait = No. After a user requests a service that is unavailable, the user chooses to exit the Telecentre.

Scenario 4 – Service is available and Successful usage is no – (Use chooses to reuse service after unsuccessful usage)

The following is a sequence of activities that follows when a service is unavailable at the Telecentre. Start > Start of Day > Request Service > Log User Profile > Service Available = Yes > Bill Usage > Allocate Service > Use Service > Successful Usage = No > Reuse Service = Yes. After a user requests a service that is available the user profile is logged and the user is billed, allocated a service and then uses the service. Whilst using the service if it is interrupted and the usage is not complete, the user can choose to reuse the service. When the service is available again, it is offered to the user again. The user is advised when a service becomes available.

Scenario 5 – Service is available, Successful usage is No and Reuse service is No – (User chooses a refund after unsuccessful usage)

The following is the sequence of activities that follows when a service is unavailable at the Telecentre. Start > Start of Day > Request Service > Log User Profile > Service Available = Yes > Bill Usage > Allocate Service > Use Service > Successful Usage = No > Reuse Service = No. After a user requests for a service that is available, the following steps occur: the user profile is logged and the user is billed; the user is

allocated a service; and the user uses the service. Whilst using the service it is interrupted and the usage is not complete and the user chooses to receive a refund. Once the refund is paid, the user exits the system.

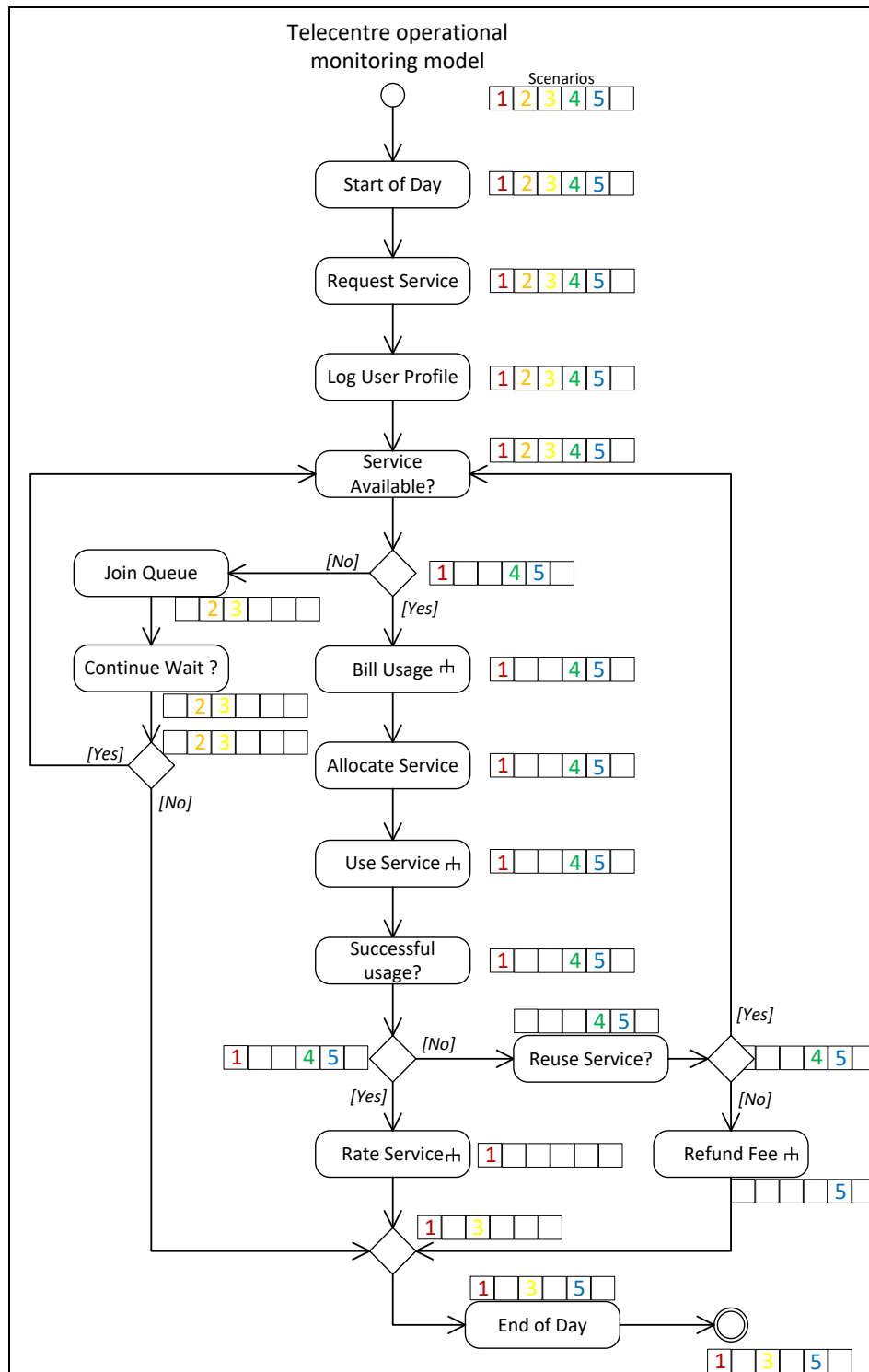


Figure 4.16 Walkthrough of TeleMun with checklist

4.3.5 Communication

Once the evaluation of these sets of scenarios resulted in an exact match, the design was validated. The result consisted of the original coded scenarios with their corresponding generated scenarios.

4.4 Final Results of TeleMun

The managers noted that there were variations in the mode of operation across Telecentres, even for the same activity. It was identified, for instance, that there are different modes of charging. For example, one Telecentre will charge for PC usage which will include printing, word processing and internet usage, whilst another Telecentre will classify and charge for these services separately. Consequently, TeleMun was designed to be adaptable to these different modes of operation. It will log the service usage whether it is charged separately or charged as a combined service. In the case of a combined service the service usages will be logged using monitoring software on the PC. In the case of separate charges, such as PC usage or photocopying, the individual service usages will be logged by the administrator.

The area manager and Telecentre manager recognized the potential for monitoring the Telecentre activities live from a remote site. Activities such as number of people in the queue, service usage and user profiles can all be monitored live from the data captured by TeleMun.

4.5 Summary

The literature review identified the importance of Telecentres and the need for a common monitoring model. Hence a draft TeleMun was developed, verified and partially validated using various industry accepted methods. These methods included walkthrough, expert opinion and prototyping. Walkthrough and expert opinion verified the requirements and the initial design of TeleMun. Prototyping of the model was used to gain first hand feedback from the Telecentre managers on the use of the model and on potential reporting capabilities. At a later stage, prototyping, through dynamic analysis using given test data, was used to partially validate the design. In order to ensure

phase consistency a VeriScene tool was developed and tested. VeriScene re-generated exact scenarios, identified during the analysis phase, from an activity diagram in the design phase, coded using a semi-formal notation. The thorough methodology outlined in chapter 3 was followed to obtain these results.

CHAPTER 5 – Recommendations and Conclusion

5.1 Introduction

In this chapter, the researcher looks at what was accomplished in this research and makes suitable recommendations based on the findings.

5.2 Overview

The literature review indicated that although there is diversity in Telecentre operations, there are common data attributes and processes that can be consolidated for continuous operational monitoring of their activities. An electronic model represented via software design is the most feasible method of instituting this type of monitoring, as it will produce results that are more accurate over a longer period than any other available method. Results from Telecentre monitoring are critical for decision making on the sustainability and success of their operations.

Design science (Peppers *et al.* 2006) was used as the overarching methodology to allow iterative development when requirements elicitation, analysis and design methods indicated new or missing requirements that had to be incorporated into the model. The process incorporated the following stages: derive scenarios and use cases from initial user requirements; reduce and verify these derived scenarios and use cases; transform the use cases into activity diagrams in a design and then verify and validate the design. This design is flexible and incorporates the common Telecentre operational processes. The combination of methods ensured that the disadvantages of one method would be overcome by the advantages of another (Yeasmin and Rahman 2012). At each stage, this combination of methods discovered missing or new requirements that had to be incorporated in the final design (as demonstrated in Chapter 4) in order to ensure that the design was consistent and had a logical flow, as shown in the VeriScene results (see section 4.3.4). See also the letter from the software company that implemented the design into the electronic operational Telecentre monitoring system (Appendix G).

As a spin-off from the combination of identified methods to be used at each stage, a gap was identified in the verification tools (Taentzer 2003) and (Somé 2007) indicating that a simple available tool was needed that would be able to verify phase consistency between the specification and design phases in different domains. In order to address this gap, a tool, VeriScene, was developed that would be able to traverse activity diagrams, regardless of domain, in the design phase, and regenerate scenarios from which it was derived. These regenerated scenarios were compared to the original scenarios to ensure consistency. VeriScene was developed based on a semi-formal notation to ensure rigor and consistency. Each scenario was transformed, using these semi-formal rules (Maiden 1998), from a visual diagram to a formalised notation which aided comparison and enhanced rigor.

Due to the variety of methods for verification and validation and the use of a combination of verification / validation methods at several phases of development, this combinatorial use of methods produced a design model that was very consistent and easily understandable. As a result, the developers implemented the design model without any further clarifications being needed (see Appendix G). Due to the selection of the best practices, and including the use of VeriScene, the development process, which was made possible through the Technology Innovation Agency (TIA) grant, had no requirement inconsistencies. The development process also demonstrated that the scenarios and use cases encompassed in TeleMun provided a full coverage of the selected scope of Telecentre activities. Hence, this demonstration indicated that the practices chosen in this research were appropriate.

5.3 Research Objectives Met

1. Determine whether there are feasible models for electronic operational Telecentre monitoring that are able to gather user and usage profiles as well as monitor internet and power failures.

A review of the literature indicated that there were problems with traditional means of data collection and that these affected the potential for electronic

continuous operational monitoring of Telecentres. The literature also indicated that, despite the commonality of attributes and processes followed by Telecentres, there was no common Telecentre operational monitoring model (either electronic or manual).

2. If no appropriate model exists, to determine appropriate RE methods to capture and verify the requirements for, and to partially validate, the ensuing model. Determine appropriate RE methods for developing a draft common model for automatic monitoring and for capturing a select set of information regarding Telecentre operational activities in South Africa.

It was discovered no common operational monitoring model existed. Consequently, a review of the literature and of best-accepted industry practice led to the selection of an appropriate worldview, methodology, and research methods to develop the TeleMun. The methodology adopted allowed for flexibility for software iterations regardless of the phases. The combination of methods for a single purpose ensured a rigour in the approach through a triangulation of methods such that the disadvantage of one method would be compensated by the advantages of another (Yeasmin and Rahman 2012).

3. Investigate existing, and develop new feasible, methods to iteratively validate the requirements for this draft model and to ensure that this model meets the requirements of

3.1 Consistency

3.2 Path coverage

The literature review indicated that a gap existed in software verification tools able to compare scenarios derived from the requirements and scenarios from the design phase. This gap led to the development of VeriScene to ensure phase consistency between the requirements and the design phases that are commonly specified using the appropriate UML diagrams. The results of VeriScene

ensured that the requirements and design phases were consistent. This consistency by implication indicated that all paths were traversed.

5.4 Recommendations

The common Telecentre model developed in this thesis is suitable for its implementation elsewhere. Because it is a high-level design, it has flexibility to allow for local customised implementations. The identification of a common set of attributes while monitoring this model indicated that these are suitable for providing information generally to Telecentre managers.

The attributes identified from the literature and from the Telecentre requirements were adequate for the purpose of this research. The model can also be extended, allowing for more flexibility to accommodate future changes as the priorities are bound to change over time – for example a decade ago age was an important attribute but currently attributes such as usage and websites accessed, are considered to be more important. Furthermore, the role of Telecentres may change to provide eGovernment services to limit the public travelling long distances at considerable cost. Historically, Telecentres were set up to provide access but now a popular service provided by the Telecentres is training of groups of individuals, including end users, in computing skills. Other opportunities that can be explored by the Telecentres are customised services for entrepreneurship, ecommerce, education, and research, amongst others. The monitoring model can be extended to provide specific statistics on each of these services.

Through the experimental work completed, the combination of methods at each phase of the selected software development methodology produced both a system design and a prototype. The combination and sequence of these methods produced a system design that met the stakeholders' needs and that exhibited consistency from the requirements to the design phases, with no ambiguity. This was attested to by the implementers (see Appendix F). In addition, these combinations and sequences of methods can be incorporated into a model for verification and partial validation of system design within

a software development methodology. Similarly, combination and sequence of methods used for the design of VeriScene can be incorporated into a model for use in the prototyping development domain.

Since the VeriScene tool was successfully tested using different domains, as well as the Telecentre domain, it can be utilised in activity diagrams of other software development domains using similar constructs.

5.5 Future Work

At the time of writing of this thesis, TeleMun was developed fully and was implemented at a selected site with plans to implement it at further selected pilot sites. TeleMun, being an electronic mode, produces continuous real time data. This data could be used to support management, researchers and sponsors in their decision making. Furthermore, the reporting capability of this tool could be enhanced to better visualise the data.

The aggregation of such a large volume of data opens the possibility of various big data mining approaches, which could provide more useful insights than TeleMun's reporting capability currently provides.

The VeriScene tool can be further enhanced to accommodate a graphical web user interface for better usability. Activity diagrams are generally constructed at different levels of complexity. The VeriScene tool can be enhanced to check for consistency in multi-level activity diagrams.

5.6 Summary

In this chapter, an overview of what was accomplished in the research, as well as recommendations and areas for future work within the scope of this thesis are discussed. The methodology and methods chosen demonstrated that the design produced could be easily implemented. The original objectives were evaluated against

the results to ensure that these objectives were met. The chapter highlights the further possible enhancements for TeleMun and VeriScene.

Appendix A – Researcher’s Attributes

Table A1 - Researchers on Telecentres

No	Author
1	(Hudson 2001)
2	(Rajapakse 2012)
3	(Cheuk, Atang and Lo 2012)
4	(Abdulwahab and Dahalin 2012)
5	(Naik, Joshi and Basavaraj 2012)
6	(Cheang and Lee 2010)
7	(Hassan <i>et al.</i> 2010)
8	(Razak, Hassan and Din 2010)
9	(Gomez 2012)
10	(Alasow, Udomsade and Niyamangkoon 2010)
11	(Gomez, Pather and Dosono 2012)
12	(Lashgarara, Karimi and Mirdamadi 2012)

Table A2 contains a list of researchers (listed by the respective number in Table A1) together with the attributes used by researchers in their studies. The columns labelled 1 to 12 correspond to the authors in Table A1. The ‘Y’ in the corresponding cell indicated whether the author used the attribute in their research.

Table A2 - Attributes on Telecentres

No	Attribute	1	2	3	4	5	6	7	8	9	10	11	12
1	Reliability	Y											
2	Technical Assistance	Y											
3	Target group	Y											
4	Gender	Y	Y	Y		Y	Y	Y		Y	Y	Y	Y
5	Suggestions	Y											
6	Application usage	Y							Y				
7	Issues in app usage	Y											
8	Usage frequency	Y											
9	Frequency of users	Y	Y				Y	Y			Y	Y	Y
10	Services offered		Y				Y				Y	Y	Y
11	Service usage		Y		Y	Y	Y	Y			Y	Y	Y
12	Internet URL Usage		Y		Y								
13	Age		Y	Y			Y	Y		Y	Y	Y	Y
14	Income			Y		Y		Y		Y			
15	Qualification			Y				Y		Y			
16	Occupation		Y	Y			Y		Y		Y	Y	Y
17	Location to Telecentre			Y	Y		Y			Y			
18	Expenses					Y							
19	Resources		Y				Y	Y			Y	Y	Y
20	No of Visitors		Y				Y				Y	Y	Y

Appendix B - Activity Diagrams

The Telecentre operations commence with a start of day routine by switching each piece of equipment on.

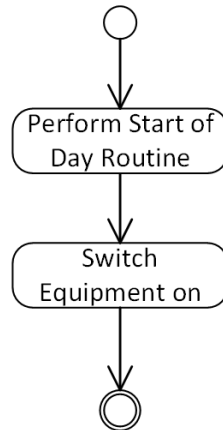


Figure B1 – Start of day routine

The user type is determined prior to usage of the service to determine whether the user is a cash or an account user. If the user type is “Cash”, the use is billed and the payment is collected, and if the use type is “Account”, the respective account is billed.

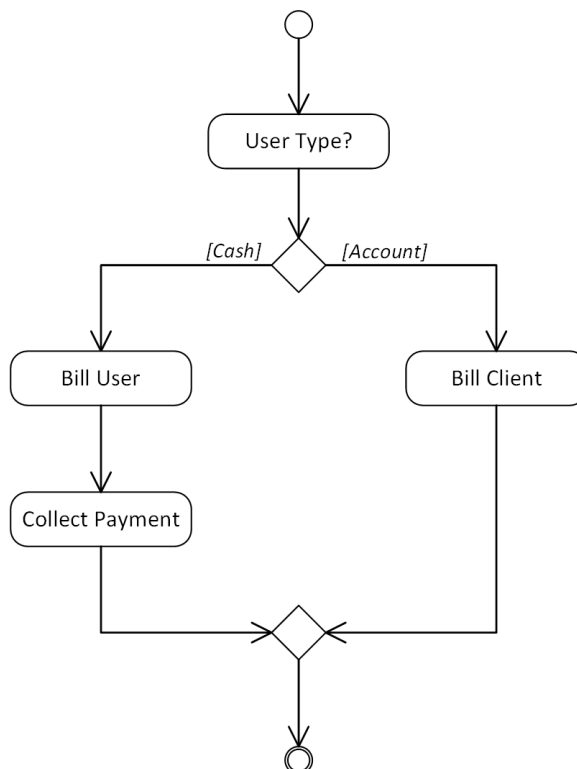


Figure B2 – Billing and payment

During the initial iterations, the processes were duplicated after a decision on “Self Service”. Thereafter the processes were reorganised to execute on decisions made once at the beginning.

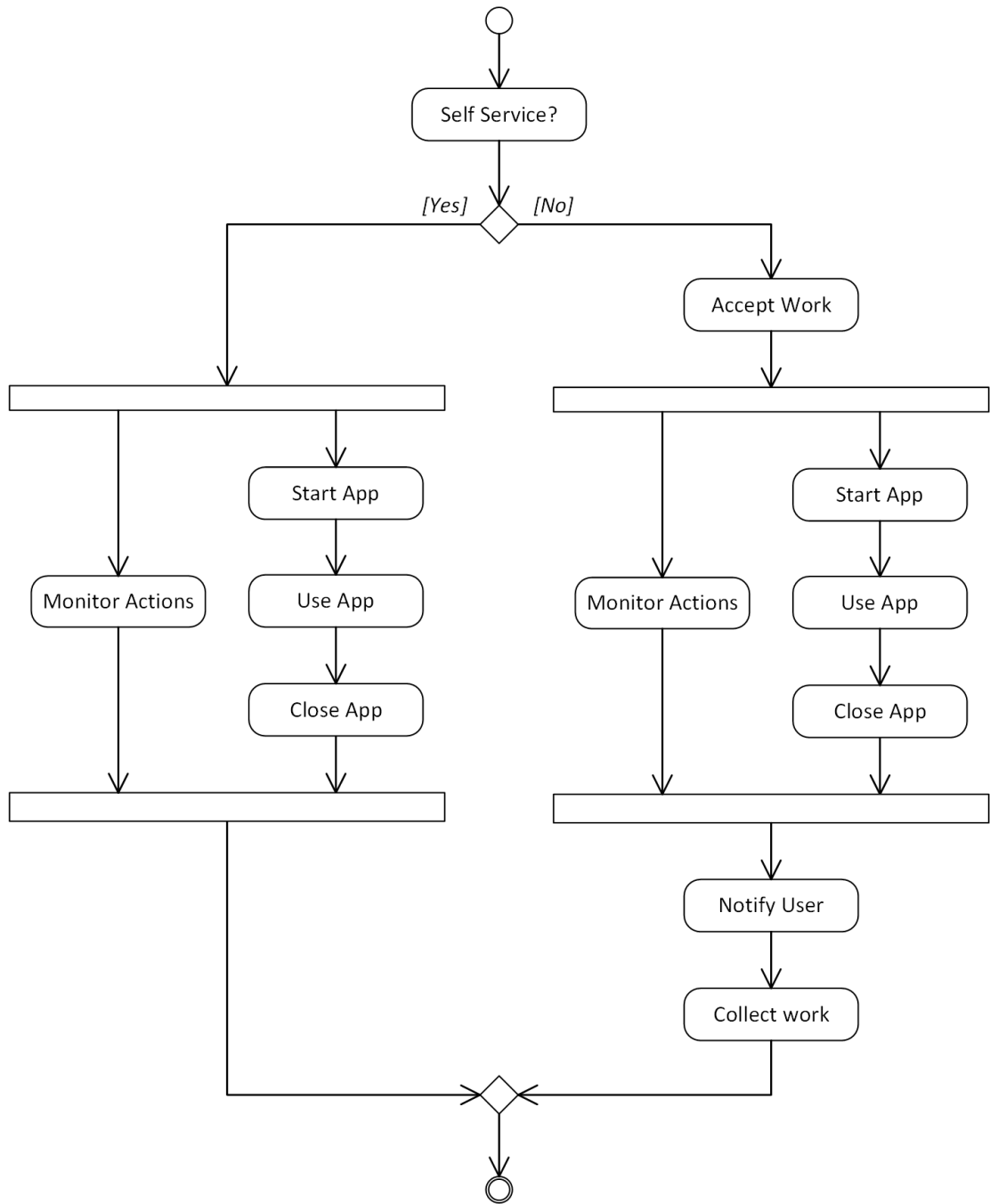


Figure B3 – Final use service process

In cases where the usage of the service is interrupted or not completed for any reason, the user may request for a refund. In the case where a user has paid “Cash”, the user is refunded and in the case where the user type is “Account”, the respective account is credited.

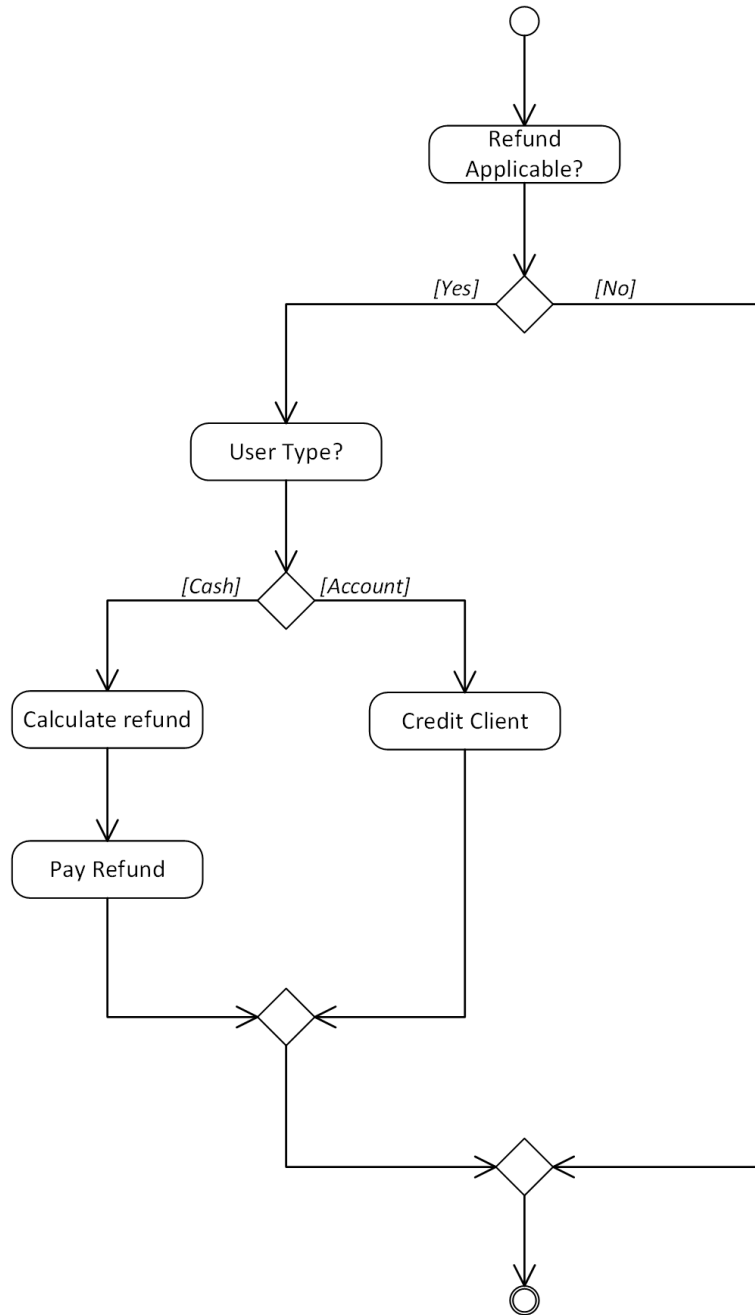


Figure B4 – Refund fee

At the end of each day, the administrator will ensure that each piece of equipment is switched off.

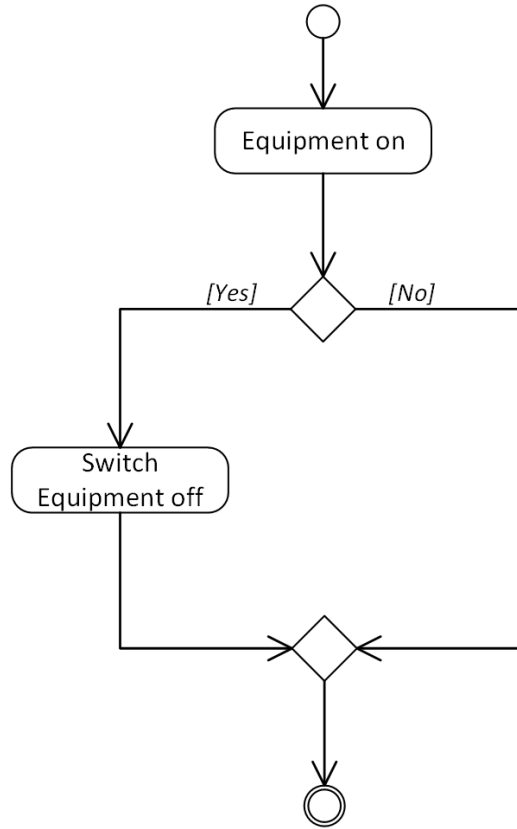


Figure B5 – End of day routine

Appendix C – Testing VeriScene with Different Domains

The following activity diagram shows different domains used to test the VeriScene Tool. This process entails an issue management system where an issue is recorded and resolved at the different stages.

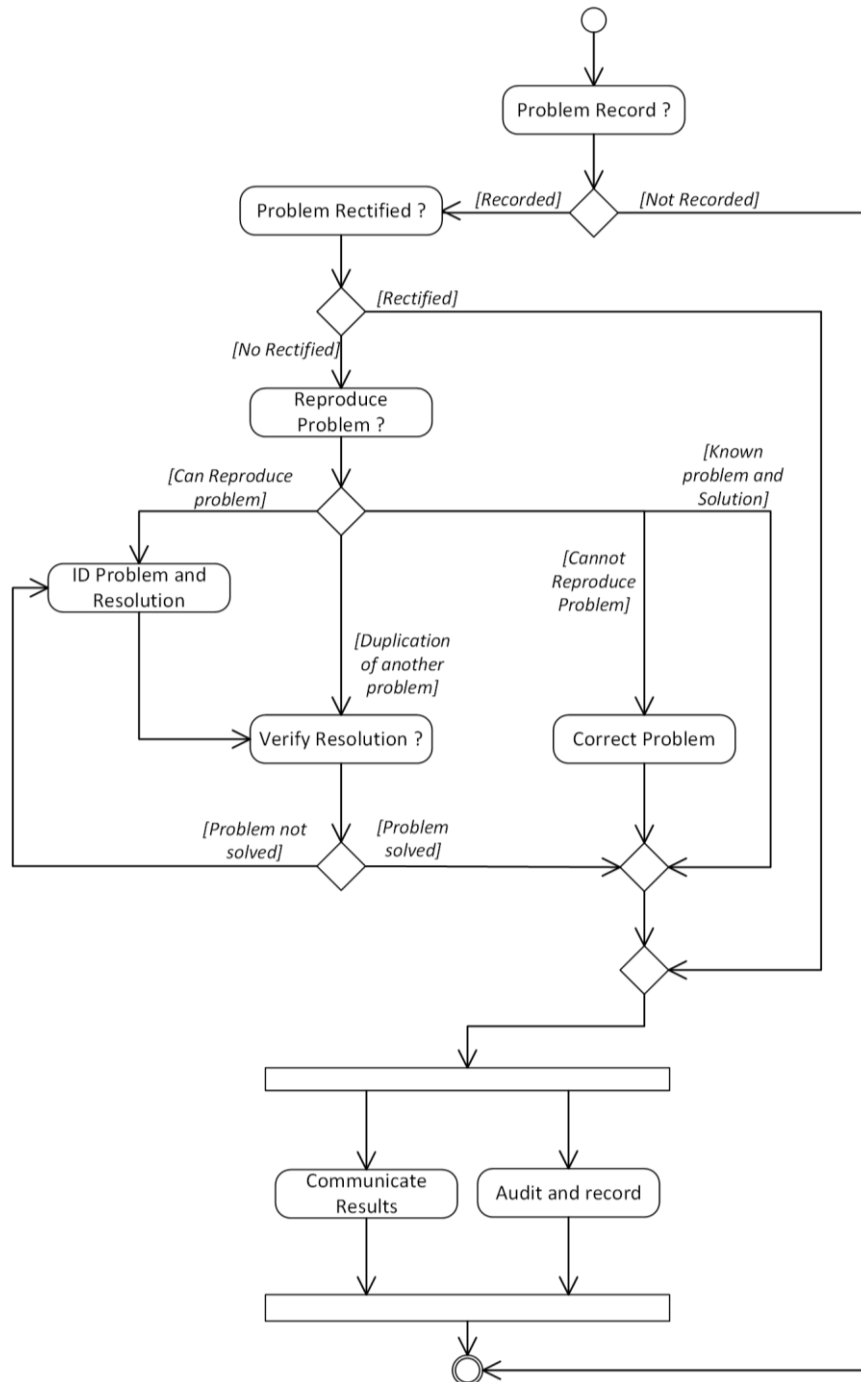


Figure C1 – Trouble Ticket

The activity diagram in Figure C2 represents an order processing system from initial reception of the order to shipping the goods. It entails decisions and concurrent processes.

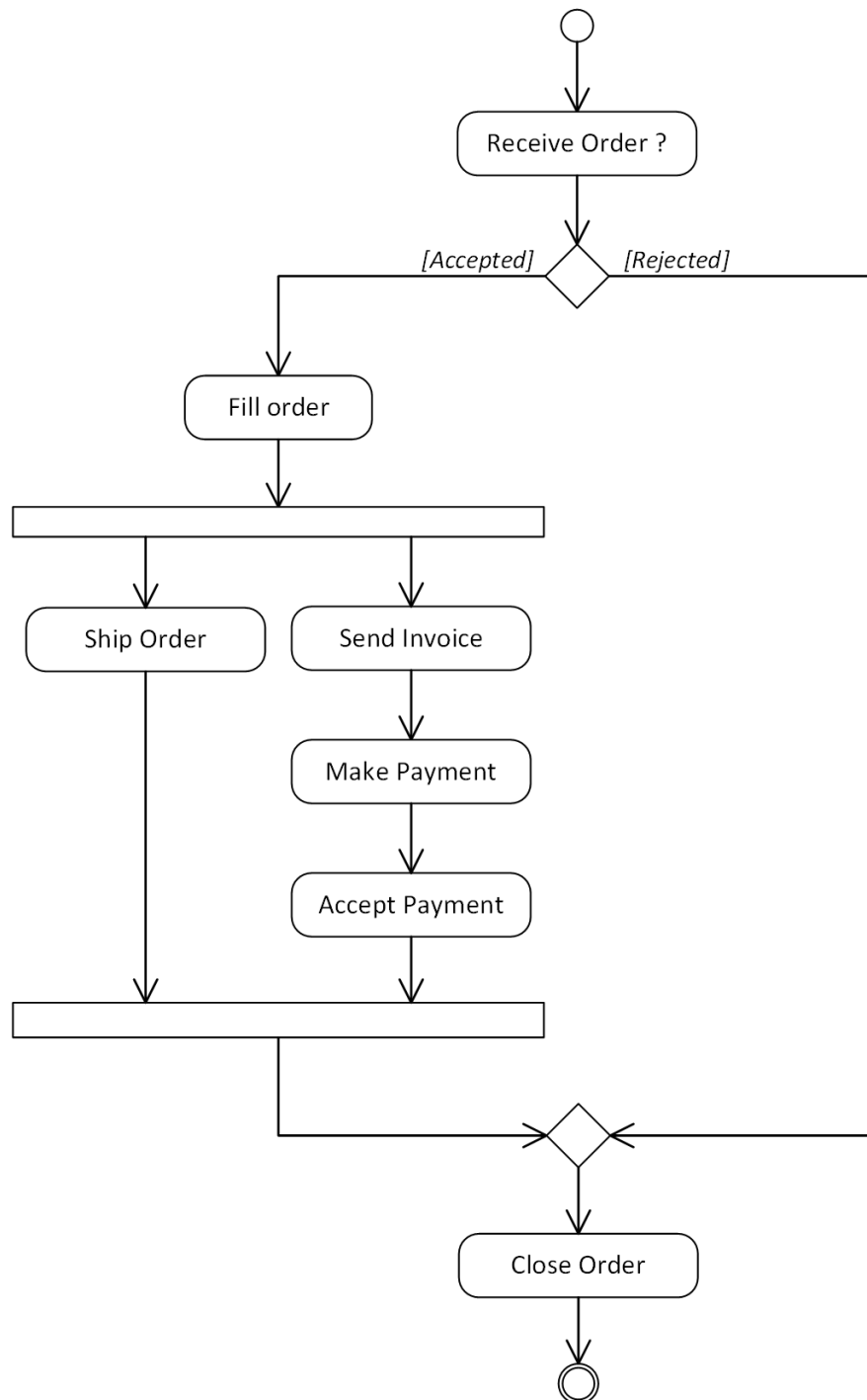


Figure C2 – Order processing

Appendix D – Initial Models

Figure D1 shows common similar processes being followed in the internet café model. Based on previous research through personal interaction with internet cafes the following process model was developed. Some of the similarities between internet cafes and Telecentres included request for services, use service etc. while some of the differences included place cash in till, end of day cash up, etc. The restrictions on payment process and flow is awkward and hard to read and understand in this model.

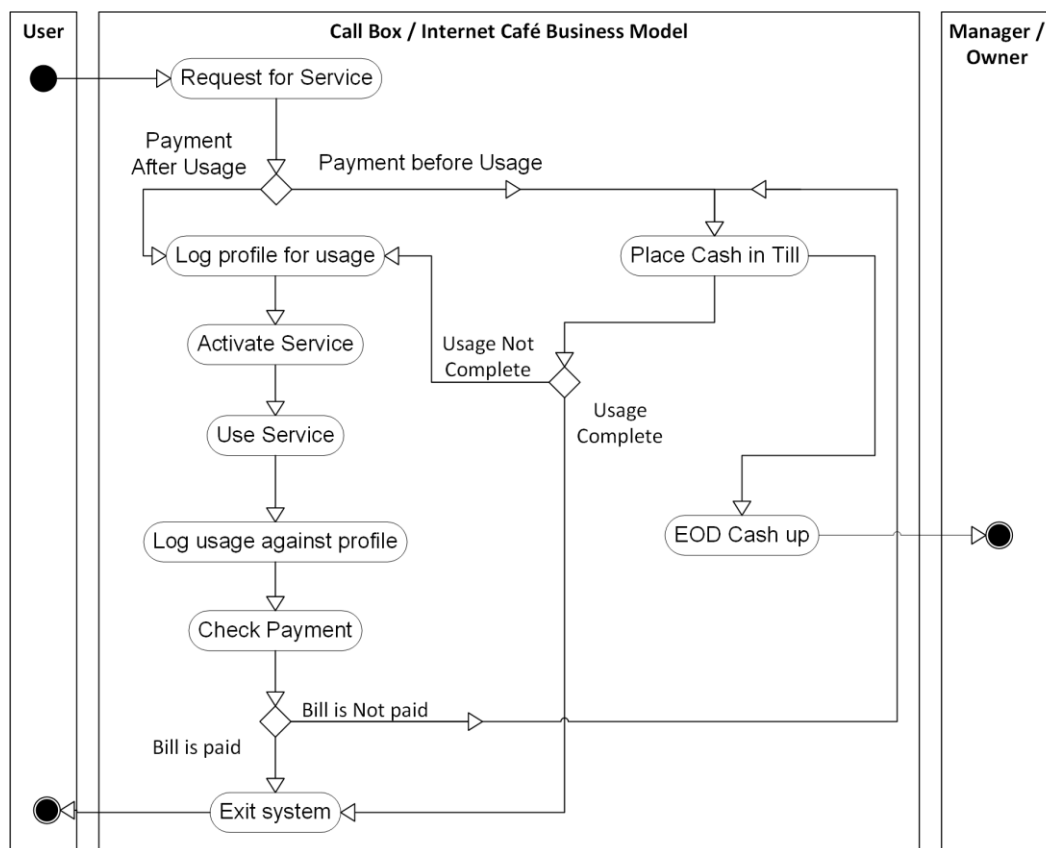


Figure D1 – Call box and internet café business model

The business process of call boxes and internet cafes have similar processes but offer limited services. These processes formed the basis for the initial requirements for a Telecentre operation.

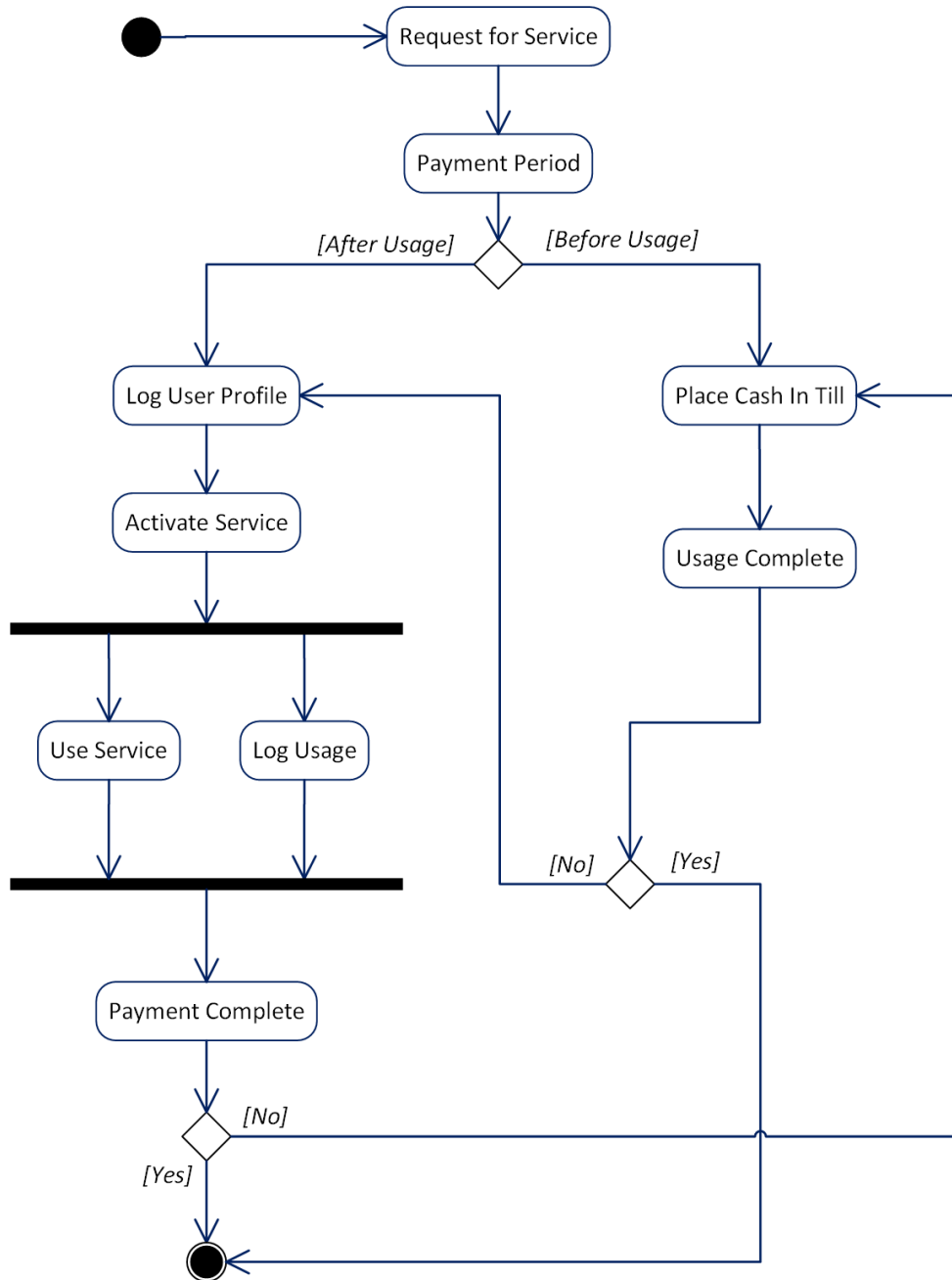


Figure D2 – Telecentre business model

Figure D3 indicates some of the improvements envisaged in a future model. These include recording of faulty equipment and software for statistics purposes.

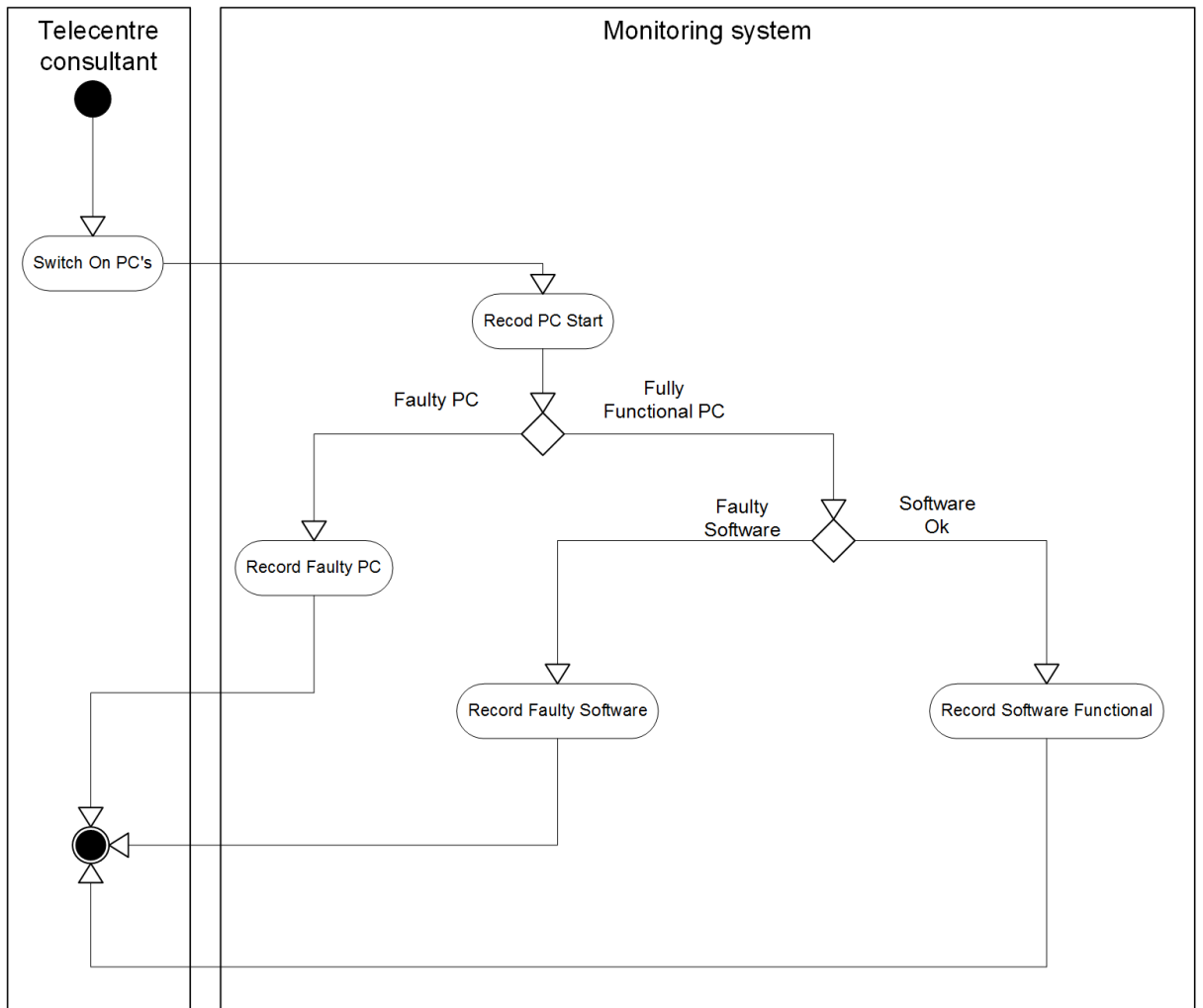


Figure D3 – Monitoring model

Appendix E – Use Case Diagrams

Initial use cases provided for different actors to initiate and receive services. These were consolidated to a single user. The user of the data was also consolidated into one user. Consequently, it became inclusive of different actors. TeleMun focuses on monitoring usage and user profiles as opposed to equipment and reporting although these three sub systems are interlinked.

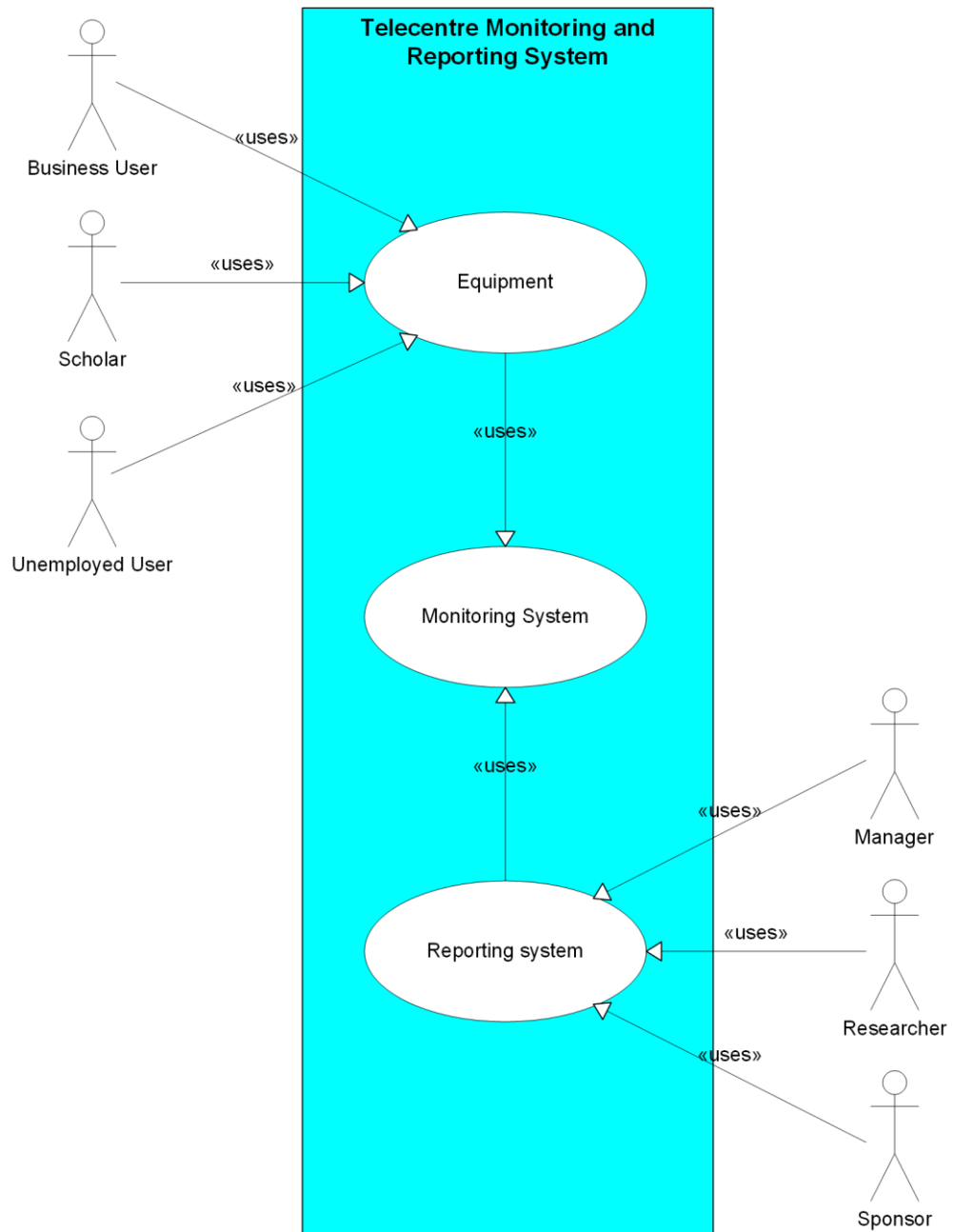


Figure E1 – Telecentre monitoring and reporting system

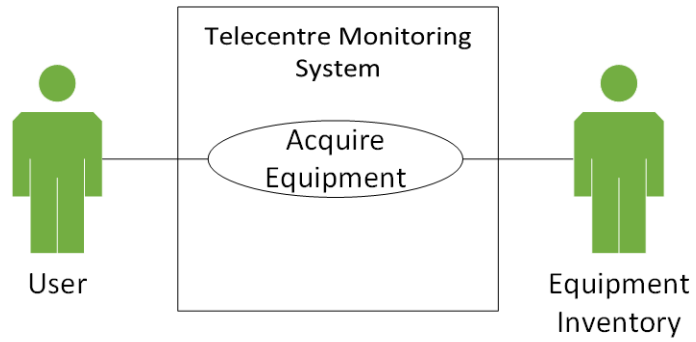


Figure E2 – Acquire equipment

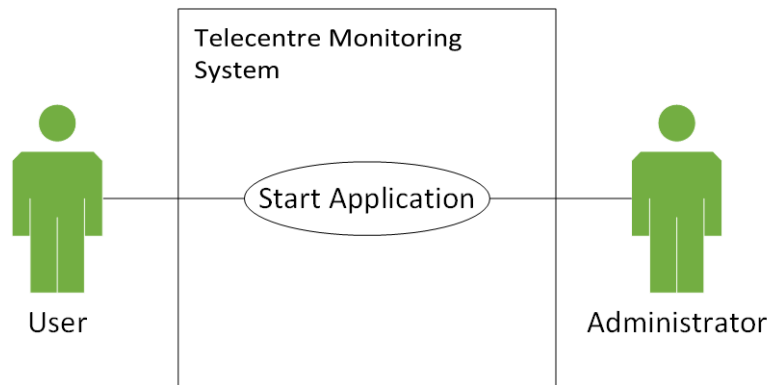


Figure E3 – Start Application

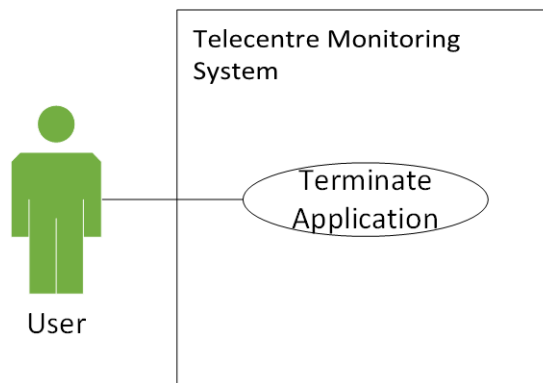


Figure E4 – Terminate Application

Initially the service was classified as 'surf internet'. The final model has an activity, which includes usage of application / services from the internet.

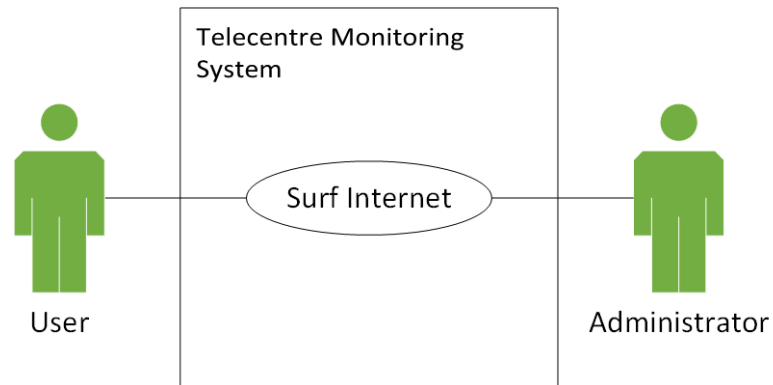


Figure E5 – Surf Internet

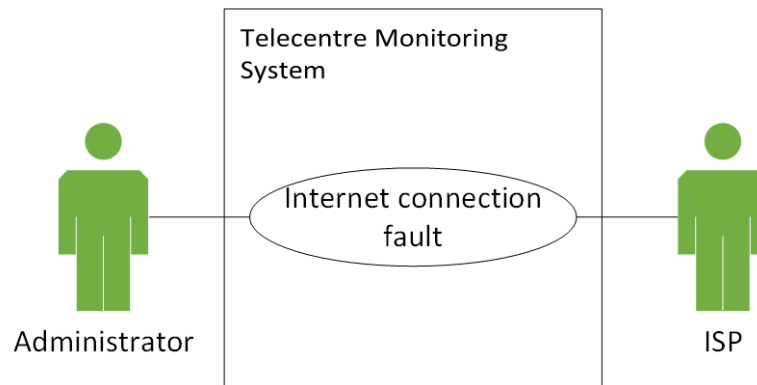


Figure E6 – Internet connection fault

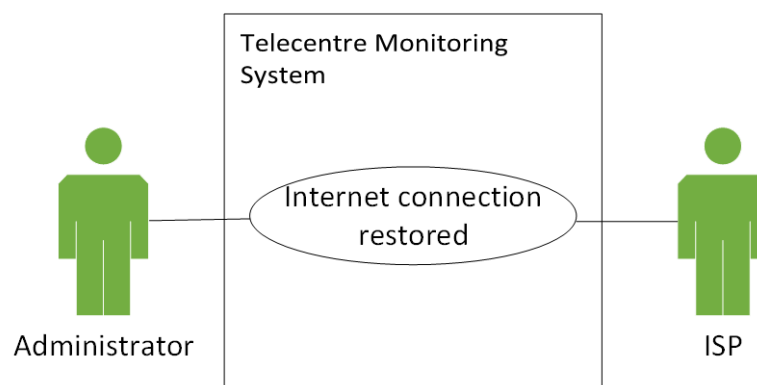


Figure E7 – Internet connection restored

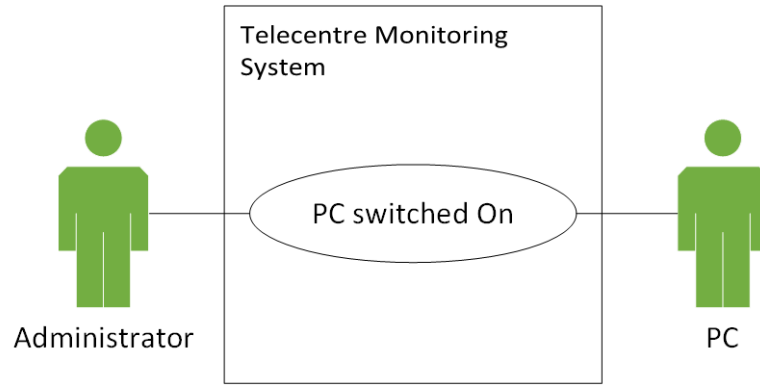


Figure E8 – PC switched on

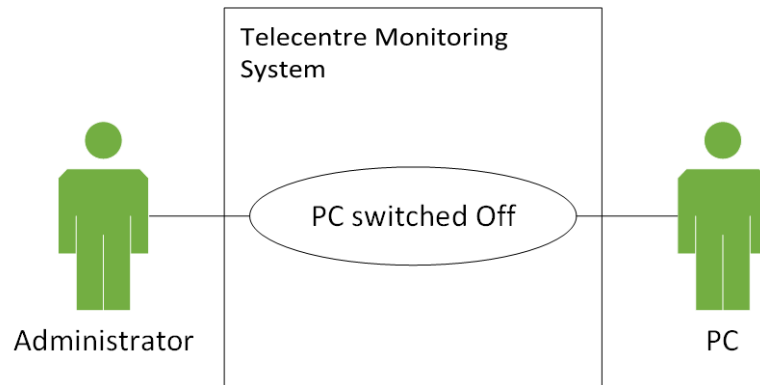


Figure E9 – PC switched off

Appendix F – Database Structure

Table F1 – Database Attributes

Table Name	Column Name	Data Type
AgeCategory	Id	int
AgeCategory	Code	varchar
AgeCategory	Name	varchar
EquipmentCategory	Id	int
EquipmentCategory	Code	varchar
EquipmentCategory	Name	varchar
EquipmentManagement	Id	int
EquipmentManagement	DateAcquired	Datetime
EquipmentManagement	SerialNo	varchar
EquipmentManagement	EquipmentCategoryId	int
EquipmentManagement	Description	varchar
EquipmentManagement	Make	varchar
EquipmentManagement	Model	varchar
EquipmentManagement	MACAddress	varchar
EquipmentManagement	EquipmentTag	varchar
EquipmentManagement	Active	bit
EquipmentManagement	DateDisposed	Datetime
EquipmentManagement	ReasonforDisposal	varchar
EquipmentManagement	Comments	varchar
EquipmentManagement	TeleCentroid	int
EquipmentManagement	TrackingDeviceId	int
InternetLoss	Id	int
InternetLoss	TeleCentroid	int
InternetLoss	DateLost	Datetime
InternetLoss	DateRestored	Datetime
InternetLoss	Comments	varchar
Occupation	Id	int
Occupation	Code	varchar
Occupation	Name	varchar
PowerFailures	Id	int
PowerFailures	TeleCentroid	int
PowerFailures	DateLost	Datetime
PowerFailures	Date Restored	Datetime
PowerFailures	Comments	varchar
Province	Id	int
Province	Name	varchar
QueueStatus	Id	Tinyint
QueueStatus	Name	Varchar

Table Name	Column Name	Data Type
ServiceAllocation	Id	int
ServiceAllocation	TeleCentrelId	int
ServiceAllocation	EquipmentManagementId	int
ServiceAllocation	ServiceRequestId	int
ServiceAllocation	StartDate	Datetime
ServiceAllocation	EndDate	Datetime
ServiceQueue	Id	int
ServiceQueue	TeleCentrelId	int
ServiceQueue	QueueNo	int
ServiceQueue	QueueDate	Datetime
ServiceQueue	OccupationId	int
ServiceQueue	AgeCategoryId	int
ServiceQueue	TeleCentreDistanceId	int
ServiceQueue	ContactNo	varchar
ServiceQueue	StudentNo	varchar
ServiceQueue	Self-Service	bit
ServiceQueue	DateOut	Datetime
ServiceQueue	UserId	Smallint
ServiceQueue	QueueStatusId	Tinyint
ServiceRequest	Id	int
ServiceRequest	ServiceQueueId	int
ServiceRequest	ServiceRequestNo	int
ServiceRequest	TeleServicesId	int
ServiceRequest	AmountPaid	decimal
ServiceRequest	DateOut	Datetime
ServiceRequest	RefundAmount	decimal
ServiceRequest	Comments	varchar
ServiceRequest	UserId	Smallint
SIMCards	Id	int
SIMCards	SerialNumber	varchar
SIMCards	PhoneNumber	varchar
SIMCards	PIN	varchar
SIMCards	PUK	varchar
Suburb	Id	int
Suburb	Name	varchar
Suburb	ProvinceId	int
SurveyQuestions	Id	int
SurveyQuestions	Question	varchar
SurveyQuestions	Active	bit
SurveyQuestions	List Order	Smallint

Table Name	Column Name	Data Type
TeleCentreDistance	Id	int
TeleCentreDistance	Code	varchar
TeleCentreDistance	Name	varchar
TeleCentres	Id	int
TeleCentres	Name	varchar
TeleCentres	ProvinceId	int
TeleCentres	SuburbId	int
TeleCentres	LineAddress	varchar
TeleCentreUsers	TeleCentreId	int
TeleCentreUsers	TeleCentreUserId	Smallint
TeleServices	Id	int
TeleServices	Code	varchar
TeleServices	Name	varchar
TrackingDevices	Id	int
TrackingDevices	GPSTag	varchar
TrackingDevices	SIMCardId	int
UserSurvey	Id	int
UserSurvey	SurveyDate	Datetime
UserSurvey	TeleCentreId	int
UserSurvey	Comments	varchar
UserSurveyDetails	Id	int
UserSurveyDetails	UserSurveyId	int
UserSurveyDetails	SurveyQuestionsId	int
UserSurveyDetails	SurveyRating	varchar

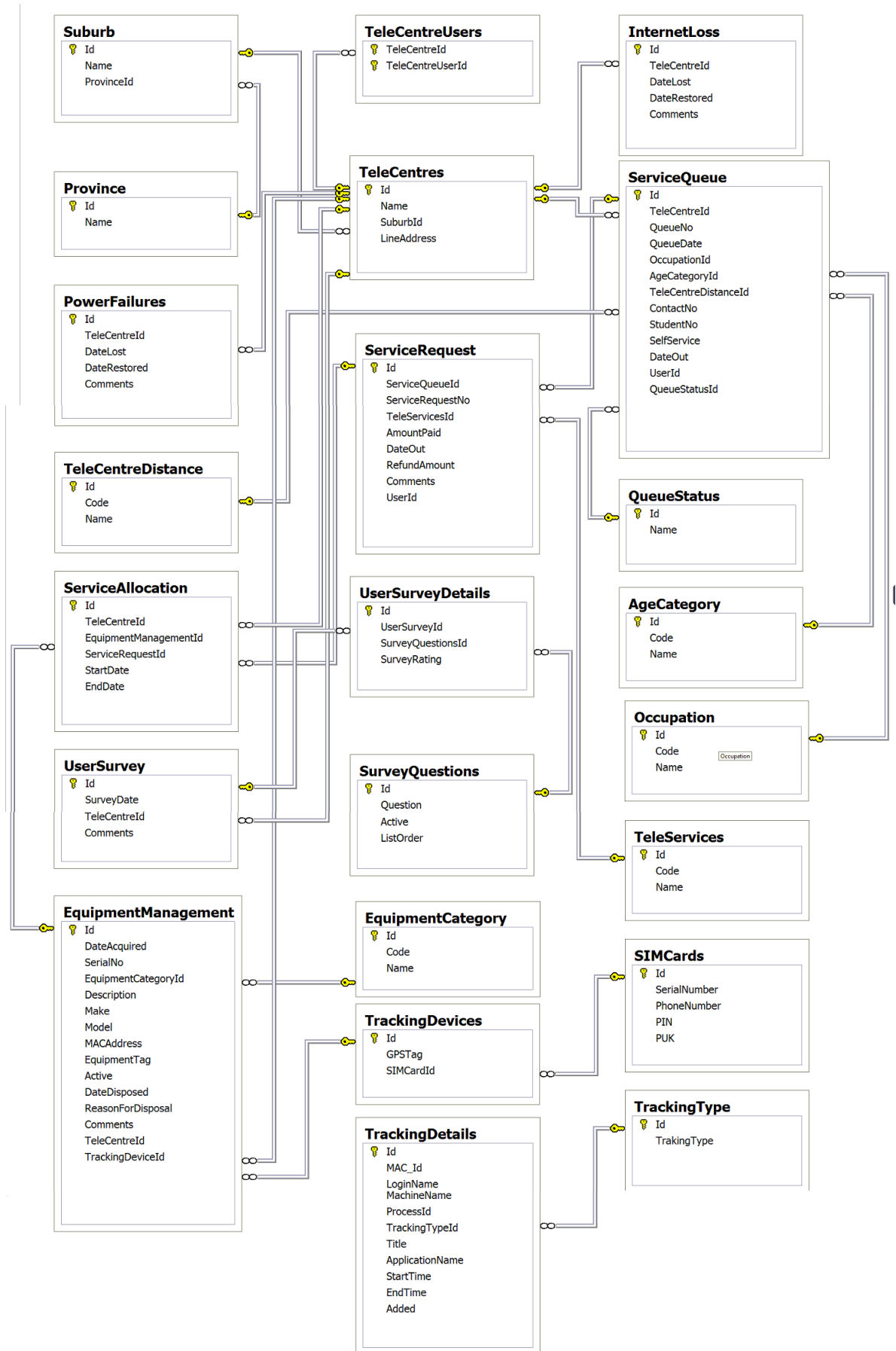


Figure F2 – TeleMun ERD

Appendix G – Letter from Media Platform



Project Telecentre Monitoring and Asset Tracking System

Date: 02/03/2016

Re: Telecentre Monitoring and Asset Tracking System Project

Dear Professor Richard Millham and Jay Pancham

The initial detailed design document was sent on 26/02/2016. This included:

1. Design of the four modules (Monitoring, Asset Tracking, Service Management and Reporting) within the project.
2. All use cases with a list of scenarios per use case.
3. Sample screens for the setup and operational features of the system.
4. Sample report to display the usage of services over a time period.

All requirements, use cases and scenarios that were supplied in the initial design document were well defined, without any inconsistencies or gaps, hence enabling the project team to complete a comprehensive detailed design document. Hence, there was no need for clarification of any of these requirements by the development team.

It was quick to analyse and design the solution based on the requirements received.

Sincerely



Varsha Ramjugath
Project Manager

References

- Abdulwahab, L. and Dahalin, Z. M. 2012. Assessing the Catalyst and the Barriers to Rural Community Based Telecentre Usage. *Journal of Emerging Trends in Computing and Information Sciences*, 3 (6): 826-832.
- Achimugu, P., Oluwagbemi, O., Oluwaranti, A. and Afolabi, B. 2009. Adoption of information and communication technologies in developing countries: an impact analysis. *Journal of Information Technology Impact*, 9 (1): 37-46.
- Aggarwal, M. and Sabharwal, S. 2012. Test case generation from UML state machine diagram: A survey. In: Proceedings of *Third International Conference on Computer and Communication Technology (ICCCT)*. Piscataway, NJ, USA: IEEE, 133-140. Available: <http://ieeexplore.ieee.org/ielx5/6387875/6394656/06394682.pdf?tp=&arnumber=6394682&isnumber=6394656> (Accessed 7 December 2014).
- Ahmad, A. and Shiratuddin, N. 2010. Business Intelligence for Sustainable Competitive Advantage: Field Study of Telecommunications Industry. In: Proceedings of *Annual International Academic Conference on Business Intelligence and Data Warehousing, Singapore (BIDW 2010)*. 96-102.
- Alasow, M. A., Udomsade, J. and Niyamangkoon, S. 2010. Notice of Retraction People attitude towards telecenter utilization in Roi Et Province of Thailand. In: Proceedings of *International Conference on Education and Management Technology (ICEMT)*. Piscataway, NJ, USA: IEEE, 595-599.
- Arnowitz, J., Arent, M. and Berger, N. 2010. *Effective prototyping for software makers*. Oxford: Elsevier.
- Attarha, M. and Modiri, N. 2011. Focusing on the importance and the role of requirement engineering. In: Proceedings of *the 4th International Conference on Interaction Sciences (ICIS)*. Busan, Korea, 16-18 Aug. 2011. 181-184.
- Avison, D. E. and Fitzgerald, G. 2003. Where now for development methodologies? *Communications of the ACM*, 46 (1): 78-82.
- Azeem, M. and Gondal, M. B. 2011. Prototype framework: Prototypes, prototyping and piloting in terms of quality insurance. *Academic Research International*, 1 (2): 301-307.
- Bailey, A. 2009. Issues affecting the social sustainability of telecentres in developing contexts: A field study of sixteen telecentres in Jamaica. *The Electronic Journal of Information Systems in Developing Countries*, 36 (4): 1-18.
- Bailey, A. and Ngwenyama, O. 2010. Community mediation and violence prevention through telecentre usage: ICTs mediating the 'Border Line'. In: Proceedings of

Proceedings of SIG GlobDev Third Annual Workshop, Saint Louis, USA, December 12.

Bakhouya, M., Campbell, R., Coronato, A., Pietro, G. D. and Ranganathan, A. 2012. Introduction to special section on formal methods in pervasive computing. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7 (1): Art. 6:1-9.

Baron, L. F. and Gomez, R. 2012. Social network analysis of public access computing: Relationships as a critical benefit of libraries, telecenters and cybercafés in developing countries. In: *Proceedings of Proceedings of the 2012 iConference*. ACM, 377-383.

Baskerville, R., Lyytinen, K., Sambamurthy, V. and Straub, D. 2011. A response to the design-oriented information systems research memorandum. *European Journal of Information Systems*, 20 (1): 11-15.

Bayo, I., Barba, M. and Gomez, R. 2012. Better Learning Opportunities through Public Access Computing. Paper presented at the *the Prato CIRN Community Informatics Conference*. Monash Centre, Prato Italy 7-9 November, 1-16.

Bee Bee, C., Bernardo, D. V. and Verner, J. 2010. Understanding the Use of Elicitation Approaches for Effective Requirements Gathering. In: *Proceedings of the 2010 Fifth International Conference on Software Engineering Advances (ICSEA)*. 22-27 Aug. IEEE, 325-330.

Benjamin, P. 2001. *Telecentres and Universal Capability*. PhD, Aalborg University.

Benjamin, P. 2009. Does' Telecentre'mean the centre is far away? Telecentre development in South Africa. *The Southern African Journal of Information and Communication*, 1: 32-50.

Besa, P. J. 2011. Advantages and disadvantages of aspect oriented design in an enterprise environment. Pontificia Universidad Católica de Chile.

Bhattacharjee, A. K. and Shyamasundar, R. 2009. Activity diagrams: a formal framework to model business processes and code generation. *Journal of Object Technology*, 8 (1): 189-220.

Boehm, B. W. 1984. Verifying and validating software requirements and design specifications. In: *Proceedings of IEEE software*. Piscataway, NJ, USA: IEEE, Available: <http://ieeexplore.ieee.org/document/5657584/?arnumber=5657584&tag=1> (Accessed 16 September 2015).

Booch, G., Rumbaugh, J. and Jacobson, I. 1999. *The unified modeling language user guide*. Reading, Massachusetts: Addison Wesley Longman, Inc.

Braude, E. J. and Bernstein, M. E. 2011. *Software engineering: modern approaches*. Long Grove, IL: Waveland Press.

- Britton, C. and Doake, J. 2004. *A student guide to object-oriented development*. Amsterdam: Butterworth-Heinemann.
- Bruegge, B. and Dutoit, A. A. 1999. *Object-oriented software engineering; conquering complex and changing systems*. Upper Saddle River, N.J.: Prentice Hall PTR.
- Burnard, P. 1991. A method of analysing interview transcripts in qualitative research. *Nurse Education Today*, 11 (6): 461-466.
- Chanda, J., Kanjilal, A., Sengupta, S. and Bhattacharya, S. 2009. Traceability of requirements and consistency verification of UML use case, activity and Class diagram: A Formal approach. In: Proceedings of *International Conference on Methods and Models in Computer Science*. Piscataway, NJ, USA: IEEE, 1-4.
- Cheang, S. and Lee, J.-D. 2010. Evaluation telecenter performance in social sustainability context: A Cambodia case study. In: Proceedings of *the 6th International Conference on Advanced Information Management and Service (IMS)*. Piscataway, NJ, USA: IEEE, 135-141.
- Cheuk, S., Atang, A. and Lo, M.-C. 2012. Community Attitudes towards the Telecentre in Bario, Borneo Malaysia: 14 Years on. *International Journal of Innovation, Management and Technology*, 3 (6): 682-687.
- Colcombet, T. and Fradet, P. 2000. Enforcing trace properties by program transformation. In: Proceedings of *the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. Boston, MA, USA, January 19 - 21. ACM, 54-66.
- Colle, R. D. 2005. Memo to telecenter planners. *The Electronic Journal of Information Systems in Developing Countries*, 21 (1): 1-13.
- Creswell, J. W. 2013. *Research design: Qualitative, quantitative, and mixed methods approaches*. Thousand Oaks, Calif.: Sage publications.
- Crow, J. and Di Vito, B. 1998. Formalizing space shuttle software requirements: Four case studies. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7 (3): 296-332.
- Debus, M. 1988. *Methodological review: a handbook for excellence in focus group research*. Washington, DC.: Academy for Educational Development.
- Dennis, A., Wixom, B. H. and Tegarden, D. 2015. *Systems analysis and design: An object-oriented approach with UML*. Hoboken, N.J.: John Wiley & Sons.
- Eker, B. 2014. The effects of prototyping technologies on product design. Paper presented at the *the 8th International Quality Conference*. Kragujevac, 611-616.

- El-Attar, M. 2011. A systematic approach to assemble sequence diagrams from use case scenarios. In: *Proceedings of the 3rd International Conference on Computer Research and Development (ICCRD)*. Piscataway, NJ, USA: IEEE, 171-175.
- ESA Board for Software Standardisation and Control (BSSC). 1994. *Guide to Software Verification and Validation*. The Netherlands: European Space Agency.
- Evans, A., France, R., Lano, K. and Rumpe, B. 2014. Developing the UML as a formal modelling notation. *arXiv preprint arXiv:1409.6928*,
- Filman, R. E. and Friedman, D. P. 2000. *Aspect-oriented programming is quantification and obliviousness*. Moffett Field, CA.: Research Institute of Advanced Computer Science. Available: <https://ntrs.nasa.gov/search.jsp?R=20010071445> (Accessed 15 October 2014).
- Gable, G. G. 1994. Integrating case study and survey research methods: an example in information systems. *European journal of information systems*, 3 (2): 112-126.
- Gibson, J. P. and Méry, D. 1998. Teaching Formal Methods: Lessons to Learn. In: *Proceedings of IWFM*. Citeseer,
- Gomez, R. 2012. *Libraries, telecentres, cybercafes and public access to ICT: International comparisons*. Hershey, PA: Information Science Reference (an Imprint of IGI Global).
- Gomez, R. and Baron-Porrás, L. F. 2011. Does public access computing really contribute to community development? Lessons from libraries, telecenters and cybercafés in Colombia. *The Electronic Journal of Information Systems in Developing Countries*, 49: 1-11.
- Gomez, R., Pather, S. and Dosono, B. 2012. Public access computing in South Africa: Old lessons and new challenges. *The Electronic Journal of Information Systems in Developing Countries*, 52 (1): 1-16.
- Grønmo, R., Krogdahl, S. and Møller-Pedersen, B. 2013. A collection operator for graph transformation. *Software & Systems Modeling*, 12 (1): 121-144.
- Guba, E. G. and Lincoln, Y. S. 1994. Competing paradigms in qualitative research. In: *Handbook of qualitative research*. Thousand Oaks, CA: Sage, 105-117.
- Harris, R. W. 2007. Telecentre evaluation in the Malaysian context. Paper presented at the *the 5th International Conference on IT in Asia*. Kuching, Sarawak, Malaysia, 10-12 July.
- Hassan, S., Yusof, Y., Seman, M. A. A. and Sheik, W. R. 2010. Impact Analysis on Utilization of Telecenter: The Case of Telecentre in Baling.

- Heitmeyer, C. 1998. On the need for practical formal methods. In: *Proceedings of Formal Techniques in Real-Time and Fault-Tolerant Systems*. Lyngby, Denmark, September 14–18. Germany: Springer, 18-26.
- Hevner, V. A., R , March, S. T., Park, J. and Ram, S. 2004. Design science in information systems research. *MIS quarterly*, 28 (1): 75-105.
- Holtzhausen, S. 2001. Triangulation as a powerful tool to strengthen the qualitative research design: the Resource-based Learning Career Preparation Programme (RBLCPP) as a case study.
- Hon, Y. M., Gayen, J.-T. and Ehrich, H.-D. 2008. OOLH: A Formal Framework for Specifying System Requirements. In: *Proceedings of SIGSAND-EUROPE*. 75-78.
- Hudson, H. E. 2001. Telecentre evaluation: Issues and strategies. In: Latchem, C. and Walker, D. eds. *Telecentres: Case studies and key issues*. Vancouver: The Commonwealth of Learning, 169-181.
- Hull, E., Jackson, K. and Dick, J. 2010. *Requirements engineering*. 2nd ed. London: Springer Science & Business Media.
- Hunt, P. 2001. True stories: telecentres in Latin America and the Caribbean. *The Electronic Journal of Information Systems in Developing Countries*, 4 (5): 1-17.
- Hussein, A. 2009. The use of triangulation in social sciences research: Can qualitative and quantitative methods be combined. *Journal of Comparative Social Work*, 1 (8): 1-12.
- Jacobs, S. and Herselman, M. 2006. Information access for development: a case study at a rural community centre in South Africa. *Issues in Informing Science and Information Technology*, 3: 295-306.
- Juan Zheng, X., Liu, X. and Liu, S. 2010. Use case and non-functional scenario template-based approach to identify aspects. In: *Proceedings of the 2010 Second International Conference on Computer Engineering and Applications (ICCEA)*. Piscataway, NJ, USA: IEEE, 89-93.
- Khan, K., Kumar, P., Ahmad, A., Riaz, T., Anwer, W., Suleman, M., Ajmal, O., Ali, T. and Chaitanya, A. 2011. Requirement Development Life Cycle: The Industry Practices. In: *Proceedings of the 9th International Conference on Software Engineering Research, Management and Applications (SERA)*, . Piscataway, NJ, USA: IEEE, 12-16.
- Kim, S. 2013. iScholar: A mobile research support system. M.Sc, University of Regina.

- Kneuper, R. 1997. Limits of formal methods. *Formal Aspects of Computing*, 9 (4): 379-394.
- Kof, L., Gacitua, R., Rouncefield, M. and Sawyer, P. 2010. Ontology and Model Alignment as a Means for Requirements Validation. In: Proceedings of *the IEEE Fourth International Conference on Semantic Computing (ICSC)*. 22-24 Sept. Piscataway, NJ, USA: 46-51.
- Köhler, H. J., Nickel, U., Niere, J. and Zündorf, A. 2000. Integrating UML diagrams for production control systems. In: Proceedings of *the 22nd international conference on Software engineering*. ACM, 241-251.
- Kontio, J., Lehtola, L. and Bragge, J. 2004. Using the focus group method in software engineering: obtaining practitioner and user experiences. In: Proceedings of *the International Symposium on Empirical Software Engineering, ISESE*. Piscataway, NJ, USA: IEEE, 271-280.
- Kordon, F. 2002. An introduction to rapid system prototyping. *IEEE Transactions on Software Engineering*, 28 (9): 817-821.
- Kosalge, P. and Chatterjee, D. 2011. Look before you leap into ERP implementation: an object-oriented approach to business process modeling. *Communications of the Association for Information Systems*, 28 (1): 509-536.
- Krishnan, P. 2003. *A framework for analyses of use case descriptions*. Available: https://www.researchgate.net/profile/P_Krishnan2/publication/228973656_A_Framework_for_Analyses_of_Use_Case_Descriptions/links/0c96053bef44ec55a0000000.pdf (Accessed 15 February 2015).
- Lashgarara, F., Karimi, A. and Mirdamadi, S. M. 2012. Effective factors on the villagers use of rural telecentres (case study of Hamadan province, Iran). *African Journal of Agricultural Research*, 7 (13): 2034-2041.
- Lethbridge, T. C., Sim, S. E. and Singer, J. 2005. Studying software engineers: Data collection techniques for software field studies. *Empirical software engineering*, 10 (3): 311-341.
- Lethbridge, T. C., Singer, J. and Forward, A. 2003. How software engineers use documentation: The state of the practice. *IEEE Software*, 20 (6): 35-39.
- Lincoln, Y. S., Lynham, S. A. and Guba, E. G. 2011. Paradigmatic controversies, contradictions, and emerging confluences, revisited. In: Denzin, N. K. and Lincoln, Y. S. eds. *The Sage Handbook of Qualitative Research*. Los Angeles: Sage, 97-128.
- Ling, Y., Jing, L. and Xiaoshan, L. 2009. Validating Requirements Model of a B2B System. In: Proceedings of *the Eighth IEEE/ACIS International Conference on*

Computer and Information Science, ICIS. 1-3 June 2009. Piscataway, NJ, USA: 1020-1025.

Linzhang, W., Jiesong, Y., Xiaofeng, Y., Jun, H., Xuandong, L. and Guo, Z. 2004. Generating test cases from UML activity diagram based on gray-box method. In: Proceedings of *the 11th Asia-Pacific Software Engineering Conference*. Piscataway, NJ, USA: IEEE, 284-291.

Liu, F. and Yang, M. 2005. Validation of system models. In: Proceedings of *the IEEE International Conference Mechatronics and Automation*. Piscataway, NJ, USA: IEEE, 1721-1725.

Maiden, N. A. M. 1998. CREWS-SAVRE: Scenarios for acquiring and validating requirements. *Automated Software Engineering*, 5 (4): 419-446.

Martakis, A. and Daneva, M. 2013. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners. In: Proceedings of *the IEEE Seventh International Conference on Research Challenges in Information Science (RCIS)*. Piscataway, NJ, USA: IEEE, 1-11.

McConnell, S., Richardson, D., Doehler, M. and Wong, W. 2001. *Telecentres Around the World: Issues to be considered and lessons learned*. Available: http://portal.unesco.org/ci/en/file_download.php/053c2bb713f94903fc72a2a910a4e495Telecentres+around+the+world.pdf (Accessed 01 March 2014).

McDermid, J., Galloway, A., Burton, S., Clark, J., Toyn, I., Tracey, N. and Valentine, S. 1998. Towards industrially applicable formal methods: Three small steps, and one giant leap. In: Proceedings of *the Second International Conference on Formal Engineering Methods*. Piscataway, NJ, USA: IEEE, 76-88.

Mertens, D. M. 2015. *Research and evaluation in education and psychology: Integrating diversity with quantitative, qualitative, and mixed methods*. 4th ed. Thousand Oaks, Calif.: Sage.

Min, S.-J. 2010. From the digital divide to the democratic divide: Internet skills, political interest, and the second-level digital divide in political internet use. *Journal of Information Technology & Politics*, 7 (1): 22-35.

Mohamed, A. Y. A., Hegazy, A. E. F. A. and Dawood, A. R. 2010. Aspect Oriented Requirements Engineering. *Computer and Information Science*, 3 (4): p135.

Moore, A. and Stilgoe, J. 2009. Experts and anecdotes the role of “anecdotal evidence” in public scientific controversies. *Science, Technology & Human Values*, 34 (5): 654-677.

Moscove, S. A. 2011. Prototyping: An Alternative Approach To Systems Development Work. *Review of Business Information Systems (RBIS)*, 5 (3): 65-72.

- Mossberger, K., Tolbert, C. J. and Hamilton, A. 2012. Broadband Adoption| Measuring Digital Citizenship: Mobile Access and Broadband. *International Journal of Communication*, 6: 37.
- Munassar, N. M. A. and Govardhan, A. 2011. Comparison between traditional approach and object-oriented approach in software engineering development. *International Journal of Advanced Computer Science and Applications*, 2 (6): 70-76.
- Muskens, J., Bril, R. J. and Chaudron, M. R. 2005. Generalizing consistency checking between software views. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05). Piscataway, NJ, USA: IEEE, 169-180.
- Naik, G. 2011. Designing a sustainable business model for e-governance embedded rural telecentres (EGERT) in India. *IIMB Management Review*, 23 (2): 110-121.
- Naik, G., Joshi, S. and Basavaraj, K. 2012. Fostering inclusive growth through e-governance embedded rural telecenters (EGERT) in India. *Government Information Quarterly*, 29: S82-S89.
- Nunamaker Jr, J. F. and Chen, M. 1990. Systems development in information systems research. In: Proceedings of System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on. IEEE, 631-640.
- Ogata, S. and Matsuura, S. 2010. Evaluation of a use-case-driven requirements analysis tool employing web UI prototype generation. *WSEAS Transactions on Information Science and Applications*, 7 (2): 273-282.
- Ostroff, J. S. 1992. Formal methods for the specification and design of real-time safety critical systems. *Journal of Systems and Software*, 18 (1): 33-60.
- Ouyang, M. 2014. Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability Engineering & System Safety*, 121: 43-60.
- Pancham, J. and Millham, R. 2015. Design phase consistency: A tool for reverse engineering of uml activity diagrams to their original scenarios in the specification phase. In: *Computational Science and Its Applications-ICCSA 2015*. Heidelberg: Springer, 655-670.
- Pancham, J., Millham, R. and Singh, P. 2013. A validated model for operational monitoring of telecentres' activities in a developing country. In: Steyn, J. and Van der Vyver, A. G. eds. Proceedings of *Public and private access to ICTs in developing regions, 7th International Development Informatics Association Conference*. Bangkok, Thailand, 103 November 2013. 103-125.
- Pandey, D., Suman, U. and Ramani, A. 2010. An effective requirement engineering process model for software development and requirements management. In:

Proceedings of *the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom)*. Piscataway, NJ, USA: IEEE, 287-291.

Parkinson, S. 2005. *Telecentres, access and development: experience and lessons from Uganda and South Africa*. Warwickshire, UK: IDRC.

Pather, S. and Gomez, R. 2010. Public Access ICT: A South-South comparative analysis of libraries, telecentres and cybercafés in South Africa and Brazil. In: Proceedings of *AMCIS*. Lima Peru, 12- 15 August. Paper 526.

Peppers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V. and Bragge, J. 2006. The design science research process: a model for producing and presenting information systems research. In: Proceedings of *the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006)*. Claremont, California, DESRIST, 83-106.

Peterson, J. L. 1977. Petri nets. *ACM Computing Surveys (CSUR)*, 9 (3): 223-252.

Petty, N. J., Thomson, O. P. and Stew, G. 2012. Ready for a paradigm shift? Part 1: introducing the philosophy of qualitative research. *Manual therapy*, 17 (4): 267-274.

Pohl, K. 2010. *Requirements engineering: fundamentals, principles, and techniques*. Berlin: Springer.

Raatikainen, M., Mannisto, T., Tommila, T. and Valkonen, J. 2011. Challenges of requirements engineering: A case study in nuclear energy domain. In: Proceedings of *the 19th International Requirements Engineering Conference (RE)*. Piazza Duomo Trento, Italy, 29 Aug - 2 Sept. Piscataway, NJ, USA: 253-258.

Rafe, V., Rafeh, R., Azizi, S. and Miralvand, M. R. Z. 2009. Verification and validation of activity diagrams using graph transformation. In: Proceedings of *Computer Technology and Development, 2009. ICCTD'09. International Conference on*. IEEE, 201-205.

Rahmat, R., Ahmad, A., Razak, R., Din, R. and Abas, A. 2013. Sustainability model for rural telecenter using business intelligence technique. *International Journal of Social Human Science and Engineering*, 7 (12): 1356-1361.

Rajapakse, J. 2012. Impact of telecentres on Sri Lankan society. In: Proceedings of *the 2012 8th International Conference on Computing and Networking Technology (ICCNT)*. Gyeongju, Korea (South), IEEE, 281-286.

Ramjugath, V. 2015. *Private Interview* Durban, South Africa: Durban University of South Africa

Rantapuska, T. and Millham, R. 2010. 15P. Applying Organisational Learning to User Requirements Elicitation.

- Razak, N. A., Hassan, Z. and Din, R. 2010. Bridging the Digital Divide: An Analysis of the Training Program at Malaysian Telecenters.
- Robertson, S. and Robertson, J. 2012. *Mastering the requirements process: Getting requirements right*. Upper Saddle River, NJ: Addison Wesley.
- Robson, C. 2011. *Real world research: a resource for users of social research methods in applied settings*. Wiley Chichester.
- Rodríguez, A., Fernández-Medina, E., Trujillo, J. and Piattini, M. 2011. Secure business process model specification through a UML 2.0 activity diagram profile. *Decision Support Systems*, 51 (3): 446-465.
- Royce, W. W. 1970. *Managing the development of large software systems*. Available: <http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf> (Accessed 17 June 2014).
- Runge, O., Ermel, C. and Taentzer, G. 2011. AGG 2.0—new features for specifying and analyzing algebraic graph transformations. In: *Applications of Graph Transformations with Industrial Relevance*. Berlin, Heidelberg: Springer, 81-88.
- Rushby, J. 1997. Formal methods and their role in the certification of critical systems. In: Shaw, R. ed. *Safety and reliability of Software based systems*. London: Springer, 1-42.
- Sanou, B. 2016. *ICT Facts & Figures The world in 2015. ICT Data and Statistics Division Telecommunication Development Bureau International Telecommunication Union. 2015*. Geneva, Switzerland: International Telecommunication Union. Available: <http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2016.pdf> (Accessed 12 June 2016).
- Satzinger, J., Jackson, R. and Burd, S. D. 2011. *Systems analysis and design in a changing world*. Boston, MA: Course Technology, Cengage Learning.
- Satzinger, J. W., Jackson, R. B. and Burd, S. D. 2012. *Introduction to systems analysis and design: An agile, iterative approach*. Boston, MA: Course Technology, Cengage Learning.
- Schach, S. R. 2008. *Object Oriented and Classical Software Engineering*. 8th ed ed. New York: McGraw-Hill.
- Seman, M. A. A., Ibrahim, H. H., Kasiran, M. K., Yusop, N. I., Aji, Z. M., Dahalin, Z. M. and Yasin, A. 2013. Community Characteristics for Self-Funding and Self-Sustainable Telecenter. *Global Journal on Technology*, 3: 1666-1671.

Sey, A. and Fellows, M. 2009. *Literature review on the impact of public access to information and communication technologies*. 2009: Available: <http://library.globalimpactstudy.org/sites/default/files/docs/CIS-WorkingPaperNo6.pdf> (Accessed 25 February 2013).

Simmons, E. 2005. The usage model: a structure for richly describing product usage during design and development. In: *Proceedings of the 13th IEEE International Conference on Requirements Engineering*. Paris, France, 29 Aug. - 2 Sept. Piscataway, NJ, USA: IEEE, 403-407.

Sitole, M. 2014. *Private Interview* Durban, South Africa: Durban University of Technology.

Somé, S. S. 2005. Use cases based requirements validation with scenarios. In: *Proceedings of the 13th IEEE International Conference on Requirements Engineering*. Paris France, 29 Aug.-2 Sept. 2005. Piscataway, NJ, USA: IEEE, 465-466.

Somé, S. S. 2007. *Use Case Editor (UCed) User Guide Version 1.6.2*. Available: <http://www.site.uottawa.ca/~ssome/UCedWeb/publis/userGuide.pdf> (Accessed 10 March 2014).

Sommerville, I. 2011. *Software engineering*. 9th ed. Boston: Pearson Education.

Sopazi, P. N. and Andrew, T. 2008. Evaluation of a Telecentre Using Stakeholder Analysis and Critical Systems Heuristics: A South African Case Study. *Scientific Inquiry*, 9 (1): 19-28.

Stevens, P. and Pooley, R. J. 2006. *Using UML: software engineering with objects and components*. 2nd ed. Harlow, Essex: Pearson Education.

Stoica, M., Mircea, M. and Ghilic-Micu, B. 2013. Software Development: Agile vs. Traditional. *Informatica Economica*, 17 (4): 64-76.

Stoilov, T. and Stoilova, K. 2006. Automation in business processes. In: *Proceedings of International Conference" Systems for Automation of Engineering and research-SAER*. 23-24.

Swain, R. K., Panthi, V. and Behera, P. K. Generation of test cases using activity diagram. *International Journal of Computer Science and Informatics*, 3 (2): 1-10.

Taentzer, G. 2003. AGG: A graph transformation environment for modeling and validation of software. In: *Proceedings of International Workshop on Applications of Graph Transformations with Industrial Relevance*. London, UK.: Springer, 446-453.

Telecommunication Regulatory Authority. Sultanate of Oman. 2012. *TRA Position paper on Telecentres*. Available:

https://www.tra.gov.om/pdf/563_trapositionpaperontelecetnersen.pdf (Accessed 21 June 2015).

Townsend, L., Sathiaseelan, A., Fairhurst, G. and Wallace, C. 2013. Enhanced broadband access as a solution to the social and economic problems of the rural digital divide. *Local Economy*, 28 (6): 580-595.

UML, O. 2005. 2.0 specification. URL <http://www.omg.org/technology/documents/formal/uml.htm>,

USAASA. 2011. *USAASA Business plan 2011 - 2012*. South Africa: Available: http://www.usaasa.org.za/export/sites/usaasa/resource-centre/download-centre/downloads/USAASA_Business_Plan_2011-2012.pdf (Accessed 16 May 2015).

Van Lamsweerde, A. 2001. Goal-oriented requirements engineering: A guided tour. In: Proceedings of *the 5th IEEE International Symposium on Requirements Engineering*. Toronto, Canada, 27-31 Aug. Piscataway, NJ, USA: IEEE, 249-262.

Van Lamsweerde, A. 2004. Goal-oriented requirements engineering: a roundtrip from research to practice [engineering read engineering]. In: Proceedings of *the 12th IEEE International Requirements Engineering Conference*. 6-11 Sept. 2004. Piscataway, NJ, USA: 4-7.

Veeraraghavan, R., Singh, G., Toyama, K. and Menon, D. 2006. Kiosk usage measurement using a software logging tool. In: Proceedings of *Information and Communication Technologies and Development, 2006. ICTD'06. International Conference on*. IEEE, 317-324.

Walsham, G. and Sahay, S. 2006. Research on information systems in developing countries: Current landscape and future prospects. *Information Technology for Development*, 12 (1): 7-24.

Watson, A. 2008. *Visual Modelling: past, present and future*. Available: http://www.omg.org/UML/Visual_Modeling.pdf (Accessed 25 Aug 2015).

Wieringa, R. 2010. Design science methodology: principles and practice. In: Proceedings of *the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. Cape Town, South Africa, 01-08 May. New York: ACM, 493-494.

Wilson, W. M., Rosenberg, L. H. and Hyatt, L. E. 1997. Automated analysis of requirement specifications. In: Proceedings of *the 19th International Conference on Software Engineering*. Boston, MA, 17-23 May. New York: ACM, 161-171.

Woodcock, J., Larsen, P. G., Bicarregui, J. and Fitzgerald, J. 2009. Formal methods: Practice and experience. *ACM computing surveys (CSUR)*, 41 (4): Article 19.

Xiaodong, L. 2007. A review of SOA. *Computer Applications and Software*, 24 (10): 122-124.

Xuping, J. 2008. Modeling and Application of Requirements Engineering Process Metamodel. In: *Proceedings of the IEEE International Symposium on Knowledge Acquisition and Modeling Workshop*. 21-22 Dec. 2008. Piscataway, NJ, USA: 998-1001.

Yang, H., De Roeck, A., Gervasi, V., Willis, A. and Nuseibeh, B. 2011. Analysing anaphoric ambiguity in natural language requirements. *Requirements Engineering*, 16 (3): 163-189.

Yeasmin, S. and Rahman, K. F. 2012. Triangulation research method as the tool of social science research. *BUP Journal*, 1 (1): 154-163.