Design-Reality Gaps in Open Source Information Systems Development:
An Action Research Study of Education and Healthcare Systems in Tanzania

by

Juma Hemed Lungo

Thesis Submitted in Partial Fulfilment of the Requirements

of the Degree of Doctor of Philosophy

at the Faculty of Mathematics

and Natural Sciences

University of Oslo

NORWAY

11th June 2008

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
Nr. 808

ISSN 1501-7710

Cover: Inger Sandved Anfinsen.
Printed in Norway: AiT e-dit AS, Oslo, 2008.

Produced in co-operation with Unipub AS.
The thesis is produced by Unipub AS merely in connection with the
thesis defence. Kindly direct all inquiries regarding the thesis to the copyright
holder or the unit which grants the doctorate.

Unipub AS is owned by
The University Foundation for Student Life (SiO)

To my wife, Masala, my son, Madende, and my daughter, Msule,
who bring meaning to everything I do

ACKNOWLEDGEMENTS

I am indebted to all who provided their support to my study in various ways. First and foremost I would like to sincerely thank my supervisors Prof. Jens Kaasbøll and Prof. Jørn Braa for their invaluable input, in the form of very useful guidance, advice, comments and constructive criticisms that helped shape my work. To them I say *Ahsanteni Sana!*

I owe a great deal of thanks to the HISP and Zalongwa projects for providing me with such conducive fieldwork case studies in Tanzania. Specifically I would like to show my appreciation to Prof. Jørn Braa as the overall project coordinator for HISP and to other HISP members in Zanzibar: Abubakar Diwani, Yahya El Hamad, Masudi Mahundi, Patrick Burassa, Dr Suleiman Omary and Dr Boudewijn Peters for insightful discussion, implementation and training workshops. Zalongwa Project members have given invaluable support to my work and I would like to thank the manager, Jillahoma Mussa, and all other members at the University of Dar es Salaam; Simphorosa Mchallo, Tatu Issike, Yusta Mutalemwa, Dr Ibrahimu Juma, Dr Abdallah Chungu, Bakari Rashidi, Dr Frank Tilya, Stanley John, Jonas Tiboroha, Mustafa Mgera, Sophia Kivina, Jamillah Isalwa, Mariamu Miraji. Bernada Ernest, Ally Said Masomaso and Luba Pascoe.

I extend my appreciation to my employer, the University of Dar es Salaam, for granting me study leave. I would like to mention especially Dr Hashimu Twaakyondo, the Head of the Department of Computer Science, and Prof. Rogarth Kivaisi, the then Dean of the Faculty of Science, for granting me initial release.

I would like to convey my gratitude to my fellow doctoral students Faraja Igira Mukama and Gertrudes Macueve for providing a tranquil academic and social atmosphere for the whole duration of my study. I appreciate their friendship over the years we have been together. I would also like to thank Rinda for English proofreading of this thesis.

I wish to express my deepest and most sincere gratitude to my family. Very special thanks and credit goes to my parents; father Hemed Mkulago, and mother Khadija Zuberi, who inspired me to have a mind of my own. Moreover my gratitude goes to my sister Salma Mkulago and brothers Shabani Lungo, Abdallah Lungo and Sadiki Lungo for the exuberant support they have shown.

I would like to convey my gratitude to my loving wife, Masala-Tatu, for her enormous support and encouragement all through the years of this study. Thanks to my son, Madende-Hemed and daughter Msule-Fatuma, who at their toddler age, displayed a high degree of patience for the nights and days I was away wrestling with this intense research undertaking. I deeply appreciate the support of my sisters, Maria Kigola, Sakina Abdul and Upendo Kigoda, who took care of the children when my wife joined me in Norway.

Last but not least, I am particularly grateful to the Norwegian State Education Loan Fund Lånekassen for providing financial assistance to undertake my studies in Norway. I thank the academic and administrative staff at the Institute for Informatics (IFI), University of Oslo, for providing me with necessary facilities that aided my study and for providing courses that were enlightening in respect to my doctoral thesis.

Thank you again to you all and to those that I have here forgotten to mention.

Juma Hemed Lungo

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

This thesis, *Design-Reality Gaps in Open Source Information Systems Development: An Action Research Study of Education and Healthcare Systems in Tanzania*, is submitted in partial fulfilment of the requirement of degree of Doctor of Philosophy at the Faculty of Mathematics and Natural Sciences, University of Oslo, Norway. The research was funded by the Norwegian State Education Loan Fund Lånekassen. This thesis comprises of seven introductory chapters and five scientific papers. The papers, as listed below, are included at the end of the introductory chapters.

i. Lungo, J. H., & Kaasbøll, J. (2007). The Use of Open Source Software in the Public Sector: Cases from Tanzania and Norway. Submitted to: *Information and Organization Journal (*previous version has been published In Silva**,** L**.,** Westrup, C. & Reinhard, N (Eds.), *Proceedings of the Ninth International Working Conference of IFIP WG 9.4: Social Implications of Computers in Developing Countries*, (pp.1-14), São Paulo, Brazil.

ii. Lungo, J. H., & Igira, F. (2008). Development of Health Information System in Zanzibar: Practical Implications. *Journal of Health Informatics in Developing Countries*, 2(1), 24-32.

iii. Lungo, J. H. (2008). The Reliability and Usability of District Health Information Software: Case Studies from Tanzania. *Tanzania Journal of Health Research 10*(1), 39-45.

iv. Lungo, J. H. (2005). Re-inventing Higher Learning Institutions Communication Media: The Case of University of Dar es Salaam Student Information System. In A.O. Bada & A Okunoye (Eds.), *Proceedings of the Eight International Working Conference of IFIP WG 9.4: Social Implications of Computers in Developing Countries*, (pp.194-208), Abuja, Nigeria.

v. Lungo, J. H. (2006). Critical Issues Associated with Adoption and Use of Open Source Software in Public Sector: Insights from Tanzania. In J. Ljunberg & M. Andersson (Eds), *Proceedings of the Fourteenth European Conference on Information Systems*, (pp.732-744), Göteborg, Sweden.

ABSTRACT

This thesis, *Design-Reality Gaps in Open Source Information Systems Development: An Action Research Study of Education and Healthcare Systems in Tanzania*, presents a theoretical and empirical informed analysis of Free Open Source Software (FOSS) development in the domain of health and education information systems in Tanzania. Historically, FOSS development has been driven by user-developer communities who are also the users of FOSS applications. The use of FOSS applications in information systems (IS) characterised by distinctive users and developer communities as separate groups has received limited attention in FOSS literature. The FOSS development approach, as well as the justifications for using FOSS in the infrastructure domain where users are developers, are inherently problematic when applied in the IS domain in developing countries. There is an urgent need to identify alternative conceptualisations of the FOSS phenomenon suitable to the goals and context of information systems in developing countries. The thesis focuses on the interplay between the sociotechnical conditions of IS in developing countries and the FOSS development approach. The thesis objectives are: (i) to develop an alternative explanation of the Free Open Source Software phenomenon in the context of information systems in developing countries and (ii) to analyse and address the challenges shaping FOSS development in order to enable Tanzania in particular and developing countries in general to benefit from adopting FOSS.

The thesis is informed by power, translation, and network analysis perspectives of the Actor Network Theory, with additional concepts from networks of action and the design-reality gaps analysis. These concepts are used to build a framework for the *design-reality* gaps in the context of FOSS development in developing countries. The framework identifies three archetypal situations: *developer – sponsor*, *global developer – local developer*, and *local developer – local user* gaps.

The thesis draws its empirical material from two case studies of implementing open source information systems in the health and education sectors in Tanzania from 2005 to 2007. The research design is based on participatory action research in specific information systems: the District Health Information Software (DHIS) in the health information system and the Student Academic Register Information Software (SARIS) in the education information system.

FOSS development in developing countries centres on the formation of sustainable collaborative networks through sharing of software and knowledge. These networks are important in helping a developing country to support the day to day customisation and managing of FOSS products. Implementing FOSS in IS requires substantial investment on localising the software, training users, and developing support networks. An alternative conceptualisation of FOSS development which emphasises co-located project organisations as a coping strategy to meet the challenges of social-technical influences is advisable. This is a different approach from working on virtual teams. Furthermore, the thesis identifies the role of political negotiations in supporting FOSS development in IS.

Proposed strategies for bridging the *developer–sponsor* gap are to facilitate understanding of FOSS philosophy among global and local networks, to facilitate political negotiations, and to promote the private sector. The *global developer–local developer* gap can be addressed through focusing on capacity building, mutual learning between global and local developers (through how-to and hands-on support), and implementing FOSS technologies curriculum in the education system. The last gap, *local developer–local user*, can be addressed through organising FOSS projects in the form of participatory actions between users and developers and by creating focused user training.

CHAPTER 1: INTRODUCTION

1.1 The Research Motivation and Concern

There is increasing consensus that FOSS is a good opportunity for developing countries to catch-up with the increased widening of the digital divide.[1] Currently, the case for a developing country like Tanzania to adopt FOSS driven Information and Communication Technologies (ICT) implementation strategies is a compelling one. Many justifications for developing countries to adopt FOSS have been cited, including the notion that FOSS reduces software licensing costs, helps developing countries avoid being locked-in to proprietary software, advances knowledge through access to the source codes, and it is a means for setting up an information economy (Câmara & Fonseca, 2007; May, 2006; Meystre & Müller, 2005; Weber, 2003; Weerawarana & Weeratunga, 2004).

Most attempts for implementing ICTs have ended up on the shelves (Bhatnagar, 2000; Bhatnagar & Bjørn-Andersen, 1990; Heeks, 2002). Even in government-backed ICT projects, there is a huge gulf between the hype about the role of Information Technology (IT) and reality (Heeks, 2006). Like any other process of introducing technology, FOSS implementation in developing countries is subject to the same challenges other technologies face due to the nature of the context.

Using examples of ICT initiatives in government supportive systems, Heeks (2003) argues that central to e-government success and failure is the amount of change between 'where we are now' and 'where the e-government project wants to get us'.

> 'Where we are now' means the current realities of the situation. 'Where the e-government project wants to get us' means the model or conceptions and assumptions built into the project's design. E-government success and failure therefore depend on the size gap that exists between 'current realities' and 'design of the e-government project'. The larger this design-reality gap, the greater the risk of e-government failure. Equally, the smaller the gap, the greater the chance of success. (Heeks, 2003, p.3)

According to Heeks (2003), the design-reality gap exists around seven dimensions abbreviated as ITPOSMO: Information, Technology, Processes, Objectives and values, Staffing and skills, Management systems and structures, and other resources such as

[1] the gap between those able to benefit by digital technologies and those who are not (www.digitaldivide.org)

time and money. The design-reality gap analysis indicates that FOSS, like other ICT initiatives in developing countries, would face many common constraints from hardware, training, and basic infrastructures such as electricity, Internet, and telecommunications lines (Musa, Mbarika, & Meso, 2005).

Moreover, the freedoms envisioned in FOSS are by themselves a threat to its adoption in developing countries. For example, as FOSS advocates low cost software procurement, it may be seen as a threat to corrupt politicians who assume that FOSS would reduce their potential kick-backs from the procurement of software packages. Thus, in addition to technical skills and infrastructure, politics is another major problem for FOSS development. A thorough explanation of the FOSS phenomenon through detailed empirical study would enlighten the politicians and businessmen in developing countries.

1.2 Statement of the Problem

Weber (2004) points out that combining FOSS tools with the technical workforce available in developing countries can enable technology transfer. Weber (2004) argues that 'the essence of open source is not the software; it is the process by which software is created' (p.56). FOSS should have far-reaching effects, as Weber (2004) says, 'of course information technology and open source in particular is not a silver bullet for longstanding development issues; nothing is. But the transformative potential of computing does create new opportunities to make progress on development problems that have been intransigent' (p. 254).

However, FOSS as technology is context sensitive in terms of practical implementation in different organisations and in different geographical areas, with various levels of income and IT infrastructures. Fitzgerald (2006) agrees that FOSS offers a real paradigm shift in how organisations adopt ICT, but he points to many challenges in making FOSS work effectively in developing countries. The issues are limited institutional mechanisms to support, business models that support FOSS ideology, licensing arrangements, technical capacity to deal with FOSS development, capacity of universities to offer relevant training, and knowledge of FOSS and language barriers (Fitzgerald, 2006). This implies that there is much potential for FOSS in developing

countries, but many challenges with its implementation.

Currently, there is an increasing rate of FOSS uptake in developing countries (FOSSFA, 2004). However, many are based on the infrastructure side, adopting the Linux operating system and server side programs (web server, mail server, database, and file sharing utilities). The use of FOSS in specific software applications such as human resource databases, payroll, accounting, education systems, and health information systems is still emerging and few studies have been conducted. The use of FOSS applications in information systems domain characterised by distinctive users and developer communities as separate groups has received limited attention in the FOSS literature. The FOSS development approach, as well as the justifications for using FOSS in the infrastructure domain where users are developers, are inherently problematic when applied in the IS domain in developing countries. There is an urgent need to identify alternative conceptualisations of the FOSS phenomenon suitable to the goals and context of information systems in developing countries.

Tanzania, one of the least developing countries, would benefit from FOSS if accepted and practiced. This thesis was an attempt to decode the FOSS liberation (Chopra & Dexter, 2008). The result was rich insights into FOSS and better strategies for the adoption of FOSS in developing countries.

1.3 Objectives of the Thesis

My thesis was informed by the following objectives.

- To develop an alternative explanation of the Free Open Source Software phenomenon in the context of information systems in developing countries.
- To analyse and address the challenges shaping FOSS development in order to enable Tanzania in particular and developing countries in general to benefit from adopting FOSS.

In this thesis, I studied the conceptualisation of FOSS and the way the FOSS community was organised and practiced software development, source code sharing, and economic incentives. This exploration of the FOSS phenomenon served as

background information for my case studies and was useful for introducing FOSS to those unfamiliar with this phenomenon. The current FOSS literature is focusing on the use of FOSS in infrastructure where users are developers. Little is said on the context where users are not computer professionals. The first objective is concerned with identifying ways to present FOSS in the IS domain of a developing country.

The FOSS phenomenon is treated as a viable alternative strategy for ICT adoption in developing countries. The main justification is that FOSS enables countries to save money from software licensing costs; it promotes indigenous technological development and facilitates technology translation (Weerawarana & Weeratunga, 2004). However, there is a need to determine how to increase participation of Tanzanians in FOSS development projects.

To meet these objectives, first, I conducted an explorative case study in eight organisations from Tanzania and Norway. Exploration of the use of FOSS in a developed country (Norway) and a developing country (Tanzania) provided insight with which to discuss the FOSS phenomenon. Action research is the second route I took in this study. I was emphatically engaged in the actual implementation of two large-scale Open Source Information Systems (OSIS). The first case study was the implementation of the Health Information System (HIS) in Zanzibar. The second case study was the Student Academic Register Information System (SARIS) at the University of Dar-es-Salaam. The experience gained from my participation in the two projects allowed me to share my personal and scientific insights on FOSS development in Tanzania.

1.4 Theoretical Perspectives Overview

Lessons from the design-reality gaps (Heeks, 2003), which explain the failure of most ICT initiatives, indicate that the gaps are widening as developers ignore social conditions (people, culture, and politics) in which technological change process occurs. This technological deterministic approach, which takes for granted that there is no influence from the social conditions embedded in the context of ICT implementation, is a major reason for project failures (Dada, 2006, Heeks, 2003). An alternative outlook of ICT implementation is to take the social systems approach. Social systems perspectives

basically take account of the social conditions while implementing technological change. I draw on Actor-Network Theory (ANT) notions of *translation* (Callon, 1986) and *network analysis* (Law & Callon, 1992) as my analytical framework for studying the implementation of FOSS in Tanzania. The ANT reveals that technological changes in information systems imply not only technical re-design, but re-design of an entire socio-technical network and translating and aligning different actors' interests.

In ANT perspective, the interests of heterogeneous actors (human and non-human) are inscribed in artefacts and interact in order to be translated. As a result, actors form alliances of networks in order to mobilise support for a particular solution of their interest (Bijiker, Hughes & Pinch, 1987; Latour, 1987). Thus, ANT makes roles of the social conditions as important as technology artefacts. Furthermore, the ANT notion of network analysis model introduces the concept that, in a technological change innovation project, there are two networks: *local network* and *global network* (Law & Callon, 1992). The global network represents the outside of the project's local settings and context, built up to enable the project to take place with the resources it provides (Law & Callon, 1992). Project resources include money, expertise, and political support (Law & Callon, 1992). In many projects in developing countries, the global network represents the donor community, who funds and provides technical expertise to the projects. On the other hand, the usually local people work inside of a project to produce a successful working tool using the funds and expertise provided by the global network (Law & Callon, 1992).

As FOSS development is driven by geographically distributed developers located globally, analysis of global and local networks is important. It helps to analyse the proximity of the local development team to global support, e.g., how the software development team of the District Health Information Software (DHIS) in Zanzibar gets support from the global Health Information System Programme (HISP) in terms of funds and technical support. Learning the organisational arrangement of the project and the way local development overcomes the challenges imposed by limited infrastructure and communication lines would help to advise working strategies for adopting FOSS in a developing country like Tanzania.

In addition to ANT perspectives, I drew on other two perspectives: *technology translation* (Nhampossa, 2006) and *networks of action* (Braa et al., 2004). Although the technology translation perspective is based on ANT's translation notion, it carries additional concepts that influence the transfer of technology from one developing country to another. Nhampossa (2006) argues that, 'technology needs to be sustainable, at the same time needs to remain flexible enough to accommodate changes occurring over time and space' (p.57). This implies that not only the process of transferring technology, but also the characteristic of the technology itself has a role in its diffusion in the destination context.

A more flexible technology like FOSS, which ships with its trade secrets, has the potential to be localised and hence appropriated locally while maintaining its international flavour (Nhampossa, 2006). Furthermore, Braa et al. (2004) argue that localised individual initiatives should be connected as large networks of action through sharing experiences and mutual learning to ensure long-term scalability and sustainability. However, the networks of action proposal on sharing knowledge could better be implemented under technology free of intellectual property laws; otherwise, intellectual property rights would become barriers.

Drawing from the theoretical discussion, the thesis's analytical framework on the development of health and education open source information systems in Tanzania is framed as being influenced by *politics*, *development process*, *infrastructure,* and *relevant skills*. These four social conditions influence FOSS development process in developing countries.

1.5 Research Contributions

The study contributions were two-fold: theoretical and practical. The theoretical contribution was the following:

- ☑ *Re-conceptualisation of the FOSS phenomenon perceived as an ongoing learning process on the way FOSS was interpreted and its development was practiced in the context of information systems in the developing countries.*

The practical contribution was the following:

- ☑ *Practical implications as guidelines for ICT professionals and managers in the health and*

1.6 The Study Context: Tanzania and Ujumaa Policy

In this study, I adopted multiple case studies, but the main audience of the study findings was Tanzania, where the main research settings were located. Situated in East Africa, mainland Tanzania became independent from British rule in 1961 and was united with Zanzibar in 1964, when it became the United Republic of Tanzania. In 2002, Zanzibar had a population of 984,625 and the mainland had a population of 33,584,607. By the year 2007, Tanzania was estimated to have a population of over 35 million people.

As a Tanzanian, I have had personal contact with the FOSS philosophy. Tanzania is a country that has embraced socialism for years. Though currently the government of Tanzania does not practise socialism, its legacy is very much alive. Although FOSS developers deny having a political agenda, sociologists believe that this denial is enacted through a particular cultural exercise of free speech facilitating the broad mobility of FOSS as artefacts and metaphors, laying the groundwork for its informal political scope, which is 'its key role as a catalyst by which to rethink the assumptions of intellectual property rights through its use and inversion' (Coleman, 2004a, p. 508). Kelty (2000) adds that this makes FOSS 'the most powerful political movement on the Internet, even though most of its proponents spend all their extra energy denying that it is political' (p.6).

The core concept of FOSS is that full access to software source codes must be granted (Stallman, 2002). FOSS can be used on any computer and in any situation (Perens, 2005). Users can improve FOSS by fixing bugs and augmenting functionality; they can then redistribute it normally as FOSS (Rosen, 2005). This vision is clearly stipulated in two key documents that guide the choice and creation of a Free Open Source Software Licence: the Free Software Definition (Stallman, 2002) and the Open Source Definition (Perens, 2005). FOSS underscores the freedom of an individual's right to create, use, and distribute software in a manner that allows the same for others. FOSS emphasises the logic of non-discrimination to create conditions for free action and thought. The software source code is treated as a form of speech (Raymond, 2001).

FOSS as a new paradigm shift of software development and ownership (Fitzgerald, 2006) is reminiscent of the old *ujamaa* policy of Tanzania. The ideas of *ujamaa* were developed by Nyerere, the first President of Tanzania. To the outside world, *ujamaa* is a form of African socialism, advocated across the continent immediately after its independence. African socialism in Tanzania, however, is unique and outstanding in that it was a deliberate attempt to redefine the Western idea of socialism in an African context, expressing it in an indigenous language, Kiswahili (Tsuruta, 2006). Nyerere (1962) argues that 'Socialism – like democracy – is an attitude of mind. In a socialist society, it is the socialist attitude of mind, and not the rigid adherence to a standard political pattern, which is needed to ensure that the people care for each other's welfare' (Nyerere, 1962, p. 162).

First formulated in an essay by Nyerere published in 1962 (Nyerere, 1962), *ujamaa* was adopted as state policy when the landmark Arusha Declaration was issued in 1967 (Nyerere, 1968). *Ujamaa* derives from *jamaa* (relative or companion), a very familiar word to Kiswahili speakers; thus *ujamaa* can be translated as familyhood. Therefore, the term *ujamaa* does not escape the connotations and associations of kinship, tribal hospitality, and the welfare obligations of the extended family, even when it is used simply to mean modern socialism (Tsuruta, 2006). It is important to note that, in the case of the *ujamaa* policy, there was no socialist ideology copied from the East or the West; rather, an African Socialism was developed.

Nyerere describes the basic principles of this socialism as a society in which all members have equal rights and equal opportunities; in which all people can live at peace with their neighbours without suffering or imposing injustice, being exploited, or exploiting; and in which all people have a gradually increasing basic level of material welfare before any individual lives in luxury (Neyerere, 1968). African socialism did not have the 'benefit' of the Agrarian Revolution or the Industrial Revolution. It did not start from the existence of conflicting 'classes' in society. Nyerere insists that the foundation, and the objective, of African socialism is the extended family. The true African socialist does not look on one class of men as his brethren and another as his natural enemies. He does not form an alliance with the 'brethren' for the

extermination of the non-brethren (Nyerere, 1962).

The *Ujamaa* ideology represents two basic principles of moral economy which are 'the right to subsistence and the norm of reciprocity' (Scott, 1976, p. 167). Nyerere (1968) argues that '*ujamaa* is essentially an attitude of mind, or ethic, based on three key elements: mutual respect, sharing of property, and work' (p. 107). The most important implementation of *ujamaa* in Tanzania was during the years after 1967, with the *Ujamaa Vijijini* (village collectivisation scheme), which aimed at a gradual and later complete transformation of the rural areas into socialist communities so that all political and economic activities were collectively organised (Boesen, Madsen, & Moody, 1977; Nyerere, 1968). From 1968 until 1973, the mobilisation of peasants to set up such communities was a high priority of the government.

Today, Tanzania no longer embraces *Ujamaa*; the policy was formally abandoned in the mid-1980s and some authors argues that *Ujamaa* ideology is now considered by today's Tanzanians as a nostalgic relic of a bygone age, or an outmoded ideology (Tsuruta, 2006). *Ujamaa* failed mainly because of several contradictions. For example, the *ujamaa* village policy was supposed to be based on the initiatives of the farmers themselves because self help and mutual co-operation were the keywords. The role of the government was to merely support such initiatives. However, Scheigman (2003) criticises that the initiatives were taken by the government and hence turned into a top-down implementation process. The top-down approach ended with little or no participation of the people themselves. The policy also failed because equality, respect for human dignity, and the desire to prevent exploitation of man by man are of moral and normative nature (Ngotyana, 1973). There was no incentive to work together, to invest more in agricultural practices, or to increase agricultural production; such changes should be based on social or economic incentives (Ngotyana, 1973).

Despite the failure of the *ujamaa* policy, it was a serious attempt to seek an alternative based on African experience and perceptions, which is not very dissimilar from the community-based development approach of today. When working in the public sector (government-owned establishments), most individuals, including managers, are well aware of *ujamaa;* many still believe it was a correct policy, but its implementation was

problematic. FOSS can be considered another form of *ujamaa* being practiced in the Information and Communication Technologies (ICT).

As discussed in the upcoming sections, FOSS, like *ujamaa,* also advocates sharing, but of software, and makes it a publicly owned good. Software differs from physical goods because copying software to your neighbour (helping your neighbour) is not the same as giving physical goods, where the donor's resources are depleted. Ghosh (1998) uses the economic term 'cooking-pot markets' arguing that it works well in software because software can be copied without losing the original and hence taking out of the pot will not adversely affect other participants. For example, if you own a cow, giving the cow to your neighbour will simply mean you no longer own a cow; by dividing the cow into two pieces, you both end up with no cow. With software, copying software to your neighbour means that now you both have the same resource, the software.

1.7 Organisation of the Thesis

The thesis is organised as follows. Chapter 1 introduces the research domain, objectives, and provides an overview of the adopted theory. Chapter 2 is an overview of the FOSS phenomenon, and Chapter 3 is a presentation of the theoretical perspectives adopted as analytical framework of the empirical material. The framework identifies three gaps sponsor-developer, global developer-local developer and local developer-local user gaps. Chapter 4 is the research approach, which is interpretive approach to participatory action research. Chapter 5 provides a summary of the papers, and a synthesis of the findings in a preliminary analysis. Chapter 6 presents the implications and contributions of the thesis. This presents strategies for bridging the three gaps identified in Chapter 3. Lastly, in Chapter 7, are the concluding remarks.

CHAPTER 2: UNDERSTANDING FREE OPEN SOURCE SOFTWARE

This chapter presents the underlying philosophy of the Free Open Source Software (FOSS) phenomenon and its perspective on intellectual property rights in Sections 2.1 and 2.2, respectively. Section 2.3 provides a detailed description of the FOSS development approach, also referred to as "transformation." Section 2.4 presents economic perspectives that explain the motivations for software developers to freely reveal their innovative software and the source code. Section 2.5 presents more use-focused motivations that lead individuals and organisations to use FOSS products. An idea on stakeholders who develop, fund, and use FOSS products is presented in Section 2.6. Section 2.7 relates the advantages, constraints, and case studies of the FOSS in developing countries. A summary of all sections of this chapter is presented in Section 2.8.

2.1 Definition and Philosophy

2.1.1 Proprietary Software

May (2006) notes that computer software is expensive because it is subject to intellectual property rights. Fortunately there is a cheap alternative – Free/Libre Open Source Software (FOSS). Weber (2003, p. 2) illustrates the difference between proprietary and open source software as follows:

> …when a person purchases a proprietary software he buys a right-to-use license. You can use proprietary software on a computer but only under very specific terms: you cannot reproduce it, modify it, improve it, or redistribute your own version of the software to others. Copyright, licenses, patents, and other legal structures provide a layer of legal protection to this regime, but there is an even more fundamental mechanism that stops you from doing any of these things: proprietary software makers do not release their source code.

While in proprietary software, the source code is the touchstone of the conventional intellectual property regime for computer software, the FOSS process simply inverts this logic; the source code is released along with the software to anyone and everyone who chooses to use it (Weber, 2004). There are two movements that guide the FOSS phenomenon: Free Software Foundation (FSF), formed in 1985, and the Open Source Initiative (OSI), formed in 1998. The FSF argues that the word "free" was intended to

mean free as in "free speech," an intangible right and not a physical good, emphasizing the freedom to distribute software, rather than a freedom from cost (FSF, 2008). Due to its availability, free software has become associated with zero cost, making it seem anti-commercial. In contrast, the OSI use the term open source to eliminate ambiguity, particularly for individuals who perceive free software as anti-commercial (OSI, 2007). Although the two movements disagree on ideological issues, they share a philosophy and a common enemy - the proprietary software. Thus, any software that is qualified by one movement as a FOSS product is done so by the other as well. There are two main terms used to define FOSS: Free Software Definition (FSD) and Open Source Definition (OSD). These definitions are discussed in subsequent sections.

2.1.2 Free Software Definition

The FSD maintains that free software is a matter of liberty, not price. It is a matter of the users' freedom to run, copy, distribute, study, change, and improve the software (Stallman, 2002). As stipulated in the FSD, software users have the right to four kinds of freedom (Stallman, 2002, p.41):

> Freedom 1: The freedom to run the program, for any purpose. Freedom 2: The freedom to study how the program works, and adapt it to your needs (access to the source code is a condition for this). Freedom 3: The freedom to redistribute copies so you can help your neighbour. Freedom 4: The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (Access to the source code is a condition for this.)

Stallman (2002) gives further explanations of the four kinds of freedom, stating that a program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, with or without modifications, either gratis or for a fee, to anyone. Being free to do these things means (among other things) that you do not have to ask or pay for permission. You should also have the freedom to make modifications and use them privately in your own work or play. If you do publish your changes, you should not be required to notify anyone. The freedom to use a program means the freedom for any person or organization to use it on any kind of computer system, for any kind of overall job, without being required to communicate subsequently with the developer or any other specific entity.

2.1.3 Open Source Definition

An alternative definition of FOSS is the Open Source Definition (OSD), which takes into account the distribution of FOSS as well as access to source codes. The OSD defines terms of rights (Perens, 2005) to which a software licence must conform in order to be certified as a FOSS product.

> 1. Free Redistribution: The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.
>
> 2. Source Code: The program must include source code and must allow distribution in source code as well as compiled form.
>
> 3. Derived Works: The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
>
> 4. Integrity of The Author's Source Code: The license may restrict source code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time.
>
> 5. No Discrimination against Persons or Groups: The license must not discriminate against any person or group of persons.
>
> 6. No Discrimination against Fields of Endeavour: The license must not restrict anyone from making use of the program in a specific field of endeavour.
>
> 7. Distribution of License: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
>
> 8. License Must Not be Specific to a Product: The rights attached to the program must not depend on the program's being part of a particular software distribution.
>
> 9. The License Must Not Restrict Other Software: The license must not place restrictions on software distributed along with the licensed software.
>
> 10. No provision of the license may be predicated on any individual technology or style of interface.

Feller and Fitzgerald (2002) underline that for software to qualify as a FOSS product, first, the conformance of a software product's terms of distribution to all criteria of the OSD is *necessary*. Second, conformance of a software product's terms of distribution to all criteria of the OSD, even without actual OSI certification, is *sufficient* for all intents and purposes. Thus, the definition of FOSS helps to qualify software on its conformity as a FOSS product.

The differences between FSF and OSD are explicitly explained in the article 'why free software is better than open source.' Stallman (2002, p.55) unpacks the differences as follows:

> The fundamental difference between the two movements is in their values, their ways of looking at the world. For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one. As one person put it, "Open source is a development methodology; free software is a social movement." For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution.

Despite these fundamental differences, the Free Software and Open Source movements are treated the same in this thesis; focus is on the issues common and useful to both movements.

2.1.4 Philosophy and Values

The definitions of FOSS indicate an understanding that software is an important building block in the information society and that the control of this infrastructure needs to remain accessible to all (Klang, 2005). Stallman (2002, p. 57) argues that 'talking about freedom, about ethical issues, about responsibilities as well as convenience, is asking people to think about things they might rather ignore. This can trigger discomfort and some people may reject the idea for that.' Stallman's stance gives hints that there are ethical values embedded in the FOSS phenomenon. One such ethic is the 'hacker' ethic. In the FOSS community, those who identify themselves as hackers enjoy exploring the details of programmable systems, enjoy programming, and are good at programming quickly (Raymond, 2003).

Raymond (2003) underscores that hacking is characterised by an appropriate application of ingenuity. As presented in the Hackers' Jargon File,[2] ethics common to all hackers maintain that information sharing is a powerful positive good, and that it is an ethical duty of hackers to share their expertise by writing open source codes whenever possible. Also, hackers believe that system-cracking and exploration are ethically fine, provided the cracker commits no theft, vandalism, or breach of confidentiality (Raymond, 2003). Williams (2002) notes that when Richard Stallman

[2] The Jargon File is a Hacker's Dictionary file maintained by Eric Raymond - http://catb.org/jargon/

was working at the MIT, his stance to oppose security measures that required the use of passwords in the computer room was ethically driven in that the entire art of hacking relied on intellectual openness and trust. The use of passwords would have imposed barriers on intellectual openness.

Ideological issues in society have been long recognised. For example, when writing about open society and its enemies, Popper (1945) established that openness has three aspects: ideological, political, and legal. The FOSS phenomenon has a substantial ideological stance. Stallman (2002) criticises copyright laws, saying that they fit well with the printing system industry because it restricts only the mass producers of copies; it does not take freedom away from readers of books (Stallman, 2002, p.45). Stadler (2003) claims that no software is perfect; hence, the notion of free revealing software source code is a way of fixing bugs. Generally, the FSF movement maintains the ideology that society needs freedom to encourage voluntary cooperation in its citizens.

Physical goods and software differ in that while taking a physical good from someone may leave the owner's resources depleted and thus harm the owner, copying software hurts no one. This leads to the argument that software should not have an owner (Stallman, 2002) because the ideas and institutions concerning the property of material objects are about whether it is right to take an object away from someone else. Such ideas do not apply to making a copy of something because copying has no direct effect on the owner (Stallman, 2002). This implies that the general philosophy behind the FOSS phenomenon is to build a society where software is freely copied and modified without any restrictions.

2.2 Intellectual Property Rights

2.2.1 Copyrights

Any intellectual property can be freely revealed even if protected by intellectual property legal mechanisms such as patents, copyrights, or trade secrets, as long as the owners of the protected property decide to reveal it (von Hippel & von Krogh, 2003). Stallman (2002) advises that the simple way to make a software program free is to put

it in the public domain, un-copyrighted. Putting software in the public domain may fall into the hands of someone who can make changes and distribute the results as proprietary software. The solution: whenever software is placed in the public domain, it should be copyrighted. Since copyrights in FOSS products permit all things that are restricted by traditional copyrights, the term "copyleft" is used to denote that the copyright allows anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it (Stallman, 2002, p.89). Lerner and Tyrole (2001, p.821) state that 'open source software should not be confused with shareware (which is freely distributed, but whose source code remains proprietary) and public domain software (which is not licensed and is thus usable by everyone without constraint).' These arguments insist on the importance of licensing software which is a way of applying copyrights to a software application.

2.2.2 Patents

May (2006) argues that the choice between proprietary software and free or open source software is a policy problem that requires urgent attention. The emphasis on continual innovation in FOSS puts it into direct conflict with the ideologies of patenting (Chopra & Dexter, 2008). Stallman (2002) claims that intellectual property laws are a big threat to software development. Copyrights cover the details of expression of a work, but not ideas; patents only cover ideas and the use of ideas. Thus, copyrights cover copying only, while a patent is the absolute monopoly of using an idea (Stallman, 2002). A patent in software is the function of the software that is protected, even if the actual code has been modified sufficiently to avoid copyright infringement (May, 2006, p. 131). Since writing software involves a collection of ideas, patenting ideas would simply make it impossible to write software without committing a violation. Stallman (2002) elaborates further by saying that a product is the result of one idea; hence, a patent on a product is applicable to one idea. However, patenting software obstructs the progress of software development because software is a collection of many ideas; hence, there is the likely chance of infringing upon many patents (Stallman, 2002).

2.2.3 Licenses

For software to be qualified as FOSS, only its license is important (von Hippel & von Krogh, 2002). License is fundamental to the definition of FOSS as a significant marker and required characteristic of Open Source software (Feller & Fitzgerald, 2002). Bonnaccorsci and Rossi (2003, p.1248) insist that 'licenses are the most important institution in the governance structure of Open Source projects.' The FOSS phenomenon 'must not be confused with public domain software, which is unconditionally free and not copyrighted because even users of public domain software may have access to the program source code' (Krishnamurthy, 2003, p.47). Thus, licenses are the key differentiating feature between FOSS and Public Domain Software (Hansen, Köhntopp, & Pfitzmann, 2002; Krishnamurthy, 2003; Lerner & Tirole, 2001).

Lerner and Tirole (2005, p.22) categorise FOSS-based licenses into 'three classes: unrestrictive, restrictive, and highly restrictive licenses.' An *unrestrictive* allows licensees to do anything with their software. Under unrestrictive license, such as a Berkely Software Definition (BSD) license, taking a FOSS product and turning it into proprietary software is allowed. A *restrictive* license requires that when a modified version of the software is distributed, the source code must be made generally available (Lerner & Tirole, 2005). An example of a restrictive license is the Lesser General Public License (LGPL). A *highly restrictive* license restricts licensees from mingling their source codes with software that does not employ such a license. Lerner and Tirole (2005) argue that highly restrictive licenses are sometimes termed "reciprocal" or "viral" provisions because they require even their respective derivative software to be licensed under the same license. A good example of such a license is the General Public License (GPL). The basic concept of the GPL is that software under this license cannot be taken from the public through proprietary modifications (Ackermann, 2003). Three key points in the GPL are (1) software object code distribution must provide access to the source code at no charge; (2) software derivative works fall under GPL; and (3) subsequent licensors cannot change license terms (FSF, 2007; Stallman, 2002).

In any FOSS-based license, Rosen (2005) argues that the following principles should be fulfilled: (a) licensees are free to use FOSS for any purpose. This indicates that an open source licence may not interfere in any way with the use of the software by licensees. (b) Licensees are free to make copies of FOSS and to distribute them without payment of royalties to a licensor. However, this principle does not mean that a licensor cannot sell open source software. It merely says that a licensee need not pay the licensor for additional copies he makes himself, even if those copies are distributed to others. (c) Licensees are free to create derivative works of open source software and to distribute them without payment of royalties to a licensor. This is based on the notion that quality software is built upon earlier software and promotes the progress of science and useful arts. (d) Licensees are free to access and use the source code of open source software. This requires the licensor to make source codes available to licensees upon request at no cost, not necessarily to distribute the code to everyone. Lastly, (e) Licensees are free to combine open source and other software. Open source licenses may not impose conditions or restrictions on other software with which the licensed software is merely combined or distributed.

Perens (2005) insists that the freedom envisioned in FOSS products provides users with the option of providing their own support or the economy of a number of competing support providers. Furthermore, the strength of the FOSS phenomenon is that any programmer can tailor an open source program to specific markets in order to reach new customers (Perens, 2005). This is because people who customise and sell FOSS products are not compelled to pay royalties or licence fees to the original author of the software (Rosen, 2005).

2.3 Transformation

2.3.1 Bazaar

In explaining the software development process (transformation) in FOSS, Weber (2003) argues that the standard way of organising the production of proprietary software has been much like the standard way of building a complex industrial product; there is a formal division of labour that uses proprietary knowledge, guarded

by restrictive intellectual property rights, enclosed within a corporate hierarchy, to guide and govern the process. However, FOSS is organised differently in that 'a large and complex system of code can be built, maintained, developed, and extended in a non-proprietary setting where many developers work in a highly parallel, relatively unstructured way and without direct monetary compensation' (Weber, 2003, p. 1). The generic FOSS development process is characterised as follows (Feller & Fitzgerald, 2002, p. 84):

> [It] is parallel, rather than linear; involves large communities of globally distributed developers; utilizes truly independent peer review; provides prompt feedback to users and developers contributions; includes the participation of highly talented, highly motivated developers; includes increased levels of user involvement; and makes use of extremely rapid release schedules.

This view on the way FOSS development is organised is also shared by other scholars (Cook & Horobin, 2006; Raymond, 2001; Scacchi, Feller, Fitzgerald, Hissam, & Lakhani, 2006). Raymond (2001) uses the Cathedral and the Bazaar metaphor to represent the transformation of FOSS, arguing that proprietary software production is like the carefully planned building of a cathedral, while FOSS production is a chaotic interaction of participants analogous to an oriental bazaar. This hints at a major difference between the two types of software development: strong powerful management on one side (i.e., proprietary) and loosely related developers and users organised in several thousand seemingly interdependent projects on the other side (i.e., open source). The central assumption in the bazaar model is that several talented developers can successfully work on the same piece of code in parallel without much coordination, and eventually they will fix a software bug. Raymond (2001) simplifies the process of resolving software bugs: given enough eyeballs, all bugs are shallow.

In addition to the parallel development approach of FOSS, there is much redundancy and parallel development. That is, more than one developer works on the same module of software. Theoretically, this characteristic of FOSS development improves the quality of products by allowing multiple solutions to the same problem to compete with each other (Weerawarana & Weeratunga, 2004). The parallel and redundancy development feature in FOSS is peculiar since it appears to have side-stepped Brooks' Law, which states that adding manpower to a delayed software project delays it

further (Brooks, 1995). That is, FOSS projects leverage the advantages of large numbers of software developers (Weber, 2004).

The FOSS life cycle differs from the traditional one of planning, analysis, design, and implementation. Studies indicate that the FOSS development life cycle is located in the implementation phase, which features submission of source codes; review of source codes; pre-commit testing of contributed codes; development release; parallel debugging; and production release (Feller & Fitzgerald, 2002; Weerawarana & Weeratunga, 2004; Wichmann, 2002). In FOSS, planning, analysis, and design phases are largely undertaken by the project initiator and therefore may not be part of the FOSS development cycle. In the other words, all FOSS projects start from the cathedral phase before they become a bazaar.

2.3.2 Modularity

The specific characteristics of FOSS revolve around the programming languages used, paradigms favoured by the FOSS development community, the strong modularity characteristic of FOSS products, and the relative complexity of FOSS products (Feller & Fitzgerald, 2002). In their study of several open source projects, Feller and Fitzgerald (2002) conclude that the most common programming language used is C, which is favoured for most low-level systems programming, but there is an increasing trend of using Object-Oriented programming languages such as C++ and JAVA, which support component-based reuse. Other scholars have noted that 'the object-orientation (OO) paradigm has increasingly been integrated in computer science education as well as in a variety of professional practices' (Kaasbøll, Fjuk, Karahasanoc & Groven, 2006, p. 205). Apart from low-level languages and OO, FOSS developers are also making use of a bootstrapping approach (Ciborra, 2000). Bootstrapping in FOSS is the tendency of using FOSS products, especially scripting languages and source code management tools, to develop more FOSS products. Today, Internet applications are on the lead and scripting languages like PHP, JAVA, and Perl are widely used to develop those Internet applications.

Weerawarana and Weeratunga (2004) observe that parallel development is a key aspect

of the FOSS process and is the result of the highly modular nature of many FOSS products. Individuals or small groups of developers in a FOSS project work on one aspect of a large system while others work on another aspect of the system. Thus, each member of a FOSS project does not need to know everything about the project; instead, each concentrates on a particular part of the project (Feller & Fitzgerald, 2002). Thus, modular development is essential for production of this type of software to be sustained (Câmara & Fonseca, 2007). Lack of modularity prevents a large community of developers from engaging in the project (Sharman & Yassine, 2004). The modularity in FOSS products also supports re-use of the software source code. An investigation of many FOSS products reveals that most of the lines of source codes in the majority of open source projects are taken from the commons of other open source software projects (von Krogh et al., 2005).

According to Câmara and Fonseca (2007), the essential properties of FOSS are the degree of shared *conceptualisation* and the degree of *modularity*. Because of geographical distance between FOSS developers who work on the same piece of software, a shared conceptual view of the design of software is of particular importance. Two conditions are necessary for shared conceptualisation (Câmara & Fonseca, 2007, p.124): the post-mature perspective and the standards-led perspective. In the post-mature perspective, a private company develops a software product, for which it holds the intellectual property rights. As the product becomes popular, its functionality and conceptual model becomes well-established and part of the "public commons." The popularity and usability of the software motivates other institutions to develop a public domain equivalent, as in the Open Office suite. In the standards-led perspective, standards consolidate a technology and allow compatible solutions from different producers to compete in the marketplace. Câmara and Fonseca (2007) presented an example of the Portable Operating System Interface for UNIX (POSIX), the popular multitasking, multiuser operating system standard for operating systems, which has guided Linux.

This view of shared conceptualisation in FOSS is also noted by other authors, who note that the bulk of FOSS products belong to software engineering domains, where the general requirements and the architectural reference models are well-known and

accepted (Feller & Fitzgerald, 2002). Since many FOSS projects, especially the infrastructure-based applications (operating system, database management systems, and server side applications), are implementations of complex but well-understood specifications, modularity facilitates the distributed, parallel process, and the software's complexity represents an attractive challenge for the FOSS programmer (Feller & Fitzgerald, 2002). This is to say, because of modular architectures and decentralised parallel development in FOSS projects, geographically distributed developers find it easier to contribute to these projects.

2.3.3 Tools

One more compelling feature of the FOSS development process is the tools used. Since FOSS development is described as an oriental bazaar (Raymond, 2001), no one knows the source of feedback and/or contributors. In this case, configuration management tools are very important. According to Fogel (2006), Concurrent Versions Systems (CVS) is the most common tool used for configuration management. CVS simplifies the process of incorporating changes to the repository and gives anonymous read-only access to a project's source code repository (Fogel, 2006). According to Feller and Fitzgerald (2002), developers are working in parallel with no formal division of labour, and the use of CVS enables them to download the source codes with a single command.

2.3.4 Communities

The structure of FOSS communities could be analysed along the dimensions of division of labour, co-ordination mechanisms, distribution of decision-making authority, and decision-making boundaries (Nohria, 1995). The norm is modesty and self-deprecation on the part of developers because contributions from others drive the entire FOSS model (Wichmann, 2002). Most FOSS activities take place online and thus 'the Internet is the primary enabler of the FOSS development and distribution process, making it possible for widely distributed groups to share ideas and software extremely quickly at negligible cost' (Feller & Fitzgerald, 2002, p. 126). E-mail, mailing lists, discussion

forums, and collaborative websites are among the tools used. Thus, FOSS development discussions are conducted on e-mail, mailing/forums, and websites.

Feller and Fitzgerald (2002) recognise that FOSS development can take place 'offline,' arguing that 'in many ways, face-to-face community is re-emerging, and although much FOSS activity occurs in virtual spaces, there is an increasing amount of "real" world activity as well' (p. 129). They argue that there is an increasing number of co-located teams for FOSS development, especially in large companies such as Red Hat and IBM (Feller & Fitzgerald, 2002). For them, while online communities are bounded by interest, offline communities are often geographically bounded and are the result of seeking closed-look control of the development process (Feller & Fitzgerald, 2002). This implies that co-located development of FOSS is not due to a lack of resource and skills.

Again, although FOSS development is represented as the chaotic activity in an Oriental bazaar (Raymond, 2001), several studies show that many FOSS projects somehow have strict controlling and coordinating mechanisms. The study by Bonaccorsi and Rossi (2003) concludes that in a FOSS project there is a well-respected leader or core development group where 10% of developers write more than 70% of the software source code. A study of the Apache project reveals that 15 developers contribute almost 90% of the code (Mockus, Fielding, & Herbsleb, 2000). Another study concludes that 10% of 2,784 developers make up almost three-quarters of the software source code of a specific FOSS project (Ghosh et al., 2002). This suggests that in most FOSS projects, the decision maker with the final say about project development relies on a small number of developers who are team members with technical skills, sometimes referred to as 'benevolent dictators' (Bezroukov, 1999).

2.4 Economics

Lerner and Tirole (2000) asked why a person would take the time to write complicated software programs for free given economic theories. They suggest that FOSS brings developers specific, tangible, and favourable economic benefits that are sensible and quite lucrative. Sauer (2007) study offers similar conclusion, though it suggest more data are need to explore the issue. There are both immediate and delayed benefits. Immediate benefits include monetary compensation and the opportunity to fix bugs

and customise a program for the developer's own use. The delayed benefits include a signalling incentive (which promises future jobs) and ego gratification (peer recognition) (Lerner & Tirole, 2000).

FOSS is a peculiar way of realising software products because the software source codes (the core innovation of the software) are freely revealed so that others may use them, learn them, and perhaps improve them (von Hippel & von Krogh, 2006). Through the discussion on copyrights and patents, it is clearly demonstrated that FOSS goes against the traditional ideas of intellectual property rights on software. The tendency to spend countless hours developing valuable software, only to give both it and software codes away, can be better conceptualised through economic perspectives.

2.4.1 Private Investments

Traditionally, there are two types of investment models of innovation: *private* and *collective* (von Hippel & von Krogh, 2006). In the private investment model of innovation, individuals or organisations invest in the development innovation if they foresee private rewards. In this model of innovation, 'manufacturers rather than product users have traditionally been considered the most logical private developers of innovative products and services because private financial incentives to innovate seem to be higher for manufacturers. This is because a manufacturer has the opportunity to sell to an entire market place of users' (von Hippel & von Krogh, 2003, p.214). Since innovation requires a significant investment, manufacturers want to protect their innovations through the intellectual property rights structures such as copyrights and patents. Now, when private innovators reveal their development without compensation, this action simply represents a loss of potential returns, something which should be avoided. In most cases, FOSS initiatives are funded by individuals and organisations and are thus compatible with the private investment mode of innovation. However, the FOSS phenomenon deviates from the private investment model of innovation by going against intellectual property rights. Furthermore, the software developers, who are also the main users, are the pioneers of the innovation process, a situation which is the opposite of the traditional manufacturer and user

relationship. In addition, although innovations are funded by individuals, the resulting products are revealed freely.

2.4.2 Collective Investments

The FOSS phenomenon is conceptualised as a novelty technology for producing software which represents a new model of production in the form of commons based peer production and is a critique of existing laws, contracts, and business practices (Kelty, 2001). This model assumes that 'innovators relinquish control of knowledge or other assets they have developed to a project and so make them a *public good* in order to avoid social loss associated with the restricted access to knowledge of the private investment model' (von Hippel & von Krogh, 2006, p.302). FOSS as a public good has the features which are core characteristics associated with classical physical public goods (Marwell & Oliver, 1993): (a) software can be copied and used by many people simultaneously; use by one individual does not limit usage by another. (b) Each user has his or her individual copy with the right to modify and distribute. (c) Most FOSS licences, e.g., the GPL, do not allow any single user to take away usage rights from other users. (d) Like any public good, FOSS has the free rider phenomena, meaning that many individuals acquire and use the software without contributing to the respective software project. Furthermore, FOSS is developed through collective action by numerous individuals

Fulk and others (1996) put forward that *communality* and *connectivity* are additional characteristics of public goods with interactive communication systems nurture; software applications are an interactive communication system. FOSS projects use webpages, frequently asked questions, and bug tracks to share knowledge about the project. Since these documents are freely available online, they fulfil the communal requirements for an interactive public good. Connectivity refers to the ability for a member to communicate with any other member (Fulk et al., 1996). E-mail lists and discussion forums fulfil this capability; as a result, there is very little cost to communicate directly to any member of a FOSS project.

Since public goods exhibit free rider characteristics, where a member of society can use

a public good without contributing back, this model of innovation suffers from the problem of recruiting and motivating contributors. To address the recruitment problem, selective punishment or incentives are necessary. However, selective punishments and reward mechanisms work well in smaller groups of society (Friedmann & McAdam, 1992). An alternative solution is to state the goal of the project clearly so that it attracts potential contributors (Taylor & Singleton, 1993). Thus, the collective investment model of innovation recommends a small group of society members who lead innovation practices. However, the FOSS phenomenon is characterised by many geographically distributed developers (Feller & Fitzgerald, 2002). Benkler and Nissenbaum (2006) add that FOSS is a good example of common-based peer production. This again disputes the notion that a small sect of society is the best way of organising a collective investment model of innovation. The goals for developing software are very diverse; some are based on technology, some on ideology. For example, Stallman (1999, p.64) argues that GNU software was developed in order to have a complete free open system. This is contrary to other proponents of FOSS, who claim that good software starts by scratching a developer's personal itch (Raymond, 2001), suggesting that FOSS products are the result of impulses. Thus, although FOSS products fit well as interactive public goods, the process of developing them deviates from the collective investment because of the large number of developers involved and the diversity of various goals and motivations for participating in FOSS projects.

2.4.3 Private-Collective Investment

Free revealing has been practiced in many cases. One attempt to study why innovators decide to reveal their innovation freely is the study of von Hippel and von Krogh (2006), which concludes that free revealing is the best practical option available when others know something close to their secret, when profits from patenting are low, and when incentives for free revealing are positive. These insights lead to a hybrid investment model of innovation labelled the "private-collective investment model of innovation" (von Hippel & von Krogh, 2003; 2006). The model gives explanations for the free revealing practice in FOSS. Although there is a higher rate of private

investment, resulting software is freely revealed because an individual's benefits in FOSS have been tied to participation in communities surrounding the project, which to a large extent outweighs individual rewards from the collective good being jointly developed (von Hippel & von Krogh, 2006).

2.4.4 Business

The FOSS-oriented business models rely on shifting the commercial value away from the actual products to services such as systems integration, user support, tutorials, and software documentation (Riehle, 2007). There are two varieties of FOSS business models: (a) distribution and retail of FOSS products and (b) offering of FOSS-related services (Ghosh et al., 2002; Riehle, 2007; Schmitz, 2001; Weerawarana & Weeratunga, 2004; Wichmann, 2002). Individuals and companies who develop FOSS could give away their software products and concentrate on making profits with related services and supports. That is, their business is based on the knowledge gained in developing the software and their popularity as the original author. With this model of not focusing on the selling price of the software but on its related products, known revenue related activities include: *Software Distribution* – even if users can download a particular software, distributors make life easy for users who are willing to pay a small amount for comfortable access to the software. *User Support* – users can subscribe to support services such as disaster recovery, backups, training, and bug fixing. *Information* – there is a business for publishing books, magazines, news tutorials, and software manuals on respective FOSS products.

On the consumer side, the total cost of ownership of software may cover not only the selling price of the software, but also any cost that is incurred by the decision to install the software in the organisation (Evers, 2000; Samuelson, 2006). While in the proprietary software market nothing is free, in FOSS it is possible to purchase and update software free of charge. The total cost of ownership (TCO) in FOSS is in most cases less than that of proprietary software. However, users of FOSS products are likely to pay for the following activities: *purchase* - the selling price of the software; *system setup* – a budget for additional hardware and software may be required to

facilitate smooth running of the software; *user training* - training users in additional skills is costly; *user support* - in the event the training does not deliver all required skills to users, additional support costs are required; *updates* - after a system is put into use, software updates might be required to fix bugs and introduce new features, and the users may be charged.

Riehle (2007) argues that the price of proprietary software does not directly depend on the actual cost incurred to develop, maintain, and provide the software. This is because while the cost of developing a FOSS product is the same as in proprietary software, in FOSS, the total cost of software development is a result of shared costs from different contributing developers (Riehle, 2007). Since different developers have different development costs depending on their contribution, those with low investments sell the resulting FOSS product at a low cost. In a FOSS product, usually there are no fees associated with using the system, and there is competition between service providers to provide the best and most cost-effective support. Service providers are more readily available because the source code enables those with knowledge on how the software was built to identify exactly what processing is performed by the system (Weber, 2004; Weerawarana & Weeratunga, 2004).

There are many critiques of this novelty claims about FOSS (Fitzgerald, 2005; Glass, 2005). Some authors suspect that trade secrets can actually be hidden and used to generate large amounts of income as if royalty payments were allowed (Chiao, 2003). Original developers can hide trade secrets by providing partial software documentation, making slow software downloads, and lengthening the time needed to publish links where the software can be downloaded (Chiao, 2003). Whether deliberate or unintentional, FOSS products are not readily available to average users. Weerawarana and Weeratunga (2004, p. 24) state why: 'due to the focus on system function and code quality rather than ease-of-use, classic FOSS is typically not suitable for use by average developers and most certainly not by average users. Distributors address this gap by packaging quality FOSS in a manner that is much more widely accessible to the user community.'

2.5 Motivations

Although the private-collective investment model of innovation offers good explanations for why FOSS practices enable new knowledge to be created by private funding and then offered freely to the public (von Hippel & von Krogh, 2003), some scholars have attempted to find the motivation behind why thousands of software developers contribute freely to the public good (see e.g., Lerner & Tirole, 2000). Dedrick and West (2003), who conducted a study on the adoption of Linux in firms, state that major motivations for firms to adopt FOSS products include low total cost of ownership, compatibility with current technologies and skills, and the availability of external technological resources. Other authors cite creativity and reputation as incentives for individuals to participate in FOSS development (Bezroukov, 1999; Raymond, 2001). Wichmann (2002) identifies four motivations for firms to participate in FOSS related activities: standardisation to overcome the ghost of the Unix wars; reduction of costs through low-cost open source components; compatibility; and strategic consideration, for example, to release software as open source to weaken a competitor. Furthermore, a growing number of programmers develop FOSS as part of their primary, paying jobs (Feller & Fitzgerald, 2002). In a survey on the use of FOSS in Europe, it was found that most FOSS developers receive some monetary reward for their work (Ghosh et al., 2002). The study by Ghosh and colleagues (2002) put the numbers as follows: 71% of FOSS developers seek to increase their skills, while 43% of the developers want to gain a reputation or increase their job opportunities. Feller and Fitzgerald (2002) propose three broad motivational areas behind participation in FOSS: technological, economic, and socio-political. The motivation factors in each category are summarised in Table 2.1.

Table 2.1: Motivations for FOSS

	Micro Level (Individual)	Macro Level (Organisation)
Technological Motivations	To meet a personal technological need	To address the software crisis – particularly poor quality
	To exploit the efficiency of peer review, etc.	To share tedious development tasks (testing, documentation) with users
	To work with 'leading-edge' technology	To leverage FOSS community for research and development
		To promote innovation

	Micro Level (Individual)	Macro Level (Organisation)
		To ensure transparency of the application
Economic Motivations	To gain future career benefits	To exploit investors' infatuation with FOSS
	To improve coding skills	To embrace the paradigm shift from software as a commodity industry to a consumer-driven service model
	To 'strike it rich' through stock options, etc.	To raise mind share and strengthen brand
	Low opportunity cost – nothing to lose	To exploit indirect revenues from selling related products and services
		To make software affordable in developing countries
		To cut costs – cheaper platform than proprietary alternative
Socio-political Motivations	Ego gratification and signalling incentives	Social movements require an enemy – e.g., Microsoft
	Intrinsic motivation of coding	Overcomes 'digital divide'
	Sense of community	Ideology – software must be free
	Altruism	Model for wider domain – future model for work

Table adopted from Feller & Fitzgerald (2002, p. 139)

Similar findings were identified in Rossi (2006). Lakhani and Wolf (2005, p. 18) contend that 'developers feel a high degree of personal sense of creativity with regard to their FOSS projects. That sense of creativity in projects is underscored by three main drivers: (1) enjoyment-related intrinsic motivations, (2) extrinsic motivations in the form of payment, and (3) obligation/community-related intrinsic motivations.' According to this study, use of the created output is one of the three most important incentives inducing developers to innovate. Von Hippel (2005) insists that many innovators have a use-incentive for innovating in FOSS projects. For example, contributors of source codes to open source projects agree that facilitating their own work is a motivating factor (Hertel, Niedner, & Herrmann, 2003).

2.6 Stakeholders

Four major groups of stakeholders were identified to be 'developer communities, user communities, commercial, and non commercial organizations' (Feller & Fitzgerald, 2002, p. 107). However, these groups are far from mutually exclusive, in that many FOSS users are also developers and many developers are users. Also, commercial and

non-commercial organisations both develop and use FOSS products. Feller and Fitzgerald (2002) nevertheless identify distinguishing characteristics of each group and the way each group acts at different times as clients (beneficiaries), actors (agents of change), and owners (decision makers).

Table 2.2: FOSS Stakeholder Groups

	as CLIENT	as ACTOR	as OWNER
Developers	Regularly use FOSS products to support development	Act as the main implementers of changes in systems, both in proactive and reactive mode	Exhibit prime concern for the systems direction, but do not necessarily possess the power to terminate the system
Users	Both directly and indirectly use FOSS products	Can use FOSS as a black box, or actually make changes. Can also effect change through bug reporting, etc.	Have as much claim to ownership (not authorship) as the creator of the software
Companies	Have been the most enthusiastic adopters of FOSS, and in many cases showcase their use of FOSS products	Act as both implementers and patrons of change	Assert control over direction of branded distributions, but not necessarily over actual product
Non-profit organisations	Use FOSS products in the same way as companies	Organise efforts of developers	Often assert highest level of control over direction and future of projects

(adopted from Feller & Fitzgerald, 2002, p. 124) The shaded cells indicate the primary role(s) played by the stakeholder group.

The structuration analysis of stakeholder groups sheds light on the sustainability strategies of FOSS initiatives. For example, companies play roles both as 'Client' and as 'Actor,' which means there is a need to strengthen companies (especially private companies) in order to produce and support FOSS products. Underperformance of the private sector hampers the development and support of FOSS. Non-profit organisations are mostly the owners of FOSS products. This group of stakeholders includes universities and non-government organisations (NGOs).

2.7 FOSS in Developing Countries

2.7.1 Motivations

The European information society reports that, open software in many cases are equivalent to - or better than - commercial products. Therefore, procurement of software shall evaluate open software as well as commercial solutions, to provide better competition in the market (Europa, 2003). Developing countries and their donor partners are urged to review policies for procurement of computer software to ensure that options for using low-cost FOSS products are properly considered and their costs and benefits carefully evaluated (Barton et al., 2002). Three factors stand out when asking why developing countries should choose FOSS: cost, the anti-piracy campaign, and security concerns (Câmara & Fonseca, 2007; May, 2006; Noronha, 2003; Weber, 2003; Weerawarana & Weeratunga, 2004). The dominant factor is the lower cost. It is true that a large number of users in developing countries do not, and, more importantly, cannot really pay for software because of the high price of proprietary software compared with the average incomes (May, 2006). For example, it is estimated that sub-Saharan African countries spend US$ 24 billion each year on proprietary software (FOSSFA, 2004). Not only does FOSS reduce software licensing costs but it also fosters indigenous technological development by giving access to the source code of software products (Câmara & Fonseca, 2007; UN, 2004; Weber, 2003; Weerawarana & Weeratunga, 2004).

Other reasons for the adoption of FOSS in developing countries include avoiding being locked in to proprietary software (UN, 2004), advancing knowledge more quickly, and helping to set up an information economy (Weerawarana & Weeratunga, 2004). In the case of education in computer sciences, FOSS provides unrestricted access to the source code, an environment of unlimited experimentation and collaboration, and interaction with a community of programmers, coders, and users around the world (Noronha, 2003).

A UK based Commission on Intellectual Property Rights advises that developing countries criticise and boycott proprietary software licences. The commission argues

that

> The Internet users in the developing nations should be entitled to "fair use" rights such as making and distributing printed copies from electronic sources in reasonable numbers for educational and research purposes, and using reasonable excerpts in commentary and criticism. Where suppliers of digital information or software attempt to restrict "fair use" rights by contract provisions associated with the distribution of digital material, the relevant contract provision may be treated as void. Where the same restriction is attempted through technological means, measures to defeat the technological means of protection in such circumstances should not be regarded as illegal. (Barton, 2002, p.109)

In particular, says the commission, developing countries should allow their citizens to circumvent copyright protection mechanisms and should not follow the example of the US and the EU by enacting laws that ban such practices (Loney, 2002). In arguing why Peru should consider FOSS over proprietary software, a member of Congress in Peru said

> To guarantee free access by citizens to public information, it is indispensable that the encoding of data not be tied to a single provider; the use of standard and open formats guarantees free access; to guarantee the permanence of public data, the usability and maintenance of the software should not depend on the goodwill of suppliers or on monopoly conditions imposed by them; and to guarantee national security, the state must be able to rely on systems without elements controlled from a distance. Systems with open-source codes allow the state and citizens to inspect the codes themselves and check for back doors and spyware. (Villanueva, 2002)

This demonstrates that countries have been keen to disengage from their reliance on a single IT supplier, who may not be interested in the country's ICT strategy, and to avoid a supplier lock-in situation. Weerawarana and Weeratunga (2004, p. 28) conclude that 'with the use of FOSS, support and maintenance can be freely contracted out to a range of suppliers competing on quality and low cost for installation, enabling, support, and maintenance. Maintenance is further replicable without incurring large costs, since modifications to the source code are also free.' As Berry and Moss (2006) argue, the discourse and practice of non-proprietary software contribute to opening-up and democratising e-government by protecting and extending transparency and accountability in e-governments and by allowing technology to be shaped by citizens and associations, as well as by administrators and private interests.

2.7.2 Constraints

There are many challenging issues regarding the implementation of FOSS in developing countries. Fitzgerald (2006) presents a long list of constraining issues and argues, essentially, that FOSS as an ideology needs to be supported by alternative organisational forms and business models that would allow effective implementation, and policies need to be established to enforce FOSS-based licences. There is limited technical capacity to deal with FOSS development and support (Fitzgerald, 2006); the situation is much worse when dealing with applications for health and education information systems. Furthermore, the capacity of universities and technical institutions in developing countries needs to be strengthened in order to improve the ability of IT professionals to deal with FOSS technologies (Fitzgerald, 2006). Kshetri (2004) studied the macro and micro economics on choosing Linux in developing countries and found that the microeconomics factors include ownership, effective use, learning/switching, and compatibility. The macroeconomics factors are national security and enforcement of intellectual property laws. However, the negative effects of choosing Linux in developing countries include 'lack of supports to deal with security vulnerabilities, costs of supporting custom changes can escalate dramatically over time, higher learning and switching costs, and incompatibility between business partners' technologies' (Kshetri, 2004, p. 76).

Weerawarana and Weeratunga (2004, p.35) propose that in order for FOSS initiatives to proceed, certain IT infrastructural and skills conditions need to be met. These requirements include: intellectual property law framework and enforcement, low cost and widely available Internet access, educational infrastructure, freedom of information, skilled English speaking developers, and a trained developer pool (Weerawarana & Weeratunga, 2004). Their argument is that without working intellectual property laws, the high rate of pirating proprietary software would reduce its price and devalue FOSS products. The concept of low cost would not be seen as a significant motivation for going FOSS. As FOSS development occurs primarily via the Internet, lack of high-bandwidth would severely limit the ability of individuals and companies to participate in FOSS projects (Weerawarana & Weeratunga, 2004).

In Tanzania the most economical internet connection for home user and small business is 512kbps uplink and 2048kbps downlink (Sheriff, 2007), which is basically not sufficient for downloading large files. At the time of writing, the list price for this cheapest internet connection for home and small offices in Tanzania is TZS 95 per megabyte (TTCL, 2007). This implies that to download one CD of 600MB it costs TZS 57,000/= (~USD 48). However, the average net pay salary for IT professionals (at the time of writing) is TZS 250,000 (~USD 210) and thus the cost for downloading a CD is at 22.8% of their salary scale. It is obvious that internet access is expensive for individuals to invest on their own in order to develop a public good, the FOSS products.

In arguing for education infrastructure, Weerawarana and Weeratunga (2004) highlight that IT education infrastructure must be widely disseminated if FOSS adoption should take place. Furthermore, higher level institutions that teach software development are also critical. A culture of learning and further development of the workforce would help with faster and wider FOSS adoption (Weerawarana & Weeratunga, 2004). In terms of a trained developer pool and English-skilled developers, the authors note that success in FOSS development comes from having skilled developers and that English undoubtedly remains the *lingua franca* of computing. These arguments clearly unpack the challenges developing countries face in making FOSS implementation possible. However, there are some proposed strategies for approaching FOSS in developing countries.

2.7.3 Strategies

Weber (2003, p.13) argues that 'the vast majority of open source projects involve a small number of developers. These projects typically depend on intensive communications and the persuasiveness of the "de facto" project leader to coordinate the work of the group.' According to Weerawarana and Weeratunga (2004), this is because 'what matters is not size (in terms of the number of software engineers); instead, what matters is that there is a dedicated, highly skilled and fully committed team of developers to contribute to and lead key open source projects' (p. 95).

However, research on FOSS in developing countries suggests that the view of FOSS as a product of a team of committed individuals is not realistic (Câmara & Fonseca, 2007). Large, collaborative, networked teams are responsible for a small number of FOSS products. Câmara and Fonseca (2007) conclude that FOSS in developing countries needs strong and wise government policies to be successful. It requires a combination of institutional vision, qualified personnel, strong links to the user community, and government-funding to be viable. The high level of expertise required to develop and maintain FOSS projects requires that policy makers provide for significant investments in human resources (Câmara & Fonseca, 2007).

Câmara and Fonseca (2007, p.125) delineate four types of FOSS development: (a) high shared conceptualisation, high modularity (the high–high case), which represents community FOSS products such as Linux; (b) high shared conceptualisation, low modularity (the high-low case), which represents corporate-led products such as databases, web servers, and office automation tools; (c) low shared conceptualisation, high modularity (the low-high case), which represents the academic-led projects; and (d) low shared conceptualisation, low modularity (the low–low case), which is the case for innovation-led projects. They argue that developing countries could handle the high-high case, for these systems have a sustainable community. In the high-high case, such as Linux operating systems, developing countries need to invest in capacity building, documentation, and user training in order to increase the chances of successful adoption (Câmara & Fonseca, 2007).

However, FOSS products in the high-low case are associated with private companies; hence, the users may become dependent on the private company. However, the authors propose that governments or government agencies should actively take part as stakeholders of such projects. When dealing with the low-high case, the authors argue, governments of developing countries would need to invest significantly in human resources; and when dealing with the low-low case, governments should ensure that locally developed FOSS products have enough support to become more sustainable (Câmara & Fonseca, 2007).

2.8 Conclusion

This chapter delineated the FOSS phenomenon. FOSS is different from proprietary software in terms of its "copyleft" approach to intellectual property rights, its bazaar model of software development, and its free revealing economic practice. Specifically, this thesis highlighted the concepts discussed here to reflect the reality in Tanzanian context, as indicated in Table 2.3.

One of the Ujamaa policy assertions is in order to develop; we need four things in place: people, land, right politics, and right leadership (Nyerere, 1968). In this Ujamaa's economic equation, we have few options to manoeuvre on people and land. However, we can improve this equation dramatically by having the right politics and the right leadership. In the FOSS phenomenon, project organisation and leadership are important elements that could foster the realisation of FOSS-based projects. The more we implement the right project organisation according to the context of a developing country, the more we can increase participation of various relevant stakeholders and overcome constraining issues in FOSS development.

Table 2.3: Core FOSS Concepts Informing the Thesis

FOSS Concept	Description and proposed extensions
Licenses	☑ In order for a software to qualify as a FOSS product, it must be flagged a license which fulfils the requirements stipulated in the Open Source Definition or the Free Software Definition. ☑ A FOSS license requires that in order for users to acquire the kinds of freedom envisioned, software must be distributed with its source code. ☑ Selling a FOSS product is acceptable as long as the software is distributed with its source codes. ☑ **Proposal:** license for software as a service delivery. In information systems, users are non-programmers and they are interested in their data in the information systems. A kind of freedom that argues for the data is more important than having access to the source codes.
Economics	☑ FOSS fall under the private-collective investment model of innovation in that individuals fund the development privately but release their innovations in the public domain. ☑ Like the *Ujamaa* policy, which advocates equal rights, equal opportunities, and the economy of reciprocity, a FOSS license argues no discrimination against persons and can be used in any human endeavour. ☑ **Proposal:** In a context where there is compatible political history, like the Ujamaa policy, the FOSS phenomenon could have immediate positive reception.

FOSS Concept	Description and proposed extensions
Transformation	☑ FOSS development takes place online (internet is the enabler) ☑ FOSS development takes place off-line when there is a need to control the process of development, as in large organisations. ☑ CVS and Object-Oriented technologies (which support modularity) are the appropriate technologies in FOS development. ☑ **Proposal:** co-located development. In a developing country like Tanzania, internet is a limited and expensive resource. Moreover, competence with object-oriented is hard to find, and most developers do not own their own working tools such as computers.
Stakeholders	☑ Identified stakeholders are users, developers, companies, and non-profit organisations. ☑ **Proposal:** donors and governments. In a developing country like Tanzania, donors and governments are the main source of funds and the most likely customers of FOSS.

As my study focuses on the use of FOSS applications in information systems, "transformation" and "stakeholders" are relevant key concepts. Transformation related perspectives focus on how to approach open source as a product as well as a process of realising products. Likewise, the stakeholders' discussion informs relevant participants and their roles in open source development. In order to make FOSS development work in the information systems context of a developing country like Tanzania, focus should be on both improving the process of developing FOSS and increasing participation from various stakeholders. In other words, we need to re-organise the FOSS development in order to encourage people to participate.

CHAPTER 3: INFORMATION SYSTEMS IN DEVELOPING COUNTRIES

This chapter consists of five sections. The first section presents the 'design-reality' archetypal situations attributed to the cause of failure of ICT initiatives in developing countries. In the second section is a technological determinist perspective, which assumes that technology is passed on in a neutral society. This technological deterministic perspective fails to account for the influence of society while introducing technological change. The third section introduces ANT, which helps to inform progressive and degenerative networks formed by human and non-human actors. In contrast to the technological deterministic perspective, ANT recognises the roles that can be played by both human and non-human actors. The fourth section, technology translation, presents a more refined perspective: the introduction of technological change is affected by both the technology characteristics and the process of introducing change. The last section is an attempt to synthesise the discussed theoretical perspectives as a proposal for the theoretical framework of the thesis.

3.1 The Design – Reality Gaps

The importance of ICTs in developing countries has been emphasised. Castells (1998) argues for using ICT to foster knowledge and information society. 'The ultimate objective is a knowledge and information society, one with the ability, capacity, and skills to generate and capture new knowledge and to effectively access, absorb, and use information, data, and knowledge with the support of ICTs' (Castells, 1998, p.92). The argument is that ICT can allow developing countries to leapfrog traditional problems of development like poverty, illiteracy, disease, unemployment, hunger, corruption, and social inequality (Keniston, 2002; Musa et al., 2005). Generally, ICT is seen as an enabler for knowledge revolution because ICT is an effective tool for creating, disseminating, storing, and managing information.

However, there are many challenges faced by developing countries in implementing ICTs in information systems. The evident picture of large failure cases of ICT initiatives (Bhatnagar, 2000; Bhatnagar & Bjørn-Andersen, 1990; Heeks, 2002; 2003) demonstrates how challenging it is to implement ICTs in the context of developing countries. The

Ciborra and Nevarra (2005) study on e-government implementation in Jordan concludes that it is difficult to implement ICTs in developing countries because of the characteristics of the local administration, the socio-economic context, and the dynamics of the technological infrastructure. To explain the underlying causes, Heeks (2003) argues that there are gaps between information systems design and its reality of use known as 'design – reality gaps'. Thus, even if developing countries turn to FOSS products, the design reality gaps make sense because FOSS implementation in information systems is nothing but a technological change initiative riddled by socio-technical issues. However, the design-reality gaps vary from technology to technology due to technological characteristics and its implementation strategies.

Heeks (2003) refers to three archetypal situations in which failures are likely to occur. Those situations are *hard – soft* gaps, *private – public* gaps, and *country context* gaps. Although Heeks' study focuses on e-government initiatives, the design-reality gaps apply to these case studies, which took place in the public sector and involved the implementation of ICTs in information systems of a developing country, Tanzania.

The hard-soft gaps address the difference between the notion on ICT in terms of machinery and engineering, rationality and objectivity (the 'hard' factors) and the 'soft' factors such as people, politics, emotions, and culture (Heeks, 2003). The hard-soft archetypal situation illustrates that information systems fail when individuals ignore the 'soft' (human issues) during the design of an information system project (Heeks, 2003). Madon (2004) underscores that ignoring available resources, skill-levels, values, beliefs, and motivations of those involved in the project lead to project failures. Dada (2006) insists that lack of training, skills, and change management efforts in ICT initiatives escalate the failure rates because they create a wide gap between the technology itself and the context within which it exists.

The private – public gaps are concerned with the difference between private and public sectors in that an information system designed for the private sector cannot work out of the box in the public sector context (Heeks, 2003). The private – public gap problem is associated with the public sector's non-competitive rate of pay as compared to the private sector (Dada, 2006). In the public sector, the recruitment of high quality IT

professionals is low (Ciborra & Nevarra, 2005; Nfuka, 2007), a situation which leads to the need for outsourcing IT solutions from the private sector. However, when a system developed for the private sector is adopted in the public sector, there is always a clash of culture and values (Heeks, 2003).

The last archetypal situation, country context gaps, exists when an information system developed for a developed country is implemented in a developing country (Heeks, 2003). Dimensions of this archetypal deal with the situation of technology transfer (Avgerou & Walsham, 2000), when a solution developed for a developed country is used as it is in a developing country context. Gaps arise due to differences in working cultures, skill sets, access to technology, and relevant technological infrastructure (Heeks, 2003).

The discussion on design – reality gaps reveals that technology transfer (Avgerou & Walsham, 2000) initiatives in developing countries do not account for the influence of the local context. However, the characteristics of the technology itself may exacerbate the tendency to ignore the local context. For example, if a technology ships with laws that restrict others from touching it, or if some secrets that would support participation of the indigenous are hidden, it distances itself from the local context. In ICT initiatives, proprietary software has many restrictions. In contrast, FOSS promises much freedom that would encourage higher participation of local developers and addressing the 'soft' issues. However, the question now is, as Chopra and Dexter (2008) have put it, how to decode the FOSS liberation in the context of information systems in developing countries.

3.2 Diffusion of Innovation

A dominant perspective for analysing technology transfer is the Diffusion of Innovation (Rogers, 2003). It is a *technological deterministic* perspective, which maintains that the spread of technology and routines from developed to developing countries or within the developing countries have a predefined effect on an organisation or community (Avgerou & Walsham, 2000). This dominant perspective conceptualises technology transfer as a process by which an innovation is communicated through certain channels over time among the members of a social system (Nhampossa, 2006).

However, the technological deterministic perspective has been criticised because it fails to account for heterogeneous technological innovations like ICTs, since it considers technology as a material object lacking the social element (Lyytinen & Damasgaard, 2001). This perspective ignores social problems like politics, power, skills competence, culture, and lack of capital among the potential adopters of the innovations (Nhampossa, 2006), which are fundamental issues widening the design-reality gaps.

Daly (2002) points to the special problems developing countries face in adopting ICTs: 'most…hardware, software, and applications are for developed country markets, but are frequently used with little adaptation, and sometimes carry unexpected cultural baggage' (p.236). Furthermore, the sectoral context, where ICT is adopted, is important, because some sectoral components, such as the financial services sector, lead in the application of ICTs (Avgerou & Walsham, 2000). In developing countries, the health and education sectors lag behind (Asangansi et al., 2008). Implementing ICTs in education and health sectors is difficult because of strong links to and dependencies on other sectors. As Ciborra and Nevarra (2005) note, there is a scarcity of IT professionals in government dependent sectors; thus, successful implementation of ICTs depends on outsourcing IT professionals. The technological deterministic perspective falls short in addressing the importance of creating alliances between human and material objects as networks around technological change. Actor-network theory (ANT) suggests that technologies do not pass through a neutral social medium (Latour, 1987). Technologies are in the hands of people who can appropriate it in the society (Latour, 1986). Technologies are continuously shaped and reshaped by the interplay of a range of heterogeneous forces within the networks (Bijker, Hughes, & Pinch, 1987).

3.3 Actor-Network Theory

In this study, ANT is an appropriate framework because it has a dense literature of work explaining, critiquing, developing, and applying the theory, and it covers important limitations of the technological deterministic perspective. In the next sections are the two fundamental concepts of ANT that were useful in this study. These are the *translation* process and *network analysis* model.

3.3.1 Power and Translation

While power is always assumed with authority, in ANT, power is understood as a consequence and not as cause of collective action (Stanforth, 2006). Callon (1986) presents the translation model of power as a successful command resulting from the actions of a chain of agents, each of which translates it according to their objectives. Those who are powerful are not those who hold power in principle, but those who practically define or redefine what holds everyone together in a group (Sanforth, 2006). In each group, 'you have to have spokespersons which speak for the group existence … defining who they are, what they should be, what they have been' (Latour, 2005, p.31). Thus, power is not a cause of people's behaviour, but a consequence of an intense activity of enrolling, convincing, and enlisting actors (Stanforth, 2006).

In ANT, actor network is configured and built over time through the enrolment of allies (both human and non-human) by means of the process called *translation* (Callon, 1986). During the creation of the networks, innovators attempt to create a *forum,* a central network that all actors agree is worth building and defending. Latour (1987, p.132) explains the translation process: 'it occurs as actors enrol allies in the actor network and align their interests in a continuous process of renegotiation, where claims become well-established facts and prototypes are turned into routinely used pieces of equipment.'

Callon (1986) describes translation as consisting of four moments: *problematisation*, *interessement*, *enrolment,* and *mobilisation.* Problematisation is the first moment of translation during which one or more influential or powerful actor(s) identify a real-world issue(s) and establish an obligatory passage point (OPP). An OPP is a situation that has to occur for all of the actors to be able to achieve their interests, as defined by the principal actor. Interessement, the second moment of translation, describes a process of convincing actors experiencing the problem to accept the definition of the focal actor. In the third moment, enrolment, , actors accept interests defined for them by the focal actor. Mobilisation, the last moment of translation, occurs as the proposed solution gains wider acceptance and an even larger network is created through those

acting as spokespersons for others.

The ANT power perspective reveals who has the power to hold all actors together. The translation perspective focuses on following actors to learn how they problematise, enrol, and mobilise others to support their preferred solution. While translation focuses on single networks, another ANT concept reveals that in a technological change project, there are both global and local networks. The next section presents the network analysis model.

3.3.2 Network Analysis

Law and Callon (1992) developed a network analysis framework for analysing the mobilisation of *local* and *global* networks of a technological innovation project. Global networks contain a set of relations outside of the project's local settings and context, built up, deliberately or otherwise, enabling the project to take place with the resources it provides, including money, expertise, and political support (Law & Callon, 1992). The local network is that set of relations that can be seen as the inside of the project; this set is necessary for the successful production of the working tool (Law & Callon, 1992). Callon (1991) explains that the interactions of the actors within and between the networks are achieved through items such as project deliverables, physical artefacts, and project reports.

The changing strength of each network over time can be plotted on *x* and *y axes*. The network analysis framework helps to determine the nature of a network in terms of progressive or degenerating towards achieving its intended goals (Law & Callon, 1992). If a network's trajectory turns down along the *y-axis,* global actors begin to lose their attachment; if it heads backwards along the *x-axis,* local actors cannot be properly mobilised. Law and Callon (1992) point out that a position in the bottom-left quadrant represents a weak, disaggregating project; a position in the top-right quadrant represents a solid, indispensable project.

3.3.3 Criticisms of Actor-Network Theory

Walsham (2001, p.46) argues that ANT 'is not a stable body of knowledge that can be

drawn on by researchers in an unproblematic way, since its developers themselves have frequently revised or extended elements of it.' The first criticism of ANT as a theory is that it does not typically attempt to explain why a network exists; it is more interested in the infrastructure of actor-networks, how they are formed, and how they can fall apart. However, this makes ANT more useful in this study because it does not bring in a 'how to' recipe; rather, it follows the actors to analyse the stability of the network formed, which helps to draw lessons from the ongoing project. ANT also incorporates what is known as a principle of generalised symmetry; that is, what are the human and non-human factors such as artefacts and organization structures integrated into the same conceptual framework and assigned equal amounts of agency (Goguen, 1998). Furthermore, ANT is criticised for its absurdity of assigning agency to nonhuman actors and for its amoral stance (McLean & Hassard, 2004; LTK, 2008).

Despite the many criticisms, ANT is valuable for conducting detailed empirical studies of the functions and dysfunctions of organisational processes. As a methodology, ANT has two major approaches: to follow the actor via interviews and ethnographic research, and to examine inscriptions, which includes texts (including journal articles, conference papers and presentations, grant proposals, and patents), but also images of many sorts and databases as central to knowledge work (House, 2003). ANT is a call for the close empirical study of associations. Latour (2005) says, 'if I were you, I would abstain from frameworks altogether. Just describe the state of affairs at hand' (p.144). In the next section are useful case studies on the application of ANT in IS in the context of developing countries.

3.3.4 ANT in Information Systems

Walsham (1997) acknowledges ANT as a promising theoretical vehicle for IS research. ANT is seen as a useful theory in information systems literature due to its explicit way of conceptualising technology as one of the 'actors' in any actor-network analysis (Walsham & Sahay, 2006).

ANT has been used to theorise both ICT and non-ICT based Information systems (IS) in various case studies (Lee & Oh, 2006; Ramiller, 2005; Stanforth, 2006; Walsham &

Sahay, 1999). Walsham and Sahay (1999) apply ANT to analyse a longitudinal case study of implementing GIS for District-Level administration in India. One of the implications drawn from the GIS study is that GIS is a non-human actor with inscribed interests (Walsham & Sahay, 1999, p. 58). Madon et al. (2004) apply the translation model of power to analyse non-computer based information system implementation of property tax reforms in Bangalore, India. In the question of technology transfer, ANT has been used to analyse technology transfer between countries (e.g., Akrich, 1992). More recently, Nhampossa (2006) has used the notion of translation to argue for technology transfer as technology translation in a case study from Mozambique. Stanforth (2006) uses ANT to explore the implementation of e-government information systems in Bangladesh, with a focus on demonstrating the usefulness of ANT to address the question of the diffusion and adoption of ICTs in developing countries.

The common theme in these studies is the effort to address the design-reality gaps through theorising the interplay between the social and the technical issues. Specifically, they demonstrate that most ICT projects degenerate when the social aspects of the technological change, which are tied to the local context, are ignored. The studies use ANT to analyse the mutually constitutive nature of society and technology to demonstrate the strategies key actors employ to enlist and mobilise support for their ICTs initiatives. Thus, the cited studies provide solutions for addressing the design-reality gaps by appealing for higher attention to the social aspects of any technological change.

3.4 Technology Translation

To understand the implementation of ICTs in developing countries, different perspectives, such as diffusion of innovation (Rogers, 2003), information technology transfer lifecycle (Baark & Heeks, 1998), and technology translation (Nhampossa, 2006), have been employed. The diffusion of innovation has been criticised for not taking account of the social issues of the context where the technology is implemented. The technology transfer lifecycles perspective (Baark & Heeks, 1998) conceptualises the process of technology transfer as a repetitive cyclical process starting from choosing technology, purchasing and installing, assimilation and use, adaptation and innovation

(i.e., transferring the technology to similar organisations). Recently, the technology transfer life cycle perspective has been criticised for taking technology as a whole; i.e., its specific characteristics do not influence its transfer (Nhampossa, 2006).

The drawbacks of diffusion of innovation (Rogers, 2003) and technology transfer life cycle perspectives (Baark & Heeks, 1998) suggest the rethinking of technology transfer as *technology translation* (Nhampossa, 2006). Nhampossa (2006) argues that the technology transfer life cycle perspective disregards political negotiations, which are essential for persuading bureaucratic governments in developing countries. For example, Wood-Harper and Bell (1990) suggest to donor agencies that while planning technology transfer efforts, crucial questions on available local support and the degree of necessary training must be addressed. Local support and training are issues that cannot be executed without political negotiations. They require re-directing human efforts to work with the new technology; furthermore, political brokering is a way to sensitise individuals who join the project. Technologies or systems become sustainable if they are institutionalised (Braa et al., 2004; Kimaro, 2006b). Sustainability implies that a technology is integrated into the routine of user organisations. However, while 'technology needs to be sustainable, at the same [users] need to remain flexible enough to accommodate changes occurring over time and space' (Nhampossa, 2006, p.57). The technology translation perspective then is different from the technology transfer life cycle in that it takes into account both the technology characteristics and the process of transferring the technology (Nhampossa, 2006).

There are four key factors that influence the technology translation process (Nhampossa, 2006): (a) legacy information systems, (b) customisation process, (c) user participation, and (d) balance of tensions between internationalization and localisation. Outdated systems designed with old technology exact inertia toward technological change in organisations. In the customisation process, changes in the software's configuration and/or source codes may be necessary when it is introduced into a different context. The technology translation perspective is also influenced by the participation of the systems' clients, who are local users and local developers. Participation is necessary for building human capacity development and ensures local

support of the system (Kimaro, 2006a; Wood-Harper & Bell, 1990).

Furthermore, the influence of international culture on local culture is not negligible. There is a need to balance between localisation and internationalisation in order for the technology to address local issues. Internationalisation refers to the process of isolating the culturally specific elements of the software, and localisation refers to the process of infusing cultural or business specific elements into already internationalised software (Nhampossa, 2006). This concept of balancing internationalisation and localisation is important because there is a higher order of using international software packages locally. By using FOSS products, it is possible to connect small local initiatives in building local, culturally embedded sustainable systems with international components.

Aanestad (2003) recognises the potential of small scale projects in implementing large scale information systems. She argues that if established projects were connected as dots, they could be rendered useful by virtue of a critical bootstrapping phase. This highlights the importance of re-using existing knowledge and resources – a major focus in the FOSS phenomenon. However, while there is a potential for connecting small dots (Aanestad, 2003), those small dots need to open up their ideas, software source codes, and strategies to be connected. This approach can better be executed with FOSS products. Aanestad's (2003) proposal is to consider established small scale initiatives within organisations. However, at the international level, the same concept of connecting small projects as "networks of action" is proposed (Braa et al., 2004). Those localised individual initiatives should be connected as a large network of actions through sharing experiences and mutual learning in order to ensure long term scalability and sustainability (Braa et al., 2004).

Having discussed some useful theoretical perspectives in addressing the design reality gaps, in the next section is theoretical framework of the thesis.

3.5 Conceptual Framework of the Thesis

The FOSS literature discussed in Chapter 2 reveals that the Free/Libre Open Source Software (FOSS) phenomenon is useful in developing countries because it lowers the

total cost of ownership of software (Weber, 2003); it facilitates technological development (Camara & Fonsesca, 2007); it helps developing countries escape intellectual property rights that lead to vendor locking situations (May, 2006); it is a means of acquiring knowledge; and it establishes an information economy (Weerawarana &Weeratunga, 2004). Furthermore, the FOSS phenomenon uses globally distributed software developers, and users are also the developers of the software (Feller & Fitzgerald, 2002).

Figure 3.1: Proposed Design-Reality Gaps in FOSS in IS in Developing Countries

Drawing from the design-reality gap analysis; social systems perspective on information systems and the FOSS perspectives, I develop three archetypal situations likely to hamper the development of Open Source Information Systems in developing countries. Those situations are: sponsor-developer, global developer-local developer, and local developer-local user gaps (Figure 3.1). Slicing the design – reality gaps into these small segments is useful in order to isolate, illuminate, and develop rich insights on the bottlenecks of FOSS implementation in developing countries.

3.5.1 Sponsor – Developer Gap

The FOSS literature identifies major stakeholder groups as 'developers, users, companies and non-profit organisations' (Feller & Fitzgerald, 2002, p.124). However, in developing countries, governments and donors are major clients and users of ICTs. Smith and others (2008) study of the integration of HIS in Tanzania argue that, 'donor agencies play a significant role in shaping the outcomes of health reform agenda in Tanzania' (Smith et al., 2008, p.8). Given that few high quality IT professionals are recruited in government-owned establishments (Ciborra & Nevarra, 2005); developers of government-based information systems are likely to be outsourced from private

sectors. As the developers come from a different context, they need to learn specific issues common in public sectors, especially politics and bureaucracy; otherwise, the public-private archetypal situation escalates (Heeks, 2003). Thus, there is a need to address the gap between sponsors (governments and donors as clients) and developers, i.e., the *sponsor – developer* gap. At organisation level, this gap is analogue to the gap between general management and technicians (Bakari et al., 2007).

The ANT's power, translation, and network analysis perspectives (Latour, 1986; 2005; Law & Callon, 1992) are useful tools for extracting rich insights from the case studies on addressing the sponsor-developer gap. These perspectives focus on power relationships, network formation, and analysis, which could reveal the progressive relationship between sponsors (governments and donors) and developers (mostly local developers) commissioned to realise a working tool (the software). FOSS-based concepts, especially project organisation and stakeholders' analysis (Feller & Fitzgerald, 2002), are useful tools to compare, contrast, and draw lessons from FOSS development projects on the way the sponsor-developer gap was addressed.

3.5.2 Global Developer – Local Developer Gap

When FOSS is used to develop education and health information systems, the developers are not the users of the resulting systems. Generally, global developers, when supporting local developers, help to create the software for local users. This is because local developers have access to the local systems and would know local user requirements. The inputs of global developers are needed to overcome several problems, such as limited access to technology (Heeks, 2003), lack of training and skills-levels (Dada, 2006; Madon et al., 2004), and poor skills of IT professionals (Ciborra & Nevarra, 2005). Because of the differences in terms of infrastructure, culture, and IT competence, there is a gap between global developers and local developers; this is the *global developer – local developer* gap.

The networks of action perspective (Braa et al., 2004) reveals the importance of mutual learning through sharing of experiences in different countries as a way of scaling and sustaining systems. In this learning process, Nhampossa (2006) adds that there is a

need to balance internationalisation and local issues. This approach is compatible with the bazaar development model (Raymond, 2001), which encourages geographically distributed developers to participate in local software development.

The main issues in this archetypal situation are clash of cultures and the different skills of global developers and local developers. Hofstede (2001) argues that it is critical to understand other cultures you may be doing business with. This argument is useful even for information systems especially in developing countries. Previous studies have indicated that, understanding culture will assist system analysts in understanding their clients' work practices and understanding certain behaviour, which are shared between people in a particular society (Thanasankit & Corbitt, 2000). There is a need to negotiate with various stakeholders in order to enrol them in the networks. Heeks (2003) argues that the hard-soft gap tend to get wide as IT designers ignore the local culture of the context where technology is implemented. Madon et al. (2004) adds that values and beliefs of the local society are issues that need to be considered in addressing the design-reality gaps.

One of the cultural dimensions for understanding contextual culture is the 'individualism' which refers to the degree to which individuals are integrated into groups (Hofstede, 2001). Societies on the individualist side, the ties between individuals are loose; everyone is expected to look after him/herself and his/her immediate family (Hofstede, 2001). On the collectivist side, people from birth onwards are integrated into strong, cohesive in-groups, often extended families which continue protecting them in exchange for unquestioning loyalty (Hofstede, 2001). To me, a means that can help to tell the individualism scale of a society is through political legacy, values and beliefs. Drawing from the legacy of *Ujamaa* policy in Tanzania, I argue that the Tanzanian culture falls under the collective side of individualism cultural dimension (Hofstede, 2001). That is because *Ujamaa*, a political policy practiced for years, is emphasising collective investment mode of innovation. The *Ujamaa* ethics are mutual respect, sharing of property and work (Nyerere, 1968).

Cultural influences in technology adoption have been studied in various case studies. In Thailand, a county which its cultural individualism dimensions falls under the

collectivism category (see Hofsted, 2001); it has been argued that relationships and connections play critical roles in business negotiations (Laosethakul & Boulton, 2007). The implication here is that political negotiations to cultivate establishment of relationships and connections is necessary in order to enrol various stakeholders. Braa et al. (2004) argue that 'the health sector in developing countries is intrinsically political' (p.357). That is why the key processes of HISP project in every country involve three aspects: gaining political support, HIS development and training (Braa et al., 2004).

Ujamaa connects well with the concepts of FOSS as a result it fosters justifications for adopting FOSS development. Under Ujamaa policy, all members have equal rights and equal opportunities and no importing injustice or exploitation (Nyerere, 1968). Similarly, FOSS licenses ensure that software can be used in any endeavours and no discrimination against others (Perens, 2005). Furthermore, as FOSS encourages private-collective investment mode of innovation (von Hippel & von Krogh, 2003; 2006), the economic moral rights in *Ujamaa* are right to subsistence and norm of reciprocity (Nyerere, 1968) which is the practice of exchanging things with others for mutual benefits. Through using common vocabularies well known in the context would make the society to grasp the main ideas of the FOSS technology. As users become more informed about the technology, resistance to change would diminish.

3.5.3 Local Developer – Local User Gap

In the context of health and education public sectors, the local developers are not familiar with the working practices of the users, especially in the case of health information systems. The users of these systems have a higher rate of computer illiteracy. Most have never touched a computer. Local developers need to learn the working practices of the users, functional requirements, non-functional requirements, and motivating strategies in order for users to participate in the development process. This analysis leads to the need for addressing a *local developer – local user* gap.

As argued in the technology translation perspective, software customisation and capacity building issues influence the adoption of ICTs in developing countries

(Nhampossa, 2006). Capacity building has been recommended by various authors. For example, Madon et al. (2004) argue that lack of training could lead to system failure. Similarly, Kimaro (2006b) argues that the tendency of donor-funded projects to ignore capacity building leads to unsustainable systems. FOSS's economic concepts shed light on clients' and developers' relationships. For example, the service offering business models in FOSS (Weerawarana & Weeratunga, 2004) could sustain continuity of FOSS projects. The relationship between users and developers could be maintained in demand/supply form. FOSS development concepts (Feller & Fitzgerald, 2002) encourage re-evaluation of user contributions, as users cannot contribute a software source code.

In summary, the theoretical perspectives demonstrate that ICTs initiatives are influenced by socio-technical conditions within the context where they are being implemented. Specifically, the design-reality gaps are exacerbated by the tendency to ignore the local context issues such as politics and technological infrastructure (such as Internet, computers, and software tools). Drawing from theoretical perspectives, I propose that implementing FOSS in Information Systems is influenced by the following factors: competence on relevant skills; technical infrastructure; development process; and political support.

CHAPTER 4: CASE DESCRIPTION AND RESEARCH METHODOLOGY

The research was conducted in Tanzania from 2005 to 2007. This chapter presents the research approach, data collection methods, and the use of theory in data analysis. The first section, 4.1, presents the personal motivation for carrying out a study on the development of Open Source Information Systems in Tanzania. This is followed by a presentation of the details of the research settings and fieldwork in Section 4.2. Section 4.3, which presents the research approach, is organised into two subsections: Interpretive approach and Participatory Action Research. Section 4.4 provides details of the data collection methods. In the last section is a presentation of the data analysis and use of theory in analysing data.

4.1 Personal Motivation

When working on my Master's degree, I became involved in health information systems as the subject of my research thesis. I studied computer database implementations in the Ministries of Health in Mozambique and Tanzania when involved in the Health Information System Programme (HISP).[3] There I learned that deploying software to the public sector without its source code leads to system failures and instability. Health systems change frequently, with new diseases and drugs being registered; to adopt those changes in the software requires access to the original authors, who retain the source code of their software as trade secrets. In cases where the original author is not reachable, the systems become outdated legacy systems in a short period of time. I fully support the use of District Health Information System (DHIS) in Zanzibar because the DHIS is Free/Libre Open Source Software (FOSS). This is because of the potential advantages that FOSS promises due to its attributed freedoms.

After completing my Master's in 2003, I secured a faculty post at the University of Dar es Salaam. That same year, I was appointed to head the examination office of the Computer Science department. At that time, exams were processed manually. It was an extremely tedious task to search student examination results. The department was in

[3] The Health Information Systems Programme (HISP) is a collaborative research and development network comprising universities, ministries, and not-for-profit companies in countries like South Africa, Mozambique, Malawi, Tanzania, Ethiopia, India, Vietnam, Norway, Nigeria, and Sweden.

need of an information system in order to automate the processes. While there is existing software to manage examination results, that software was not suitable in our case because the examination regulations were different than ours. Engaging private consultants to develop the software was not a promising option. I organised a team of developers to create an original system. A prototype of the Student Academic Register Information System (SARIS) was successfully developed and named ZALONGWA.[4]

The software was released under an open source licence, and many institutions adopted the software. Because each university has its own examination regulations, for example, the number of tests students should take in each subject, customisation of the software is essential at the universities. Thus, the open source licence is especially important because each university acquires the software with its source code. At the time of writing, six universities in Tanzania had adopted the software.

Although FOSS sounds like a break-through technology suitable in developing countries, its implementation is at its infant stages. Few studies have been conducted to address the challenges developers face in realising open source information systems in developing countries. Thus I propose a strategy for adopting FOSS products in developing countries. Developing countries are peculiar in that their technical infrastructure, FOSS relevant competences, and supportive policies are hard to find. However, they must be examined in order to better understand FOSS development.

4.2 Research Design and Description of the Case Studies

The case studies of the thesis are threefold: first, the study starts with conceptualisation of the FOSS phenomenon and its performance in organisations. This was done through literature review and fieldwork visits to organisations using mature FOSS products. Next was engagement in two FOSS development projects through participatory actions on software prototyping and documentation of FOSS related motivating and constraining development issues. Finally, trajectories of the projects with respect to politics involved, technical infrastructure, organisation of the development teams, means of acquiring external supports, user participation, and user contributions were analysed.

[4] ZALONGWA is a name, not an acronym.

Table 4.1: Overview of the Research Design

2005	2005-2006	2007
conceptualisation of the FOSS phenomenon and its performance in organisations through literature review and fieldwork site visits	engagement in two FOSS development projects through participatory actions on software prototyping and documentation of FOSS related motivating and constraining issues	analysis of trajectories of the projects with respect to politics involved, technical infrastructure, organisation of the development teams, means of acquiring external supports, user participation, and user contributions

This study followed the implementation of the HIS in Zanzibar and SARIS project at the University of Dar es Salaam. Thus, HIS and SARIS are the main case studies of this research. However, in order to gather information and understand the performance of FOSS products, I conducted an explorative study on the use of FOSS in organisations. Being a Tanzanian studying in Norway gave me an opportunity to survey the use of FOSS in both Tanzanian and Norwegian organisations. This survey was useful to learn and contrast the use of FOSS in mature infrastructure-related domains, such as in operating and database management systems, with the use of FOSS in end user applications software in information systems. The survey also highlighted the differences seen when FOSS developers are also users (as in infrastructure based software) and when FOSS users are not the developers (as in the information systems of health and education sectors).

The research settings can be classified into three groups: (a) Explorative Study on the Use of FOSS in Organisations; (b) SARIS at the University of Dar es Salaam; and (c) DHIS implementation in Zanzibar.

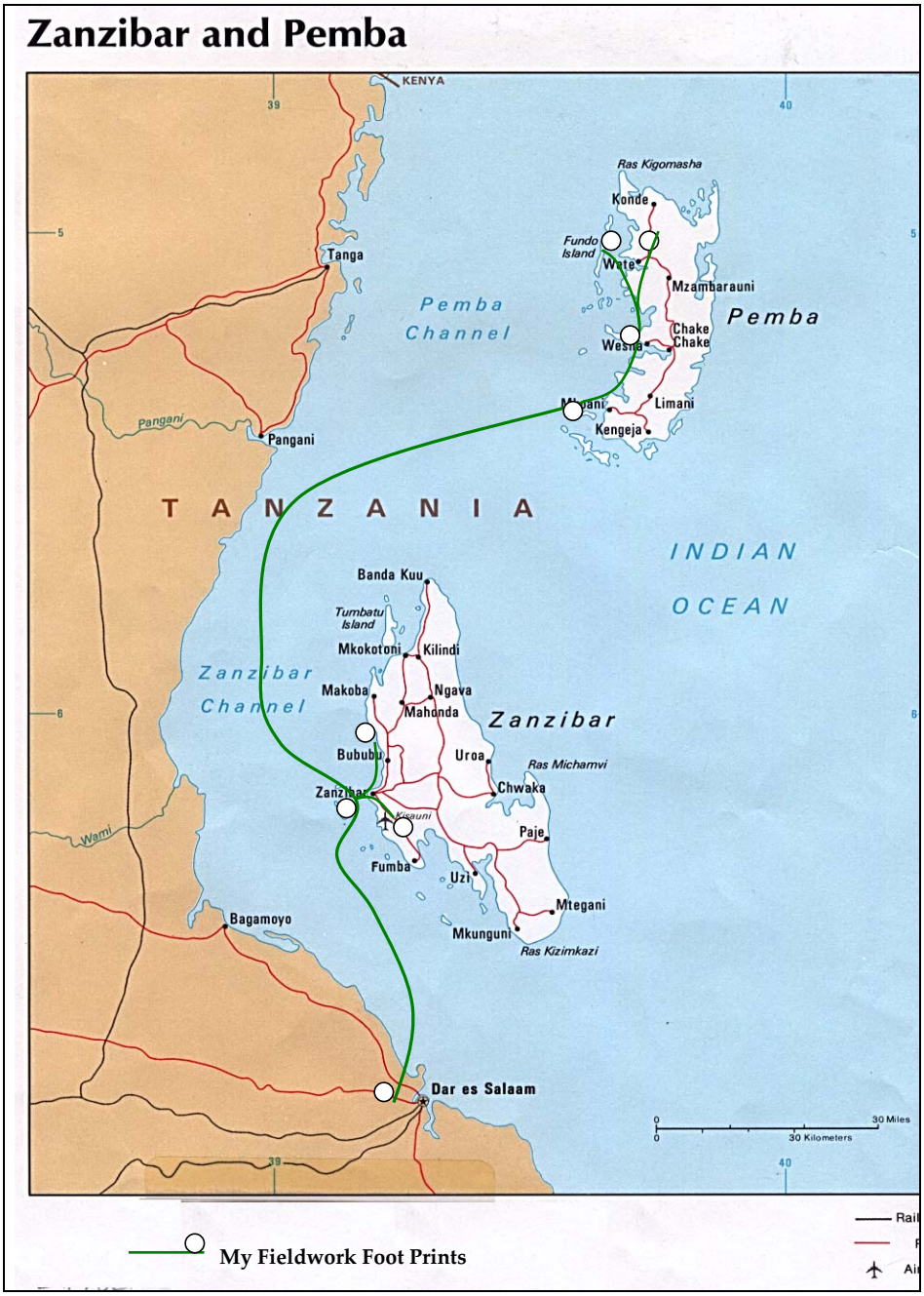


Figure 4.1 Map of Zanzibar and Pemba

4.2.1 FOSS in Organisations Case Study

In the period of October 2005 to December 2005, I conducted an explorative study on the use of FOSS in Norwegian organisations. In Norway, the settings were Hurum Municipal, Sarpsborg Municipal, University of Oslo, and the Agder University College. This was followed by a similar explorative study in Tanzania in March – May 2006 in the following settings: Tanzania Commission for Universities, University of Dar es Salaam, National Council of Technical Education, and the National Examination Council of Tanzania. These settings were selected because they run information systems with large amounts of data and they were using or intended to use FOSS products in their systems. The main agenda of the study was to learn the performance of FOSS products in terms of reliability, usability, availability of support services, and total cost of ownership (TCO).

4.2.2 SARIS Case Study

The rationale for implementing SARIS at the University of Dar es Salaam was to address three major problems in examination records processing: *nominal roll manipulation*, *arithmetic errors*, and *transcribing errors*. In *nominal roll manipulation*, student names and registration numbers are written differently in various documents; hence, it becomes difficult to track student records in various documents.

The *arithmetic errors problem* is categorised as either grading errors or summation errors. With grading errors, a lecturer can sum up the total marks correctly, but may fail to assign the correct grade. For example, if a B+ is equivalent to 60.5<=Marks<=69.4 and B is within the range of 50<=Marks<=60, a student who receives a 67 may mistakenly be given a grade of B. On the other hand, a summation error occurs when a lecturer fails to sum up coursework and final exam marks correctly. For those lecturers using Excel spreadsheets, there are different formulas for grading student examination results. For example, some lecturers use the formula presented in Box 1, while others use that in Box 2. With respect to Box 1, a candidate with 69.1 gets an 'A' grade, while the same score in Box 2, merits a 'B+.'

Box 1:

```
=IF(G14>69,"A",IF(G14>59,"B+",IF(G14>49,"B",IF(G14>39,"C",IF(G14>34,"D","E")))))
```

Box 2:

```
=IF(G14 >= 70,"A",IF(G14 >= 60,"B+", IF(G14 >= 50, "B", IF(G14 >= 40, "C", F(G14 >= 35, "D", "E")))))
```

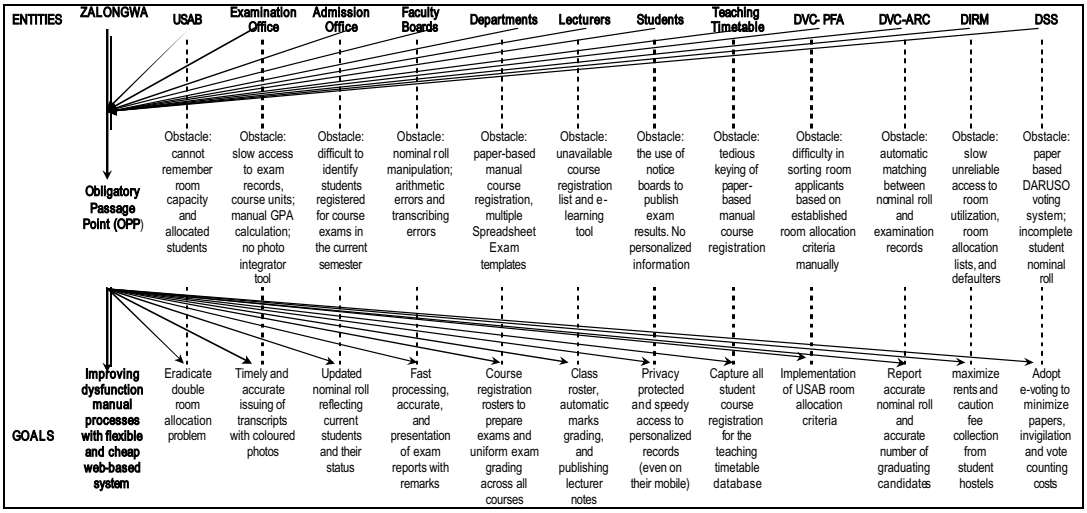
Transcribing errors occur when copying records from one source of data to another. The errors happen when examination results are transcribed from the course result sheet to a master file; they are due to manual typing errors.

The SARIS project started in 2003. The idea was that if all lecturers used the same system, they would also use the same formula for grading student records. Also, the system would be able to aggregate student records automatically, eliminating transcribing errors. However, the software was working as a departmental system. In the period of January 2005 to July 2005, I worked in the project as a case study; by that time, some faculties saw its potential as a university wide system.

During this period (January to July 2005), I was a programmer improving system functionalities for use at the faculty level. The second phase of my data collection in this case study occurred from March 2006 to July 2006, when the software was programmed to work at the university level. However, in this second phase, I participated more in political negotiations than in hardcore programming. I was a member of a special committee that designed a new university examination transcript and new student registration numbering format and proposed new working practices through the re-engineering processes of issuing examination transcripts.

Empirical material used in this study was collected when the project was implemented at the Faculty of Science, Faculty of Law, Faculty of Education, Faculty of Arts and Social Sciences, and at the University Examination Offices. Useful data collected from these settings include motivations for users to change from a paper based manual system to a computerised system. Following the actors revealed how they formed alliances in order to translate and defend their interested technical solutions.

Table 4.2: Translation in the SARIS Project

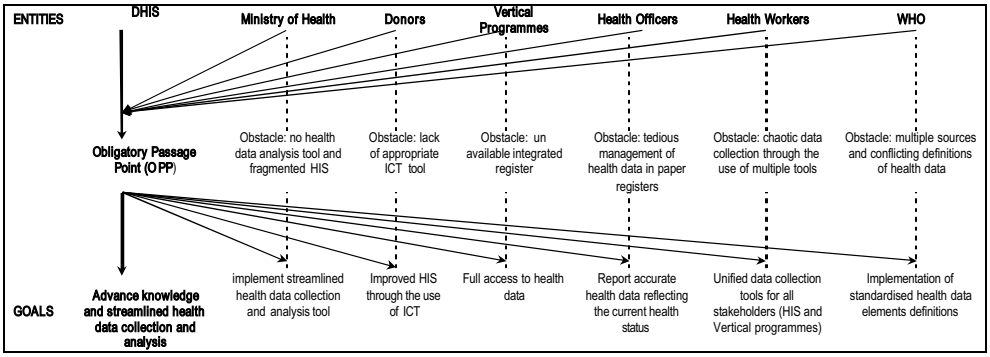


4.2.3 DHIS Case Study

The HIS project in Zanzibar started in January 2005. January to July 2005 involved project planning, recruitment of software developers, situational analysis, and formation of a special task force that represented the Ministry of Health in Zanzibar. The second phase of my participation in the HISP project in Zanzibar lasted from March to July 2006. The third phase took place in May – June 2007. Zanzibar has two main Islands: Unguja and Pemba. In Unguja, I worked at the Ministry of Health headquarters in Unguja, North A and North B districts. In Pemba, I was in the Mkoani, Chackechake, Wete, and Micheweni districts. First, I participated in the HISP team to conduct situational analysis of the availability of computers and their specifications. I again visited when testing newly designed health data collection tools (forms/registers). The third phase involved implementation of the DHIS in these districts.

Although the main setting was Zanzibar, I participated in other settings while implementing the DHIS. Specifically, I was involved in the HISP team when it was working in Tanzania mainland districts, including the Ministry of Health headquarters, Bagamoyo, Kibaha, Ilala, Temeke, and Kinondoni districts. In this thesis I used the experience gained from these other settings as well; for example, I critically evaluated the reliability and usability of the DHIS conducted in the Tanzania mainland districts. Moreover, in the time after I left the fieldwork sites, I was in contact with the subjects in the fields, especially the HISP and SARIS software developers. The post-fieldwork data collected through personal communications were invaluable to this study.

Table 4.3: Translation in the DHIS Project



4.3 Research Approach

Interpretive approach and action research were the main components of this research approach. The following sections present both interpretive and action research approaches.

4.3.1 Interpretive Research Approach

Guba and Lincoln (1994) define paradigm as the basic beliefs or worldview that guide the investigator, not only in choices of method, but in ontologically and epistemologically fundamental ways. According to Bryman (2004), an epistemological issue concerns the question of what is (or should be) regarded as acceptable knowledge in a discipline. There are three underlying epistemologies that guide qualitative researches in information systems: *positivist*, *interpretive*, and *critical* (Myers, 1997; Myers & Avison, 2002).

Positivism is 'an epistemological position that advocates the application of the methods of the natural sciences to the study of social reality and beyond' (Bryman, 2004, p.11). Positivism entails the following principles (Bryman, 2004): *phenomenalism* – only phenomena and hence knowledge confirmed by the senses can be considered knowledge; *deductivism* – the purpose of theory is to generate hypotheses that can be tested and allow explanations of laws to be assessed; *inductivism* – knowledge is arrived at through the gathering of facts that provide the basis for laws; *objectivity* – science must be conducted in a way that is value-free; and lastly, there is a clear distinction between scientific statements and normative statements and a belief that the former are the true domain of the scientist. Myers (1997) adds that positivists assume that reality is objectively given and can be described by measurable properties independent of the researcher and his or her instruments. In the positivistic approach, researchers establish propositions, quantify measures of variables, test hypotheses, and draw inferences from samples where the phenomenon is studied (Orlikowski & Baroudi, 1991).

Interpretivism is an epistemology not like positivism because there is a shared view that the subject matter of the social sciences, people and their institutions is

fundamentally different from that of the natural sciences (Bryman, 2004). An interpretive approach to information systems research assumes that reality is socially constructed (Orlikowski & Baroudi, 1991). Klein and Myers (1999, p. 69) argue that 'it is assumed that our knowledge of reality is gained only through social constructions such as language, consciousness, shared meanings, documents, tools, and other artefacts.' Walsham (1995a) contends that in an interpretive perspective, value-free or objective data cannot be obtained, since the research process itself relies on the researchers' preconceptions. It is through the interaction between researcher and subjects that the initial preconceptions of both parties are changed (Walsham, 1995a). Thus, the interpretive approach to information systems seeks an understanding of the context of the information system, as well as mutual influence between the system and its context (Walsham, 1993).

The main concerns for the critical research approach concern issues of historical and cultural contingency (Orlikowski & Baroudi, 1991) and power relations that are produced and reproduced by people (Klein & Myers, 1999; McGrath, 2005; Myers, 1997). The critical research approach focuses on the oppositions, conflicts, and contradictions in contemporary society, and seeks to be emancipator (Myers & Avison, 2002). Critical researchers have 'a cause… they may see a particular conflict and focus on that, downplaying other potential interpretations' (McGrath, 2005, p.86). A distinguishing feature of critical research is to engage with questions of an overtly political or moral nature, for example, a form of marginalisation relating to technology mediated knowledge manifested in the 'digital divide' discourse (Avgerou, 2005, p. 106).

This study used the interpretive approach. Although the study had the political agenda of liberating health workers from time-consuming processing and reporting of health data and improving a dysfunctional student records information system, the study focused on how to make FOSS implementation work in developing countries. Thus, it sought to create new knowledge on technological change using FOSS technologies. This deviates from purely critical research; critical research would be achieved by producing new knowledge on the role that ICTs plays, as Avgerou (2005, p.107)

argues, 'in contemporary society … to form streams of sustained research and debate on ICT and social change.'

The interpretive approach was used in this study to delineate the socio-technical processes involved in the implementation of FOSS products, ZALONGWA, and DHIS, from the perspective of various heterogeneous actors. The introduction of the software in the health information system (HIS) in Zanzibar, for example, was a challenging, politically contested process requiring political brokering to align various vertical programmes, health managers, and health workers in the health facilities. Also required were higher level technical innovations to customise the DHIS software.

The interpretive approach was also useful for explaining the social and organisational context of FOSS developers. They embrace FOSS development, facing the reality of their context in terms of infrastructure, skills competence, and working tools such as computers and relevant software. This fits well with interpretive research, which aims 'at producing an understanding of the context of the information systems, and the process whereby the information system is influenced by the context' (Walsham, 1993, p. 14).

4.3.2 Participatory Action Research

Action Research employs methods from both experimental and naturalistic (interpretive) traditions, but it is more reliant on naturalistic inquiry in that all research occurs within its natural context (Walsham, 1993). The ethos of action research is interpretive, incorporating social inquiry based on the views and interpretations of the participants (De-Villiers, 2005). Dick (2002) explains action research as a research approach, which has the dual aims of action and research: (a) action to bring about change in some community or organisation or programme, and (b) research to increase understanding on the part of the researcher, the client, or both.

The important distinction between Action Research and other kinds of research is the researcher's involvement in the whole action process as a change agent. Action Research aims not only to discover facts, but also to help alter certain conditions experienced by the community as unsatisfactory with intention to help the participants

to control their own destines more effectively (Greenwood & Levin, 1998; Nielsen & Svensson, 2006). Action Research is distinguished from consultancy work because it is *practical* and *useful*; *research-based*; *participatory*; *democratic;* and involves *dialogue* between insiders and outsiders (Rolfsen & Knutstad, 2007, p. 348).

Selener (1997) describes four types of action research: diagnostic, empirical, experimental, and participatory. In the diagnostic approach, a consultant collects data on a problem identified by the client and then provides a recommendation. Changes may or may not be implemented. In empirical research, a consultant tests a hypothesis about the impact of actions taken by either researcher or client, while in experimental research, control groups are used to test the relative effectiveness of the changes implemented (Selener, 1997). These three approaches have similar characteristics; they are not participatory, in that there is a clear division in terms of the roles of the researcher/consultant and the client, and they are researched on actions. In contrast, participatory action research involves participants in both the research and change process and it integrates research and action in an on-going participatory process (Selener, 1997).

Whyte (1993) argues that participatory action research has three main features: *co-learning*, *participation*, and *organisational transformation*. The emphasis here is that 'learning (or co-learning) takes place in a local context where one has the possibility to start together, researcher and personnel, in searching for the specific problem, and together decide upon how they shall be interpreted, and which ways would be most appropriate in order to solve them' (Whyte, 1993, p. 56). The participatory process should also include searching for relevant concepts and recruiting candidates from the organisation who would enhance the implementation of the solution (Whyte, 1993).

The two case studies of the thesis are about transforming dysfunctional public information systems. Thus, the study is about implementing computerised information systems to liberate workers of lower cadres in an organisation from tedious, repetitive, error-prone, paper-based information systems. It fits in the frame of the participatory action research approach. Hence, the definition that I adopt is 'action research is a participatory, democratic process concerned with developing practical knowing in the

pursuit of worthwhile human purposes, grounded in a participatory worldview' (Reason & Bradbury, 2001, p. 1). A practical definition is 'social research carried out by a team encompassing a professional action researcher and members of an organization or community seeking to improve their situation' (Whyte, Greenwood, & Lazes, 1991, p. 3).

Participatory action research features co-learning, participation, and organisational transformation (Whyte, 1993); it is practical and useful, research-based, participatory, democratic, and creates dialogue (Rolfsen & Knutstad, 2007). The projects studied, DHIS and SARIS, aimed to improve management information systems. This research study was different from a pure observation study because the researcher's purpose was to introduce streamlined data collection, reporting tools, and computerised information systems for the purpose of change. In addition, these projects were very useful to the respective organisations because they addressed real-life problems, where the insiders themselves thought something should be done.

The way in which the two projects were executed differed from mere consultancy work because the approaches for their implementation were based on experiences gained from previous projects (for example, in the DHIS case, experiences from other HISP nodes such as mainland Tanzania, Mozambique, South Africa, and India) and on literature on FOSS; the approaches also involved both outsiders and insiders in a participatory way. Co-learning happened in various ways. While the insiders were not computer professionals, the outsiders had no detailed knowledge of the working practices and local problems of the clients. A good example is the design of health data collection tools at the island of Pemba, where the outsiders proposed to keep a health data element called 'road accidents,' while the local people proposed that there were more 'clove accidents' (falls while picking cloves from the clove trees) than road accidents; therefore, the road accident data element should be dropped. This simply means that the outsiders learned the local problems instead of imposing global health data elements. During training on the computerised health information system in the HISP project, health workers learned to 'click,' but the outsiders learned that to prepare training handouts for computer literacy courses is more than a cut-and-paste of

computer screens. I participated in the two projects to understand fully the working practices. It was in this HISP project that my 'medical related vocabulary' increased dramatically. If not for my total engagement in action research, I could not have learned these terminologies. Learning the medical language helped me to communicate better and perform comprehensive interpretive analysis of HIS reports.

As an employee of the institution where the SARIS project took place, I had better access to information than if I were a mere outsider. I learned and practised many things, from politics, organisational tensions, and examination regulations, to technical issues like programming styles, server settings, and FOSS in general. This experience went beyond research conducted through the use of traditional methods like questionnaires and interviews. The SARIS software provided a lesson to the university community that platform independent software that is accessible to all heterogonous computers (Windows, Linux, Mac OS) is possible.

The SARIS project spanned from the computer professionals at the Computer Science department to non-computer professionals in other faculties. Thus, I was an insider when the project was at the initial stages, but I became outsider when I had to learn the examination regulations of the other faculties, e.g., the Faculty of Law. HISP involved many outsiders, including myself. In both projects, the dialogue between the outsiders and insiders was facilitated by many deliverables and workshops. In this thesis, those dialogues are presented thoroughly using both translation and the network analysis model of the Actor Network Theory (Latour, 2005; Law & Callon, 1992).

In action research, the main repetitive research processes undertaken include *planning*, *action, observation*, and *reflection*. These processes comprise a series of cycles that feed into each other, with action research more an ongoing process than an event. Baskerville and Wood-Harper (2002) present the action research cycle as having five phases: *diagnostic, action planning, action taking, evaluation,* and *specifying learning*, all of which occur within the *client-system infrastructure*. During the diagnosis phase, identification or definition of a problem takes place. Once the problem is identified, alternative courses of action are considered in the action planning phase. Action taking is where a course of action is implemented. After implementing a course of action, the

outcome of the action is evaluated. A circle of action ends up with the identification of general findings. The client system infrastructure is the specification and agreement that constitutes the research environment (Baskerville & Wood-Harper, 2002). It provides the authority, or sanctions, under which the researchers and host practitioners may specify actions. The client-system also legitimates those actions with the express expectation that it will eventually prove beneficial to the client organisation.

There were binding contracts for implementing the studied projects that served as agreements between both parties. While in the HIS project, the Ministry of Health selected a committee of six people to participate in the project implementation, in the SARIS project, a team of three people was formed to scrutinise the suggestions made by the technical implementers. Those suggestions included overhaul of student registration numbers, discontinued use of examination identity cards, and redesign of the transcript examination templates.

4.4 The Action Research Cycles

In the software engineering field, software prototyping is a component of "evolutionary" approach to software development (Sommerville, 2001). Software prototype is an initial version of a software system used to demonstrate concepts, try out different design options, and to find out more about the problem and its possible solution (Sommerville, 2001).

The fieldwork of this research was dominated by software development, design of data collection tools, and formatting reports to be used to input and output data on the respective software. In the two main cases of the research, HIS and SARIS, prototyping was very useful in facilitating user participation/involvement. The prototyping activities were executed as action research cycles (Baskerville & Wood-Harper, 2002), although there was significant overlap between the cycles. That is, after the first cycle, the subsequent cycles were overlapping and it was not easy to differentiate between the start and end of the research cycles.

4.4.1 DHIS Prototyping Activities

The software implemented in the HIS project in Zanzibar is the District Health Information Software (DHIS). The DHIS was not developed from scratch in Zanzibar; instead, the software was obtained from South Africa. Our work was to customise it by making the paper forms like the computerised forms and to format reports. In the beginning, the software did not have a license, but was acknowledged as Free Open Source Software (FOSS). Currently, the DHIS is deployed with a license tag which meets the requirements of FOSS products. At the time of writing, the DHIS licenses reads:

> DISTRICT HEALTH INFORMATION SYSTEM MODULES-END USER LICENSE AGREEMENT
> ... You may make and distribute unlimited copies of the Software, including copies bundled with commercial products, as long as each copy that you make and distribute contains this Agreement and is distributed for free. ... You are free to modify, translate, or create derivative works based on the DHIS software, again provided that this End User License Agreement is attached to the DHIS parts of these works and that such parts are provided for free...

Major activities executed in the HIS implementation were the design of data collection tools, testing of data collection tools, software customisation, software installation, user training, and software evaluation (Table 4.4). The DHIS was the axis of all activities in the HIS project. In ANT's terminology, the DHIS was the 'obligatory passage point' of the various actors in the project, because it was used to accomplish goals of various stakeholders of the project (see Table 4.3).

Table 4.4: DHIS Prototyping Activities in HIS Case Study in Zanzibar

Phase	Specific Activities
Diagnosing	• Semi-structured interviews with health staff in South Africa • Observations on health data collection and analysis • Use of checklists by inspecting data registers, analysis tools, and health workers staffing levels • Assessing available computers and computer programs • A kick-off workshop between HISP team and MoH Taskforce
Action planning	• Acquiring DHIS Software from HISP network • Developing strategies for importing baseline data to DHIS • Acquiring sample health indicators

Phase	Specific Activities
Action taking	• Mapping of data elements to indicators • Reviewing health data collection forms • DHIS database setup and data importation • Designing new health data collection forms • Training health workers in computer literacy • Training health workers in health data analysis
Evaluation	• Group discussions with health workers • DHIS demonstrations • Testing newly designed data collection forms • Calculating indicators • Comparative with baseline data • Retrospective testing
Specifying learning	• Presentations and Fieldwork reports • Publications: Scientific peer reviewed papers

4.4.2 SARIS Prototyping Activities

The SARIS software, code name of ZALONGWA, was developed through FOSS-based technologies, namely PHP scripting language and the MySQL database management system. ZALONGWA software is licensed under the General Public License (FSF, 2007). At the time of writing, the SARIS license agreement read:

> ZALONGWA Student Academic Register Information System (SARIS)
> Copyright (C) 2006 Zalongwa Technologies Ltd.
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version...

Table 4.5: Activities in the SARIS Project

Phase	Specific Activities
Diagnosing	• Analysis of the manual student records system • Sensitisation workshop to brainstorm a course of action • Evaluating the correctness of old examination reports
Action planning	• Review of possible open source technologies • Recruiting project members • Mobilising resources including Linux server configuration • Acquiring sample student records (nominal roll and exam results)

Phase	Specific Activities
Action taking	• Design and programming of SARIS • Testing the system • Integrating the system with other systems • Training students how to use the system • Training staff, especially lecturers and admissions officers
Evaluation	• Group discussions with health workers • SARIS demonstration • Evaluating report outputs • Conducting think-aloud usability testing sessions
Specifying learning	• Presentations and Fieldwork reports • Publications of papers

The implementation of SARIS software at the University of Dar es Salaam is conceptualised here in terms of the global and local networks involved in the process. As proposed in Law and Callon (1992), the global and local networks are distinguished by who implements the tool and who funds or supplies resources necessary for the process of implementation.

In the SARIS project, the local network was comprised of the technical team, which included individual programmers and system administrators. In the later stages of the project, an IT private company was involved in supporting users and updating the system. The global networks were the faculties and directorates, sources of the implementation and vital decisions.

Table 4.6: Trajectory of SARIS Project

SARIS Solution Trajectory	Enrolled Network Actors	Network Inter-relation
Phase 1: Departmental SARIS • SARIS Implementation at the Computer Science Department	Local Network • Examination Officer (as both developer and user)	• Zalongwa Software accepted as Obligatory Passage Point (OPP)

SARIS Solution Trajectory	Enrolled Network Actors	Network Inter-relation
Phase 2: Faculty Level SARIS • Implementation of Faculty level Exam Reports • Implementation of online Gradebook (Exam Record Sheets) • Database setup: keying course, programmes, and Department and faculty information • Faculty-level implementation of the SARIS software • Testing of software	Local Networks • Programmers • Respective Faculty System Administrators Global Networks • Faculty Deans • Faculty Examination Officers	• Negotiation of Support and Services contract • Collection of baseline data such as information on courses, programmes, and student lists • Training Workshops for both lecturers and students • Review of the implemented examination regulations and reports • Development of online suggestion box
Phase 3: University level SARIS • Implementation of the SARIS software at the Examination Office • Implementation of Accommodation Module for the University Student Accommodation Bureau • Extraction and loading of old student records from various legacy systems	Local Networks • Central Administration System Administrators • Zalongwa Technologies Ltd. Global Networks • University Examination Office • University Admission Office • Directorate of Student Services • University Student Accommodation Bureau	• Design of online transcript reports • Negotiation of registering of candidate photos • Development and implementation of student accommodation criteria • Negotiation of student hostel information and naming schemes of rooms and blocks

SARIS Solution Trajectory	Enrolled Network Actors	Network Inter-relation
Phase 4: Consolidation and orientation • Continuous training of new students and lecturers • Development of data import tools • Implementation of data mining reports	Local Networks • Zalongwa Technologies Ltd Global Networks • Directorate of Income Generating Units • Directorate of Undergraduate Students	• Long-term user supports, especially recovering passwords, running room allocation, and importing examination records • Implementation of ad hoc reports

4.5 Data Collection Methods

Throughout the study, four main data collection methods were applied: semi-structured interviews, group discussions, software prototyping, and documents analysis. The methods were used in a triangulation form; i.e., one or more data collection methods were used to gather data from one setting. The discussion of the data collection methods follows:

4.4.1 Semi-structured Interviews and Observations

An interview is a data collection technique that involves verbal interactions between interviewer and interviewee (Cohen, Manion, & Morrison, 2000). It is an interchange of views between two or more people with a mutual interest. Patton (2002) argues that observation helps a researcher to obtain some additional information about the topic being studied. Also, observation avoids report bias from someone else by overcoming language barriers and observing naturalistic behaviour.

This study relied upon face to face interviews with informants, including programme managers, software developers, system administrators, heads of departments, and various end users of FOSS products. The interview guides used were semi-structured and open-ended in nature, but with the interviewer in control so as to direct the interview and obtain as much information as possible from the respondents.

In the *FOSS in Organisation Fieldwork,* interviews were the main data collection method;

analysis observation and group discussion methods were used as well. Thirty-eight interviews were conducted. In each organisation, the informants were from lower to upper cadres and were selected strategically in order to interview those involved in FOSS implementation and use. Table 4.7 presents the occupation and number of interviewees involved in the survey of the use of FOSS in organisations.

Table 4.7: Interviews of the use of FOSS in Organisation Survey

Position	Country	Informants	Interview Sessions
Senior Administrators	Tanzania	5	5
	Norway	2	2
Junior Administrators	Tanzania	2	2
	Norway	6	6
Technical Staff	Tanzania	5	5
	Norway	8	8
End users	Tanzania	6	6
	Norway	4	4
Total		**38**	**38**

In the *DHIS Settings*, I conducted interviews and group discussions with health workers and project managers of vertical programmes in various periods. Some of the interviewees were interviewed twice or more at different points in time. During the initial phase of the project, the focus was to negotiate and harmonise a minimum list of health data elements to be collected and reported in the health facilities. However, in the later stages of the project, the interviews focused on strategies to ensure user participation in the projects. A summary of the list of informants and their organisations is presented in Table 4.8.

Table 4.8: List of Informants in the DHIS Case Study

Organisation Units	Settings	Informants	Number of Interviews
Health Facilities	Health Centres	14	14
	Hospitals	8	16
District Medical Offices	Unguja	7	17
	Pemba	4	4
Ministry of Health	National HMIS Office	4	9
Donor Agency	DANIDA Zanzibar	2	6
Vertical programme	Malaria Programme	3	3
	HIV/AIDS Programme	2	4
	Expanded Programme for Immunisation (EPI)	5	5
Total		**49**	**76**

4.4.2 Group Discussions

This method was similar to the interview method in that it involved face to face discussions. However, group discussion allows the researcher to discover ideas concerning people's attitudes, perceptions, and experiences about the phenomena being discussed through the use of a group of people to clarify attitudes or beliefs in words that are probably not easy to articulate (Hoyle, Harris, & Charles, 2002). Normally, group versus individual interviews are conducted. Because of the nature of the projects followed, group works were a common routine. I met with users, developers, and other stakeholders during workshops, project progress briefing meetings, and lunch time, and I used those opportunities to introduce questions and take notes.

During the implementation of the SARIS at the university, I was a member of a special committee commissioned to re-design the university examination transcripts template and to analyse the university format of student registration numbers. The goal of this committee was to give feedback to the development process of the computerisation of the student information system. Also, we proposed changing the university student registration numbers format to have a pattern that could be programmed and validated through the computer system. We designed the university examination transcript, which was then implemented in the SARIS system.

4.3.3 Documents Analysis

Documents analysis refers to the process of reading relevant personal, official, or public documents, which may be valuable sources of information to the research. There are various sources of information that can be used in document reviews, such as educational reports, meeting reports, conference reports, educational policies, circulars, pamphlets, journals, dissertations, and text books (Cohen, Manion, & Morrison, 2000). Document analysis provides a good source of general background of the problem and an opportunity for studying trends over time. Documents analysed included organisational ICT policies, ICT project documentation, and ICT project proposals; other useful documents were server specifications, ICT policy documents, and working

regulations such as examination regulations and statistics reports. Health data collection tools and respective reports were additional useful documents for analysis. Also collected were resolutions of meetings that approved installation and use of FOSS products.

4.6 Data Analysis and Use of Theory

In qualitative research, data analysis starts in the field and is part and parcel of data collection (Orlikowski, 1993). In the course of data gathering, ideas about possible analysis occur. It is a continuous process involving the collection and analysis of data in the field. The process involves making sense of the raw data, with the aim of transforming raw data by integrating and organising comments in connection to real experiences into major patterns and themes (Silverman, 2006). As this study is framed in the interpretive approach to action research, keeping a journal of *field notes* was a major starting point for analysing the data. Bryman (2004) argues that field notes should be fairly detailed summaries of events and behaviour and the researcher's initial reflections on observations.

Throughout this research, I was inspired by Actor-Network Theory (ANT) (Latour, 1987; 2005). In the interpretive research approach, a theory cannot predict particular outcomes of the research because the relationship between technology and organisations are dynamic. However, a theory is a sensitising device for analysing and collecting data from the field. Walsham (1995a) informs researchers that a theory can have the following roles: as an initial guide to research design and data collection; as part of an interactive process of data collection and analysis; and as a final product of the research. In this study, ANT was used to guide the process of data collection and analysis.

In ICT implementation studies, ANT is useful in following the interplay of heterogeneous actors as their interests and intentions are inscribed in artefacts; the way actors interact;, and the way actors form alliances in order to mobilise support for a particular solution (Bijiker, Hughes & Pinch, 1987; Latour, 1987). These perspectives were useful, especially in software prototyping processes, because following actors revealed their strategies used to implement the systems. In this case, ANT's concepts

highlighted the importance of paying attention to actors' politics, disagreements, and conflict resolutions. For example, I was particularly interested in following the problematisation and interessement processes to learn how focal actors in the HIS and SARIS projects maintained their open source information systems.

Throughout this study, I had the opportunity to interact with the research community and discuss my research findings. I carried the data analysis process iteratively by reading literature and through discussion with various professionals and researchers, especially my supervisors, colleagues, and conference participants. When writing my research papers, I had the opportunity to address reviewers' constructive criticisms, which helped me to reflect on my research data. The review comments from journals and conferences directed me to read more relevant literature, refine my analysis, and refine my research approach. This helped to link my research with existing body of knowledge.

Specific findings, such as feedback to the DHIS developers on the importance of mimicking the paper forms into the software, the importance of flagging FOSS-based licenses in the software, and implications of using socially embedded leaders in information systems development, were presented in my various papers. In this thesis, I pursued combined analysis of all the papers to extract a generalisation of the study. An interpretive approach can be generalised in various ways (Walsham, 1995b): by developing concepts, by generating a theory, by drawing specific implications, and by contributing to the insight of the studied phenomenon. My goal was not to "refute" any theory; rather, I sought to increase our understanding of the challenges faced by developing countries in implementing FOSS products in information systems. This was done by contributing insights on the development of FOSS in developing countries and by drawing specific conclusions about what can be done in practical terms to facilitate an implementation of FOSS applications in developing countries.

CHAPTER 5: RESEARCH FINDINGS

5.1 Introducing the papers

At different stages of my doctoral study, I drew upon five papers. These papers are attached as appendices to the thesis. Their titles and abstracts are presented in the next section, followed by a concrete synthesis of the findings addressing the research objectives. The papers are listed based on the case studies. The first paper draws its empirical material from the explorative study on the use of FOSS products in organisations. The next two papers are based on the health information system case study, and the last two papers are based on the education information system case study. The relationship between the papers and the research objectives are presented in Figure 5.1.

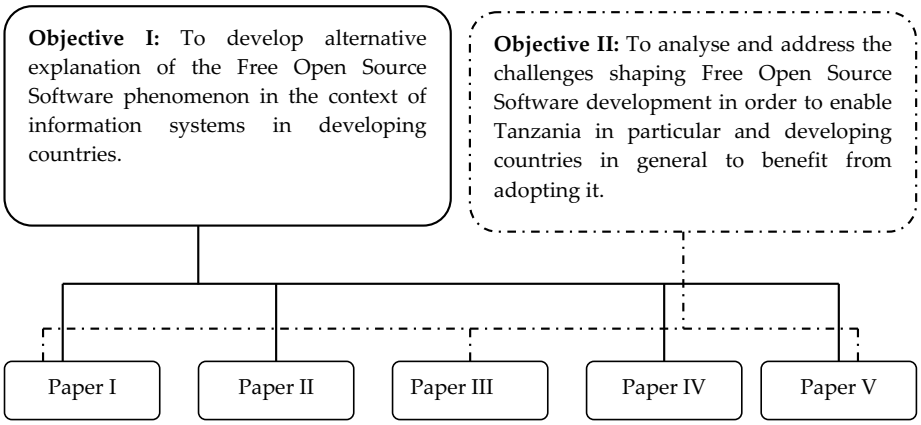


Figure 5.1: The Relationship between the Research Objectives and the Papers

5.2 Summary of the Individual Papers

5.2.1 Paper I

Reference:

Lungo, J. H., & Kaasbøll, J. (2007). The Use of Open Source Software in the Public Sector: Cases from Tanzania and Norway. Submitted to: *Information and Organization Journal (*previous version has been published In Silva**,** L**.,** Westrup, C. & Reinhard, N (Eds.), *Proceedings of the Ninth International Working Conference of IFIP WG 9.4: Social Implications of Computers in Developing Countries*, (pp.1-14), São Paulo, Brazil.

The paper presented the performance and advantages of open source software in publicly-owned establishments. Cases were drawn from public organisations in Tanzania and Norway in order to compare and contrast the perceptions of users with the performance of free open source products (FOSS). Respondents were primarily asked about their motivation to use FOSS products. The results indicated that lower cost, security, reliability, open-standards, and vendor independence are prime motivations for organisations to use FOSS products. It was argued that FOSS is more secure because of the availability of source codes, which enable developers and users to discover and fix vulnerabilities before a flaw can be exploited. Availability of source codes also facilitates reverse-engineering in order to make FOSS products comply with open standards.

Users were motivated to use FOSS products like web browsers, office suites, and mail clients because of the implementation of open standards, which enabled them to share documents. Furthermore, most FOSS products are security focused because they have been developed during the Internet era. Users argued that some FOSS products are too security sensitive at the expense of user friendliness. System administrators observed that Linux-based servers do not crash more frequently than their Windows counterparts; hence, they are more reliable. The study indicated that FOSS products lower IT expenditures in three different ways: they are cheap in terms of licence costs; the products run on cheap hardware, which eventually lowers hardware expenditure; and support contracts are cheaper when compared with support for proprietary software products.

There were, however, several shortcomings associated with the use of FOSS in organisations. Respondents of the study mentioned compatibility between open source software and proprietary software systems, switching costs, and few IT professionals to support open source products. Also, the cost of leaving a well-established network of supporting proprietary software such as Windows for open source software like Linux was not negligible.

5.2.2 Paper II

Reference:

Lungo, J. H., & Igira, F. (2008). Development of Health Information System in Zanzibar: Practical Implications. *Journal of Health Informatics in Developing Countries*, 2(1), 24-32.

The paper grounded its theoretical perspective on network analysis of ANT on implementation strategies of information systems. The paper made use of empirical materials from the DHIS in Zanzibar, with a focus of trying to understand the interplay between the social and technical issues. The goal was to streamline the design and implementation processes of Open Source Information Systems such as the health information system in Zanzibar. Concepts from the ANT moments of translation framework (Latour, 1987) and network analysis model (Law & Callon, 1992) were employed as lenses to zoom in and out of the implementation issues.

The findings indicated that the use of FOSS in implementing health information system facilitates the technology translation process. Specifically, FOSS is useful because of the availability of the software and knowledge surrounding that software. Primary actors can demonstrate their solution of their interest with practical data, which helps to entice and enrol other actors. As a result, it is easier to obtain the support of a higher authority. Generally, the paper's findings indicated that ICT projects get strengthened through carefully planned leadership of a project, clearly stated goals, and the participation of local networks and global networks. More important is the use of culturally-immersed leaders to spearhead the project. That is, local people should lead the project because culturally-immersed leaders know the context and can manoeuvre the formation of strong networks better than foreigners.

5.2.3 Paper III

Reference:

Lungo, J. H. (2008). The Reliability and Usability of District Health Information Software: Case Studies from Tanzania. *Tanzania Journal of Health Research 10*(1), 39-45.

This paper drew upon the customisation process of the District Health Information Software (DHIS). The focus was on evaluating the extent to which the DHIS has been adapted to meet the local requirements of the users in terms of its reliability and usability factors. In the usability factor, two measurement scales were used: accuracy and failure rate. In the usability factor, three measurement scales were used: training and support, format, and content. The goal was to develop constructive feedback to the developers in order to improve the software.

The software evaluation tests revealed that, at that time, although it had higher reliability, the DHIS was rated poor in terms of usability. DHIS fared poorly in terms of usability because it did not accommodate all health data. Users expected the DHIS to be a single point of contact for all kinds of health data, from routine health delivery services to vertical programmes data such as HIV/AIDS programmes. Second, the DHIS data entry forms did not match currently used data collection forms in terms of layout of the health data elements. For example, the first health data element in the paper form was the sixth health data element in the DHIS computer screen form. The same mismatch issue was noted for the manual reports when compared to the DHIS computer output. Third, the user training workshops did not equip users with the knowledge required to work with the DHIS.

The results of the DHIS evaluation demonstrated that software must be adapted to the local requirements. In the process of software adaptation, access to the source code is necessary in order to customise the user interface, business logic, and reports of the software. In other words, the customisation process goes beyond data entry and data editing. Developers need to pay attention to the local requirements, including functional requirements in terms of what the software should do, as well as non-functional requirements, such as layout of the fields in the data entry forms.

5.2.4 Paper IV

Reference:

Lungo, J. H. (2005). Re-inventing Higher Learning Institutions Communication Media: The Case of University of Dar es Salaam Student Information System. In A.O. Bada & A Okunoye (Eds.), *Proceedings of the Eight International Working Conference of IFIP WG 9.4: Social Implications of Computers in Developing Countries*, (pp.194-208), Abuja, Nigeria.

Despite the advancement of Information and Communication Technology (ICT), the information systems in most public institutions in developing countries are still in chaos. In this particular case, privacy-sensitive records, such as examination results, were published on notice boards and became open secrets to everyone, including the paparazzi. Generating transcripts involved seven steps: drafting, grading, typing, proofreading, verifying, signing, and photocopying. The university was only able to produce three transcripts per week, while the university graduates over 3,000 candidates annually. A student transcript request took at least three months. There was also a serious double allocation problem during the room allocation process for the student halls of residence. The university hostels can accommodate more than 7,000 students, and the hostels were designed in such a way that a single room was to be shared by three or four students (four beds in a room). No single student was to be allocated to more than one bed in a room, and no room was to be allocated more students than the number of beds therein. After the room allocation process, another problem followed: advertising the room allocation reports. Seven thousand names were printed on more than 300 pages, but no single notice board could display those pages. The reports were distributed on several notice boards, including on trees around the campus. A student then had to walk around the whole campus in order to find out if he or she had secured a room.

However, these problems of the manual system were mostly felt by the students and lower cadre officers of the university such as secretaries and examination officers. It was not an obvious problem for the managers. The lower cadre officers were desperately seeking an electronic system; however, an IT project is expensive. Thus, the SARIS was initiated. Here, the power of FOSS technologies, such as MySQL and PHP,

was revealed. The tools and source codes were free of charge. The project members then needed to assemble various pieces of the codes. The Zalongwa project members understood that it was a FOSS; hence, they focused on elimination of the problem, rather than on monetary gains. This approach enabled the Zalongwa project to take off, without depending much on senior administrators. From this case, we learn that the key to starting an open source software project in institutions is to get enthusiasm from volunteers in the organisation. The key project leader oversees and coordinates contributions from different members.

5.2.5 Paper V

Reference:

Lungo, J. H. (2006). Critical Issues Associated with Adoption and Use of Open Source Software in Public Sector: Insights from Tanzania. In J. Ljunberg & M. Andersson (Eds), *Proceedings of the Fourteenth European Conference on Information Systems*, (pp.732-744), Göteborg, Sweden.

The paper focused on tensions in qualification and transformation of FOSS in developing countries. The empirical materials in this paper were based on interpretive analysis of two software products studied: District Health Information Software (DHIS) in Zanzibar, and Student Academic Register Information System (SARIS) at the University of Dar es Salaam. The analysis was informed by two FOSS concepts, qualification and transformation, in order to shed light on the issues related to the implementation of FOSS in information systems of a developing country.

The development of FOSS in developing countries in Tanzania has been hampered by limited resources, both human and technical. This has resulted in the development taking a different approach than the promoted development models, especially the bazaar model. For example, while the most important vehicle of knowledge sharing in open source development is mailing lists (Sowe, Stamelos, & Angelis, 2007), these two projects did not have a mailing list because developers did not have access to the Internet all the time. Communication with international developers was conducted through personal e-mail address, mobile phone short messages, and telephone calls. All developers were co-located, and discussions were conducted face-to-face. The findings also indicated less competence on FOSS knowledge. Crucial issues on any

FOSS development were not given high priority at the beginning of the projects. On comparing the software products from these projects and the open source definitions some attributes of the open source definition were found to be missing, although the owners, both developers and the clients, claimed that their products are fully open source software.

The paper claimed that the freedom to use the software for any purpose and for any number of computers is the major distinctive advantage of FOSS over proprietary products. Additionally, rather than the bazaar model of FOSS (Raymond, 2001), a directed co-located development approach was used. The paper also revealed that while FOSS is not free of charge, administrators are more willing to pay contracted software developers than to purchase a software product. The paper concluded that open source development in the public sector of a developing country is hindered by limited ICT infrastructure, limited human resources, misunderstanding of FOSS licensing issues and FOSS business models, and preference for a formal, face-to-face negotiation culture over virtual teams and electronic communications.

5.3 Synthesis of the Findings

This section presents the links between the papers of the thesis. The research findings drew upon theoretical concepts and the analysis of empirical data. The findings are grouped into four themes:

- ☑ The performance and support of FOSS products in organisations
- ☑ The motivating and constraining issues in FOSS transformation
- ☑ The transformation process: project organisation and support proximity
- ☑ The translation as the process for building community around the FOSS products

The next sub-sections discuss these categories in detail; they are first summarised in Table 5.1.

Table 5.1: Links between the Papers

Key Themes	Papers	Key Empirical Findings
Performance and Support of FOSS applications	Paper I, Paper IV	- reliable FOSS products (do not reboot frequently) - support contract from external vendors - some FOSS products managed by internal staff

Key Themes	Papers	Key Empirical Findings
Motivating and Constraining issues in FOSS transformation	Paper I, Paper IV Paper V,	- freedom to make any number of copies - lower entry cost - political integration - misconception of the FOSS philosophy - technical infrastructure riddles the process - skills competence in FOSS technologies
The transformation process: project organisation and support proximity	Paper II Paper III, Paper IV, Paper V	- hired developers instead of volunteers - directed co-located development - bottom-up venture driven by champions - hands-on support - how-to support
Translation as the process for building communities	Paper II, Paper III, Paper IV	- use of culturally-immersed leadership - political negotiations - software demonstration with real data - user training - consultative workshops

5.3.1 Performance and Support of FOSS Applications

The empirical findings indicated that matured FOSS products such as operating systems, database management systems, and web servers have acceptable performance. For example, Linux servers do not reboot frequently when compared to windows servers. In addition to reliability, computer virus problems are less common in FOSS products. These findings confirmed the claims made in the FOSS literature arguing that FOSS development produces high quality software (see Raymond, 2001; Wong & Sayo, 2004). However, not all FOSS products perform better. This is especially true for those applications that fail to attract a significant number of developers. Informants cited lack of interoperability with other applications as a major problem with FOSS. For example, FOSS applications do not support the copying of application to another. Second, there was inconsistency with commands implementation. That is, for one application, the same command is named and located under a different menu in another application. As a result, this inconsistency lengthens the learning curve for users because they cannot use previous experience to learn a new application.

The concept of the private-collective investment model of innovation (von Hippel & von Krogh, 2006) implies that FOSS is a public good; thus, users have to depend on their own teams to support their applications in terms of software updates, customisation, and failure recovery. Interviews with users of FOSS applications in

organisations and experiences from the two case studies of the thesis revealed that FOSS products are supported by external contracted vendors. Private companies are contracted to deliver support services such as software maintenance and customisation. An example would be the case of Redhats support contract in Linux operating system in the universities, as presented in Paper I. Also, the municipals visited in this study had support contracts with a private company to support their thin-clients systems. However, there was internal support of FOSS products, as with the SARIS case study, which was initiated by internal staff. Furthermore, systems administrators in organisations supported the day to day running of the applications.

Generally, FOSS products receive both internal and external support of dealers. In objective two of the thesis, the aim was to find out how social conditions influence the development process of FOSS products in information systems. The findings indicated that FOSS is supported internally and externally. The internal support of FOSS products requires internal staff to be competent in FOSS technologies.

5.3.2 Motivating and Constraining Issues in FOSS Transformation

In this theme, two categories of the findings emerged: the motivations for users, developers and stakeholders to adopt FOSS, and the challenges developers face in adopting FOSS development (the bazaar model). In terms of motivations, the initial decision for proposing FOSS products was driven by the focal actors, or "champions of change". In the health information system case, the focal actors were the Ministry of Health information system officers, who proposed the health information system process. When the HISP team joined the reform process, a reproblematisation took place, and HISP became a focal actor (see detailed presentation in Paper II). The HISP project then proposed the DHIS software based on its experience working with this software. In the SARIS case, the focal actors proposed to use FOSS technologies to develop SARIS from scratch. However, two things found to be consistent motivating factors within the two projects led to acceptance of the software as "obligatory passage points": (1) the freedom to make and install the software to any number of computers and (2) low initial cost of the project (lower entry cost level) (see Paper I, Paper IV, and

Paper V). The health information system found that the DHIS could be acquired at no cost, customised by local developers (even if paid developers), and then installed in any number of computers in the country to be appealing. In the SARIS case, the software was developed without any contract. The university later had to pay for data migration and user support (see Paper IV). Thus, the freedom to use the software for any purpose in any number of computers (Presens, 2005) and lower entry cost level were major motivations for stakeholders to be enrolled in the contested FOSS solutions.

Despite the motivations revealed here, the transformation process was riddled with many challenges. Specifically, the constraints of the development process were threefold: (1) misunderstanding of the FOSS philosophy, (2) limited technical infrastructure, and (3) limited competence of FOSS technologies. At the beginning of the two projects, there were few efforts made to ensure that the FOSS products were open source software. A thorough qualification of the two projects was presented in Paper V. This demonstrates misunderstanding of the FOSS philosophy. Developers and users in the two projects treated their software as FOSS products. However, later on, slowly the two applications improved; for example, there were attached FOSS compatible licenses, and that then qualified them as FOSS products.

Technical infrastructures, which include availability of computers, computer accessories, and Internet, are serious issues influencing FOSS development in developing countries. Internet access is expensive; even if you have full connectivity, for example at the university, the network fluctuates (it is on and off). Developers do not own computers. As result, developers need to work at one place, a situation which limits their flexibility to work in their convenient time as the FOSS literature suggests.

Furthermore, there is an issue in managing the FOSS technologies. Despite the high profile of the developers (university graduates), managing FOSS technologies such as working in Linux environments, CVS configurations, and general programming, are big challenges. Most of the technical forces from the universities are not introduced to FOSS technologies. These findings demonstrate that the curriculum in higher learning institutions in Tanzania does not emphasise FOSS technologies.

5.3.3 The Transformation Process: Organisation and Support Proximity

This theme emerged during analysis of project organization and software development. The findings indicated that the actual processes of project implementation were organised as follows: paid developers were used in place of volunteers; there was directed co-located development instead of community virtual team development; and to large extent, project organisation was bottom-up. The SARIS project was a bottom-up process in the sense that the peripheral organs of the university (departments and faculties) adopted and used the system before being approved by a university authority. In the health system case study, although the political negotiations started at the ministry level, the implementation started at the peripherals (at district levels).

Furthermore, contributions from external developers were received in the form of "hands-on" and "how-to" forms. "Hands-on" occurred when external developers visited the local development team and demonstrated how to program a function. This was the case with the DHIS, where developers from the HISP network visited the development site in Zanzibar in order to support the local team. In the "how-to" model, local developers received external support through guidelines. The how-to support was received through telephone calls, SMS, and e-mails. In contrast, in the FOSS literature, developers contributed source codes.

5.3.4 Translation as the Process for Building Communities

The translation process theme is as follows: locate findings of the study that informs how actors (both human and artefacts) were recruited in the two projects. First, the two projects were organised in such a way that a culturally-immersed leader coordinated the respective projects (see Paper II and Paper IV). Culturally-immersed leaders are project leaders who come from the same organisation. For example, in the SARIS case study, the project leader was an internal staff member who held a post for examination regulations. In the health information system case, the development team was under the sixth health information system section of the Ministry of Health in Zanzibar. In addition, in the two projects, there was substantial involvement of political

negotiations between the local networks and the global networks. The political negotiations were needed to guarantee agreement on fundamental issues, for example, formulating a minimum list of health data elements and health indicators. Notable political negotiation outputs from the SARIS project included the design of new examination templates and a new format for student registration numbers. The net effect of using socially culturally-immersed and political negotiations was to build a community around the FOSS products. Political negotiations were featured even in selecting and contracting FOSS developers in the studied projects.

One of the key elements of the interessement phase of the translation process was software demonstration. The two projects used a similar strategy of extracting data from legacy systems and loading them to the software in order to demonstrate to users and stakeholders. Software demonstration and consultative workshops helped to convince actors that the contested tools promoted by the focal actors were fully functional. As presented in the case studies, a series of user training was organised as offsite training and onsite training. Thus, the communities around these projects were built through political negotiations, software demonstration, user training, and contracting developers.

CHAPTER 6: IMPLICATIONS AND CONTRIBUTIONS

…we have a past error to correct, and a present danger to avoid.

Mwalimu. J.K. Nyerere

This chapter contains the implications and contributions of the thesis. Section 6.1 presents implications of the findings with respect to the Free Open Source Software (FOSS) development in the context of Information Systems (IS) in developing countries. Section 6.2 presents practical contributions as strategies for making FOSS work in developing countries. The practical remarks, addressed to Tanzania in particular, are strategies for benefiting from FOSS development.

6.1 Theoretical Implications: Re-conceptualising FOSS Development

In this thesis, the term "re-conceptualisation" is used because the FOSS phenomenon is revisited in a different context: the Information Systems (IS) domain in developing countries. Unlike the infrastructure domain where FOSS developers are also the user, in the information systems domain, users are not developers. Also, while there is a significant amount of FOSS literature on infrastructure products and in the resource rich context, this thesis examines the FOSS phenomenon in a resource poor context, where FOSS development is less studied. This thesis examines the characteristics and the development processes of the DHIS and SARIS in order to compare and contrast with them the way the literature presents FOSS characteristics and development process.

In Chapter 2 of this thesis, the FOSS phenomenon was explored to better understand its philosophy, intellectual property rights, transformation, economics, motivations, and stakeholders. Based on the findings presented in Chapter 5, this thesis contributes the following key theoretical implications to the FOSS development in the domain of IS in developing countries:

- ☑ The importance of qualifying FOSS applications
- ☑ The benefit of FOSS products in developing countries
- ☑ Directed co-located development instead of bazaar model
- ☑ Technology translation as the process for building community
- ☑ The role of political negotiations in FOSS development and use

In the following subsections, each implication is discussed.

6.1.1 The Importance of Qualifying FOSS Applications

The discussion of the FOSS phenomenon in Section 2.2.3 of this thesis highlights the fact that license is the only institution in the governance structure of FOSS projects (Bonnaccorsci & Rossi, 2003) that distinguishes FOSS from other types of software. License is the significant marker and required characteristic of any FOSS product (Feller & Fitzgerald, 2002). Users in developing countries should carefully watch software license rights from the onset of their projects. The findings of this study revealed an *ad-hoc* way of qualifying FOSS products when a tentative qualification of the products was conducted (see Paper V). However, the two applications studied (DHIS and SARIS) were treated as FOSS products by their respective users, developers, and stakeholders from the beginning.

The study revealed that users relied on the way in which the software was advertised. The two applications have been referred to as FOSS products in various sources. The DHIS, for example, has been presented as flexible open source software in various works (e.g., Braa & Hedberg, 2002; Braa et al., 2004; Gjerull, 2006; Lungo, 2003; Nhampossa, 2006). In addition, organisations that support the DHIS in various countries advertise it as FOSS products, e.g., project websites in India (HISPINDIA, 2008), Nigeria (HISPNIGERIA, 2008), Tanzania (BEANISH, 2007), and South Africa (HISP, 2007).

The SARIS software is advertised as a platform independent FOSS product. The founders of SARIS have used this argument to impress universities that the software can be accessed from any computer platform at the university (e.g., Windows, Linux, or Mac). Today, the company that supports SARIS development supports other established FOSS products such as vTiger CRM, webERP accounting software, FOSS based Content Management Systems (Joomla!, Mambo, and Typo3) and it advertises various open source courses (ZALONGWA, 2007). Furthermore, the company supporting SARIS is listed as a country office, FreeCode Tanzania, of an international company that deals with FOSS products exclusively (FREECODE, 2008).

It is important to qualify FOSS products through their applied licenses. Additionally, this qualification process should take place right from the beginning of a project in order to avoid confusion with public domain software. Users are ensured the kinds of freedom envisioned in FOSS through licenses. Despite early findings on the way the software deviates from FOSS qualifications, at the time of writing, the developers of the DHIS and SARIS had made efforts to ensure that their applications qualified as FOSS products. SARIS software uses the GNU General public license (Stallman, 2002), a highly restrictive FOSS license (von Hippel & von Krogh, 2006). The DHIS license reads as follows:

> …You may make and distribute unlimited copies of the DHIS Software, including copies bundled with commercial products … You are free to modify, translate, or create derivative works based on the DHIS software…

The DHIS license gives four envisioned freedoms: the freedom to run the program, to study how the program works (as access to the source code is allowed), the freedom to distribute copies, and the freedom to improve the program (Stallman, 2002).

6.1.2 The Benefits of FOSS Development in Developing Countries

Drawing form the technology translation (Nhampossa, 2006) and the networks of action (Braa et al., 2004), the thesis indicates that use of FOSS products facilitates sustainable networks. As with the DHIS in Zanzibar, the software application was shared from a different country, and local developers in Zanzibar were supported by experienced developers from the HISP network. The formation of networks of action is important in sharing knowledge in order to facilitate the translation of software in order to accommodate local requirements. Given the low rate of IT professional recruitment in the public sector (Ciborra & Nevarra, 2005), developing countries would benefit if they could form networks around software applications, the knowledge of which should be public domain. Thus, the benefit of using FOSS products in developing countries is the facilitation of networks, collaboration, and sharing of software applications.

Specifically, this thesis recommends adoption of FOSS products in developing countries for the following reasons: FOSS is a means of acquiring knowledge; it avoids

locking into a situation, it allows culturally-immersed leaders (champions) to demonstrate their ideas live; and it fosters political integration in contested areas such as vertical health programmes (as discussed in paper II and Paper IV). In the case studies of this thesis, local software developers learned how to customise DHIS by working with the DHIS source codes and experienced developers. The DHIS license, a FOSS compatible license, gives the Ministry of Health the flexibility to choose anyone, and not just the original developers, to support the DHIS. Furthermore, FOSS products (DHIS and SARIS) help local socially embedded champions to demonstrate their ideas. In the DHIS case, the circle formed by introducing the idea of demo software was short-lived because the software was not developed from scratch. In the SARIS case, the tools for developing the software (PHP and MySQL) came pre-packaged in a server, a situation which allowed purchasing initial platforms for developing the software without discussion. A pre-working version of the system (as in the DHIS case) and pre-installed software that could be re-used to develop a new system (as in the SARIS case) expedited the development process and helped the primary actors of the project to demonstrate their ideas with a live system in a short period of time.

It is expensive to adopt FOSS products suitable for computerising information systems. Information systems are context sensitive; thus, software cannot be transferred from one context to another, but must be transferred and translated (Nhampossa, 2006). Translation means that the software needs to be adapted to local needs. The process of adaptation of software (customisation) leads to expenses that exceed the buying cost (license cost). Expenses related to FOSS products in information systems are due to personnel costs, hardware requirements, opportunity costs, and training costs, which together make up the Total Cost of Ownership (TCO) of software.

The findings of this thesis indicate that in the health information system project, there were contracted developers who earned monthly salaries. Also there were frequent visits of experts from different countries to Zanzibar in an effort to support the local development team. In addition to direct costs for sustaining the developers, there were additional costs for training users, piloting the software, and facilitating workshops between stakeholders. In the SARIS case, there were costs for migrating data from

legacy systems to the software plus user support services. This implied that FOSS is not free of charge. Rather, there are associated costs, of which FOSS adopters need to be aware. Other costs for adopting FOSS are associated with Internet dependence. With FOSS, developers need to be connected online, but the Internet is expensive. The case of downloading early versions of the DHIS in an expensive hotel attests to this.

One additional justification for proposing FOSS in developing countries is low expenditure on ICTs. It is taken for granted that because of the private-collective investment model of innovation practices (von Hippel & von Krogh, 2006), FOSS is cheaper than proprietary software. This is due to FOSS development being financed by individuals but being revealed freely in the public domain. Proponents of FOSS argue that the freedom envisioned in the FOSS philosophy does not mean that FOSS applications are not for sale. Stallman (2002) argues that the word "free" in FOSS does not imply FOSS products are free of charge. However, most FOSS products in infrastructure platforms, such as operating systems, database management systems, and many more, can be obtained for free (download and install).

In this study, the findings revealed that the use of FOSS development lowers ICT expenditure due to (a) free of charge software, (b) software running on old hardware, (c) software running on generic mass produced hardware, and (d) low support contracts (see Paper I). Paper I of the thesis, which presented a comparison of the license costs between proprietary software and open source software, revealed that FOSS products costs are low. Similar conclusions were reached in a comparison between proprietary software and FOSS software, which was almost always cheaper than proprietary software (Wong & Sayo, 2004). However, the comparisons were on direct license costs. This implied that in the event on-shelf FOSS products fulfil users' requirements, they also lower ICT expenditure in organisations.

6.1.3 Directed Co-located FOSS Development as a Coping Strategy

The thesis indicated that developers do not own the means of production, such as computers. The Internet was too expensive for developers to have constant access. In addition, the developers were not familiar with the working practices and user

requirements of the computerised information systems. Developers met in a single working office, where they were equipped with the tools needed to develop the software. This implied that co-located development was adopted due to limited access to technological infrastructure.

The concept that FOSS developers are scratching an "itch" (Raymond, 2001) does not work in information systems because the developers are not the users of the software. This research revealed that software developers have little experience with economic and technological challenges in the developing countries. As a result, developers need to learn the working practices of the information systems. In order for developers to configure the DHIS, for instance, they needed to learn health data reporting channels, aggregation levels, and calculation of health indicators. However, since the users were not IT professionals and more important users were computer illiterate, they could not contribute to source codes and user requirements online as the FOSS literature suggests. The findings of this thesis revealed a series of user training workshops which covered computer literacy courses and advanced courses on database applications. The DHIS users were trained to switch on/off computers, use operating systems, word processing, and spreadsheets; then they were trained to use the DHIS software. In contrast, the FOSS literature revealed that users contributed source codes, documentation, or translation. In other words, they were computer literate. The implication drawn here is that FOSS development process in this context takes a different route in order to accommodate these un-intended issues.

It has been argued that because of module based architecture, FOSS products enable shared and concurrent software development (Câmara & Fonseca, 2007; Feller & Fitzgerald, 2002; Raymond, 2001). Modularity is implemented through the object-oriented software development approach. However, as presented in this thesis, the two applications studied did not adopt an object-oriented approach. Lack of modularity led to challenges in adopting FOSS development principles. Developers were not able to divide the applications into small segments that could be checked out by individual developers over the Internet because of not using object-oriented development (see Paper IV and Paper V). Furthermore, in the two projects, there were no version control

tools such as CVS that enable distributed developers to check out source codes. This led to re-invention of the software development, where the development team was supported in different ways other than as told by the FOSS development principles. Instead of receiving source code contributions, the development team received a "how-to" support from distant developers. In most cases, the local development team was told how to implement a functionality instead of receiving a piece of source code to integrate with the software.

In addition to a how-to support through e-mails, SMS, and telephones, the local development team received hands-on support. Experienced global developers visited the site in Zanzibar and worked together with a local development team. Although this approach made the development process more expensive, it gave the developers greater access to technical support and generally fostered the development process. These workaround approaches contributed to our understanding of FOSS development. In addition to globally distributed developers who contribute software source codes, how-to and hands-on contributions are necessary for coping with technological and social challenges similar to the domain of the two case studies presented here.

In these case studies, technological challenges influenced the re-invention of the FOSS development process. First, developers did not own the necessary technological tools, including computers and Internet connections. Second, Internet access is limited and expensive. This resulted in re-invention of FOSS development, where developers had to use shared means of productions (computers and Internet), a situation that forced them to work together at special offices having the necessary facilities.

6.1.4 Technology Translation as the Process of Building Community

The findings indicated a delay in the decision in the two projects on which licenses to use for their respective software (see Paper V). Similar observation on delayed decisions was noted in DHIS2 project, a project which aimed to implement the DHIS software using Java technologies. It is presented that, 'during the first one and a half years of the project, there were several attempts to start a debate around the proper

license for DHIS2. None of these attempts took off, and no decision on this matter was made' (Nordal, 2006, p.61). This is exactly the opposite of the practical advice suggested in the literature, which argues that a FOSS project should start with the decision of license from the first days (Fogel, 2006). In the established FOSS development context, licensing is one of the driving factors contributors use to decide whether or not to participate in a project (von Hippel & von Krogh, 2006). FOSS's novel legal arrangements, such as "copyleft," provide an important point of engagement (Coleman, 2004b).

Use incentive is a motivation factor for developers to contribute in a FOSS project (Hertel et al., 2003; von Hippel, 2005). Other authors attribute the tendency of free revealing of FOSS developers to enjoyment, payments, and obligation motivations (Feller & Fitzgerald, 2002; Ghosh et al, 2002; Lakhani & Wolf, 2005). FOSS licenses define the restrictions and flexibility for a particular FOSS application (Lerner & Tirole, 2005). This implies that in order for developers to decide on joining and contributing on a project, they first examine the license of a particular project carefully. However, in the case studies of this thesis, the developers were not the intended users of the system, and they joined the projects without evaluating licences. As presented in Table 5.1, developers were recruited. Coleman (2004b) argued that the most common challenge in presenting FOSS in developing countries is not misconceptions on the part of stakeholders, but no conceptions. There is inadequate understanding of the technical, social, and legal intricacies of FOSS (Coleman, 2004b). In addition, FOSS technologies and its development arrangements to reflect the Bazaar model (Raymond, 2001) were not well tolerated by the social conditions of the context of my studies. This was demonstrated in my case studies during the initial days of the studied projects (Paper V).

The fact that these projects did not yet have licenses demonstrated that different strategies were used for recruiting developers. Specifically, the role of licenses at the beginning of the projects was perceived to be less important as compared to the principles of FOSS development. The justification that licensing attracts contributors then is pragmatic. In the information systems domain, the importance of FOSS licenses

is undermined because developers have no use incentive. They have payment incentive; thus they are recruited just like other developers in proprietary software. This implication confirms the findings of other authors, who argue that developers have extrinsic motivations in the form of payment (Lakhani & Wolf, 2005).

Although the decision for adopting licenses for the two applications was delayed, the two applications use highly restrictive licenses (Lerner & Tirole, 2005). As presented in Chapter 4 (Section 4.4.1 and 4.4.2), SARIS uses the General Public License (GPL). The DHIS license restricts users from using the software for commercial purposes (that is, making the software proprietary). This confirmed the observations of earlier studies that applications in user oriented domains are likely to adopt highly restrictive licenses (Lerner & Tirole, 2005) in order to prevent competitors from taking the software out of the public domain. However, the users in this context were much more concerned with their data in the system than the system's source codes. Given that the users were not developers, the value of the software increased as they inputted more data. Hence, a license justification that addresses their concerns with the protection of their data was more appealing than arguing that the license gives the software with its source codes.

The General Public License (GPL) is the most frequently used. Approximately 70% of FOSS projects use GPL (Freshmeat, 2008), the most commonly used version of which was released in 1991. Since its release (also referred to as GPL v2), much has changed; hence, it requires updating. One of the limitations of the GPL v2 is that it does not explicitly address the contributions of individuals in the form of Software as a Service (SaaS). SaaS is a software application delivery model in which a software vendor develops a web-native software application and hosts and operates (either independently or through a third-party) the application for use by its customers over the Internet (Shakermover, 2008). In the SaaS model, customers do not pay for the software itself but rather for access. This business model is more suitable in information systems, where users are charged for using the software, just as they would be charged for other services such as electricity. For example, in the SARIS, it is logical to host the system in special facilities instead of asking each small university to run an expensive computer server for hosting just one small database. This is not

covered by the GPL v2 because the licence says you need to give away the software with its source code. However, what if customers receive it as a service?

6.1.5 The Role of Political Negotiations in FOSS Development and Use

The case studies of the thesis featured two networks: local and global networks (see Section 4.4 in Chapter 4). While the local networks were featured by a technical development team and local users, the global network was featured by the donors, vertical programmes, and managers of the two information systems (see Paper II). Specifically, in the DHIS case, involvement of the Ministry of Health officers in Zanzibar and international organisations (WHO and DANIDA) was high. The Ministry of Health provided a taskforce of six highly ranked health officers to represent the government in the development process of the information system. DANIDA funded the project and WHO provided guidelines such as health data dictionary and indicators. In the SARIS case (as presented in Paper IV, Paper V and Table 4.6 of Chapter 4), the university authority appointed a team that developed a template for examination transcripts and reformed university student registration numbers. Furthermore, the university funded and authorised the process of migrating students' records from legacy systems (paper reports, word processor, and spreadsheet files) to the software. The university authority also approved the use of the SARIS software to issue transcripts of examination results.

The substantial involvement of information system officers and international agencies suggests that FOSS in information systems requires political support. This is because an information system is of organisational interest. This confirmed the findings of Câmara and Fonseca (2007), who argued that FOSS development needs to be funded to be viable. Furthermore, even if an individual can fund FOSS development privately, its use needs to be approved by the organisation in question.

6.2 Practical contributions: Decoding the FOSS Liberation

The practical contributions of the thesis are strategies to make FOSS development work in information systems, specifically in the Tanzanian context, and generally in developing countries. In Chapter 3 of this thesis is a conceptual analysis of archetypal

situations that can lead to FOSS development failures in the context of information systems in developing countries. Those situations include (a) developers – sponsors gap; (b) global developer – local developer; and (c) local developer – local user gap, as summarised in Table 6.1.

Table 6.1: Strategies for Bridging the Design – Reality Gaps

Archetypal Situation	Bridging Strategy
Developer – Sponsor	• Understanding FOSS philosophy • Political negotiations • Promoting the private sector
Global developer – Local developer	• Capacity building • Mutual learning • How-to and hands-on support • Understanding culture • FOSS technologies in higher institution curriculum
Local developer – Local user	• Participatory actions • Collaborations • Training

The practical contributions of the thesis were strategies to bridge the identified gaps arising from mismatches between the FOSS development and the nature of the information systems in the developing countries.

6.2.1 Bridging Developer – Sponsor Gap

Although the FOSS literature does not list governments and donors as stakeholders, in developing countries, the two are the main sponsors of ICT initiatives. FOSS development as well has no exceptions; it needs to be government funded in order to be viable (Câmara & Fonseca, 2007). Law and Callon (1992) argued that global networks of a project are a set of relations that can be seen as being outside of the project's local settings and context, built up, deliberately or otherwise, and enabling the project to take place with the resources it provides, including money, expertise, and political support. In contrast, the developer of a FOSS project consists of the local networks (Law & Callon, 1992), a set of relations inside the project and necessary for the successful production of the working tool. In this thesis, however, the sponsor of FOSS development project comprises the global networks of a project (Law & Callon,

1992).

The developer–sponsor gap concept goes beyond the adoption of software designed for the private sector in the public sector (Heeks, 2003). In the empirical findings in Table 5.1, there were misconceptions concerning the FOSS philosophy, political negotiations, and support contracts from external vendors. In any kind of sponsors and developers relationship, understanding the FOSS philosophy, political brokering, and promoting the private sector are important steps towards bridging the developer-sponsor gap. Specifically, in order for FOSS development to be successful in the information systems context of a developing country, there is a need to promote the FOSS philosophy. The participants of a project need to understand the FOSS philosophy.

Strategies to Facilitate Understanding of the FOSS philosophy

In bridging the gap between developers and sponsors, first both parties need to understand the FOSS philosophy. For example, in Paper V of the thesis, the two applications did not pass the FOSS qualification analysis, although the two applications were treated as FOSS products. While the developers understood the FOSS philosophy, the phenomenon was little known by sponsors. As Coleman (2004b) argued, the public sectors lack adequate understanding of the technical, social, and legal intricacies of FOSS. FOSS developers need to develop simple vocabularies connected to the goals of the clients of the information systems. For example, in the SARIS case study, although the developers were interested in FOSS products, their main justifications were to address the identified drawbacks of the manual student information system. The argument was that FOSS fosters user involvement and equips focal actors with tools not requiring significant initial investment.

With the DHIS case study, the agenda was also health information system reforms. Once the discussion between sponsors and developers became serious, in presenting the technology, FOSS terminologies were carefully chosen to ensure that those vocabularies meshed with the goals of the client systems. For example, the egalitarianism principles of access and dissemination mandated in the FOSS licenses were very attractive to the health sectors. The Ministry of Health officers and their

donors understood that with FOSS products, there are no license fees for installing the same software in unlimited number of computers in each district medical office in the country. This could be expensive if proprietary software were used because the license cost would be multiplied by the number of computers having the software.

In practical terms, distributing FOSS related issues among respective global and local networks is a good exercise for each party in understanding FOSS. This thesis contributes a strategy of mapping those motivating and constraining issues to project networks (Figure 6.1).

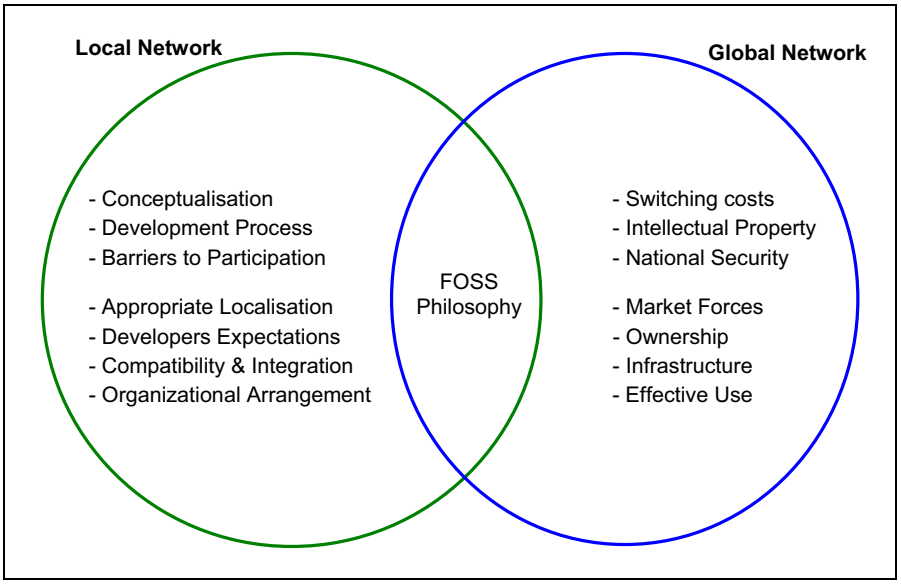


Figure 6.1: Mapping of FOSS Issues Motivating and Constraining to Project Networks

The first and most important issue is to make sure that the two networks have an understanding and are aware that their project falls under the definition of Free Open Source Software (FOSS). Hence, FOSS philosophy is at the centre of the two extremes. The constraining and motivating issues are then mapped (Figure 6.1) in order to redirect appropriately issues arising during the implementation of the project. For example, while local networks address issues like ensuring compatibility, integration, development process, and developers' expectations, issues like financing switching costs, infrastructure, and ownership could be handled by the global network. The goal is to ensure that sponsors find connections between their organisational goal and FOSS

development.

Strategies for Political Negotiations

The second strategy for bridging the developer–sponsor gap is gaining political support. Berg (2001) observed that the process of implementing patient care information system can only get off the ground when properly supported by both central management and future users. Working in the context of information systems in developing countries requires political strategies that mobilise stakeholders to accept the contested solution (Braa et al., 2004). A good example is the approach of HISP in Zanzibar and other countries. HISP enters fully into politics through two main approaches: setting up local facilities in a bottom-up fashion and engaging in negotiations with health officers (Braa et al., 2004). This implies that in order to gain political support from authorities, local presence and vivid examples of the solution are necessary. For example, when working in Zanzibar, HISP created a local development team equipped with an office and a residence house. Local politicians and health managers were convinced that the development team had a local presence with a telephone number to call in case immediate user support was required (as discussed in Paper II).

This strategy of having a local branch in a country is necessary. In proprietary software, organisations believe that the proprietary company is backing the software. In case of any problem, the supplier of proprietary software is responsible. In contrast, FOSS products are considered unsupported. In an "ideal" FOSS scenario, organisations need to depend on their own technicians to support their software. This is a challenge in non-IT intensive contexts like the health and education sectors. Thus, setting up local facilities is necessary to create a sense of security for users of the FOSS products in information systems. This strategy was also adopted in the SARIS project, where a private company was registered to ensure the clients that there was an entity supporting the development of SARIS software and hence its continuity.

This approach implies that the private sector is indispensable to providing local

technical support and competition in services that can put FOSS applications on level terrain with aggressively expanding commercial players (Coleman, 2004b). Thus, Tanzania in particular and developing countries in general should develop the private sector that focuses on FOSS development; its contribution in building technical support around FOSS product is invaluable.

6.2.2 Bridging Global Developer – Local Developer Gap

FOSS development encourages geographically distributed developers to participate in a project. As presented earlier, local developers benefit from the support of global developers. However, due to contextual social conditions, the two camps (global developers and local developers) may practise FOSS development principles differently. In Tanzania with its "collective culture" (see Section 3.5.2 in Chapter 3), software developers are unmotivated to engage in serious discussions with strangers. This generally affects their online communication behaviour. As Fitzgerald (2006) argued, language is another barrier. Although English is taught in Tanzania and is a medium of instruction at secondary schools and higher learning institutions, several studies attest to the difficulties teachers and students face in mastering the English language (Brock-Utne, 2007; Vuzo, 2007). Participating in the global community requires mastery of English in order to frame questions to be understood by a distant person. All of these factors contribute to widen the gap between global developers and local developers.

Considering the different culture and experiences between global developers and local developers, this thesis encourages "mutual learning". Global and local developers should come together as *jamaa.* That is sharing their software and knowledge for mutual benefit as in *Ujamaa* policy (Nyerere, 1968). As indicated in Table 5.1, global developers who provided hands-on support to the local team learned the local user requirements and challenges. That is, in the course of collaboration, global developers learned the local culture and other infrastructural issues, and the local developers learned technical skills. If a project is organised in such a way that global developers and local developers can meet, especially in the client system, global developers would

better understand the challenges faced by local developers in implementing FOSS principles. In the course of collaboration between global developers and local developers, learning occurs especially through sharing source codes and programming techniques.

Knowledge is translated through FOSS development when local developers obtain access to source codes of established software such as the DHIS. Access to source codes plus access to support enables local developers to advance their software development skills.

For Tanzania to benefit from and contribute to FOSS development, it needs to equip software developers with skills related to FOSS development. Higher learning institutions should be encouraged to update their IT curriculum. Currently, higher learning institutions in Tanzania offer computer science programmes. However, in order for these universities to equip computer science graduates with FOSS development knowledge, their curriculum should be focused. Specifically, object-oriented programming should be emphasised. In addition, FOSS native operating systems (e.g., Linux operating system) and office production suites (word processing and spreadsheet) need to be introduced to higher learning students.

My opinion is that FOSS Technologies curriculum should be introduced in all levels of the education system in Tanzania in the following order: at primary schools, introduce concepts on the difference between proprietary and open source technologies. At secondary schools, a more comprehensive curriculum on FOSS technologies should be introduced. For those schools equipped with computers, open source desktop applications should be used in the computer rooms. At tertiary colleges and at higher learning institutions, students who are majoring in computer science should be introduced to be able to master FOSS programming technologies in detail.

6.2.3 Bridging Local Developer – Local User Gap

This design-reality archetypal situation occurs primarily because developers are not familiar with the context of information systems. The adoption of FOSS is a special case because users have inadequate understanding of the phenomenon and FOSS

developers have little experience with the challenges users face in managing ICTs in this context. Specific strategies for bridging these gaps are important. In the HISP project, this issue has been explored in detail under various headings such as user participation (Nhampossa, 2006), human capacity building (Kimaro, 2006a), participatory action research (Lungo, 2003; Gjerull, 2007), and user training (Braa et al., 2004).

On the question of user participation, Nhampossa (2006) proposed mediation strategy. Nhampossa argued that to facilitate communication between with the strong bureaucratic and hierarchical environment of the health sector, mediation strategy would facilitate communication between local developers and users. In addition to mediation strategy, adoption of short- and long-term visions for dealing with the skills or capacity development of health staff is needed (Nhampossa, 2006). Nhampossa did not argue how to achieve mediation and capacity building, but rather provided necessary approaches for facilitating user participation.

The contribution of this thesis is a detailed insight on organising participatory action research and user training. The participatory processes in the two case studies were organised around the software products (see Paper II and Paper III). Stakeholders collaborated to design tools for collecting data to be entered in the software and printed out as reports. For example, in the DHIS case, a minimum list of health data elements was developed, health indicators were defined based on the minimum data set of health data elements, a health data element dictionary was developed, and data collection forms and reports were designed. In the SARIS case, users were involved in the design of university transcripts templates and student registration numbers. Both projects involved specially appointed committees of users dedicated to work with the technical development team.

This approach of appointing special committees to work with the technical team is a different re-organisation of user participation. This arrangement facilitates better dialogue between users and developers because both teams plan and execute activities of the project. This approach is much more sustainable than working with a single user because of the high turn-over of workers in the education and health sectors.

Moreover, special teams ensure reliable communications between the technical team and the higher authority, as they become mediators between the sponsors and developers networks.

In the DHIS project, users were trained in computer literacy (how to switch on computers, operate systems, and use word processing and spreadsheets applications); then they were trained on real issues involving the database information system. In the SARIS project, users were trained on the Internet course first, then the database information system. The developers recognised that users must be given general knowledge before being given complex training of database systems. In cases where computer illiteracy is high, first we need to introduce users to computer applications before providing them advanced knowledge on managing computer database systems. Database systems are advanced knowledge because they assume that a user knows how to switch on a computer, open an application system, and master a keyboard and a mouse.

CHAPTER 7: CONCLUSIONS

This thesis presented the Interpretive Participatory Action Research study of the adoption of Free Open Source Software (FOSS) in the domain of information systems in Tanzania. The case studies of the thesis were the implementation of the Health Information System (HIS) in Zanzibar and the implementation of the Student Academic Register Information System (SARIS) at the University of Dar-es-Salaam. The thesis focused on exploring the FOSS philosophy, principles, and development practices in order to compare and contrast the way the literature conceptualises FOSS and the way in which FOSS is practiced in the context of the study.

Two objectives were set to define the contributions of this thesis to the FOSS literature. Those objectives were (1) to develop an alternative explanation of the Free Open Source Software phenomenon in the context of information systems in developing countries.; and (2) to analyse and address the challenges shaping FOSS development in order to enable Tanzania in particular and developing countries in general to benefit from adopting FOSS. The thesis drew on the social systems perspective, which argues that technological changes are inherently affected by social-technical conditions of the society. Ignoring those conditions leads to widened design-reality gaps (Heeks, 2003) linked to the failure of many ICT initiatives in developing countries.

A thorough structuring of the FOSS literature under six headings (philosophy, intellectual property rights, transformation, economics, motivations, and stakeholders) was presented. This structural analysis of the literature allowed detailed exploration of the FOSS phenomenon. However, despite the various concepts presented, this thesis focused on issues related to FOSS development (transformation aspect). The thesis concluded that while FOSS proponents have been using various justifications for proposing FOSS in developing countries, some are pragmatic and thus hard to connect with the reality of the immediate and long-term goals of the information systems in developing countries. Some examples of the problematic justifications are: (a) FOSS license as a tool for attracting source code contributors and (b) the notion that FOSS products are cheaper. The thesis argued that there is a terrible misunderstanding of software licenses in this context; thus, the argument that developers would just join a

project due to license conditions is just not credible. In addition, implementing open source information systems involves substantial engagement of developers, who are not the users of the system. These developers would need to learn user requirements, communicate with and train users, and involve external experts, all of which would increase the total cost of owning the software.

The second objective was fulfilled through identifying and proposing strategies for bridging the design-reality gaps. Three archetypal situations that hamper FOSS development in information systems, especially in a developing country like Tanzania, were identified. Those situations were *developer – sponsor* gap, *global developer–local developer* gap, and *local developer–local user* gap. Strategies for closing these gaps were the practical contributions of the thesis. To bridge the developer–sponsor gap, the thesis argued that focusing on understanding FOSS philosophy, political negotiations, and strengthening the private sector are crucial. As FOSS development promotes participation of globally distributed developers, this thesis asserted that the effort to understand culture and capacity building through proximity to technical support and facilitating mutual learning if practiced would help to bridge the global–local developer gap. Mutual sharing concept of *Ujamaa* policy was recommended. Furthermore, the need to promote FOSS technologies curriculum in general and object–oriented software development in particular in higher learning institutions was emphasised. The last gap, the local developer – local user gap, could be addressed through participatory actions and user training, including computer literacy courses.

Summing up, the contributions of the thesis included re-conceptualisation of the FOSS phenomenon through the argument that contextual social-technical conditions influence the transformation of FOSS in developing countries. Re-conceptualisation implies that due to social and technical challenges, FOSS development does not take place in the same way in which the development is presented in the literature. In order for developing countries to benefit from FOSS development, this thesis argues to be aware of the influence of social-technical conditions on the development of open source information systems, and call for the development of specific strategies to address the design–reality gaps associated with the FOSS development process.

REFERENCES

Aanestad, M. (2002). *Cultivating networks: implementing surgical telemedicine.* Unpublished Ph.D Dissertation, University of Oslo, Oslo.

Ackermann, J. (2003). *Open source: theory and practice*. Dayton, Ohio: NCR.

Akrich, M. (1992). The description of technical objects. In W. E. Bijker & J. Law (Eds.), *Shaping Technology/Building Society: Studies in Social Technical Change*. Cambridge, MA: MIT Press.

Asangansi, I. E., Adejoro, O. O., Farri, O., & Makinde, O. (2008). Computer use among doctors in Africa: Survey of trainees in a Nigerian teaching hospital. *Journal of Health Informatics in Developing Countries, 2*(1), 10-14.

Avgerou, C. (2005). Doing critical research in information systems: some further thoughts. *Info Systems J, 15*, 103–109.

Avgerou, C., & Walsham, G. (Eds.). (2000). *Information Technology in context: studies from the perspective of developing countries*. Aldershot, UK: Ashgate Publishing Company.

Avison, D., Lau, F., Myers, M., & Nielsen, P. (1999). Action Research. *Communications of the ACM, 42*(1), 94-97.

Baark, E., & Heeks, R. (1999). Donor-funded information technology transfer projects: evaluating the life-cycle. *Information Technology for Development, 8*, 185-197.

Bakari, J. K., Tarimo, C. N., Yngström, L., Magnusson, C., & Kowalski, S. (2007). Bridging the gap between general management and technicians – a case study on ICT security in a developing country. *Computers & Security, 26*(2007), 44-55.

Barton, J., Alexander, D., Correa, C., Mashelkar, R., Samuels, G., & Thomas, S. (2002). *Integrating intellectual property rights and development policy*. London: UK Department for International Development Commission on Intellectual Property Rights.

Baskerville, R. L., & Wood-Harper, A. T. (2002). A critical perspective on action research as a method for information systems research. In D. Avison & M. D. Myers (Eds.), *Qualitative research in information systems* (pp. 129-145). London: Sage.

BEANISH. (2007). Building European- Africa collaborative Network for applying IST in Health care sector. Retrieved 15 August 2007, from http://www.hisptanzania.com

Benkler, Y., & Nissenbaum, H. (2006). Commons-based Peer Production and Virtue. *The Journal of Political Philosophy, 14*(4), 394-419.

Berg, M. (2001). Implementing information systems in health care organizations: myths and challenges. *International Journal of Medical Informatics, 64*(2-3), 143-156.

Berry, D. M., & Moss, G. (2006). Free and open-source software: opening and democratising e-government's black box. *Information Polity, 11*(1), 21-34.

Bezroukov, N. (1999). A Second Look at the Cathedral and the Bazaar. *First Monday, 4*(12), from http://firstmonday.org/issues/issue4_12/bezroukov/index.html.

Bhatnagar, S. (2000). Social implications of information and communication technology in developing countries: Lessons from Asian success stories. *The Electronic Journal for Information Systems in Developing Countries, 1*(4), 1-9.

Bhatnagar, S., & Bjørn-Andersen, N. (1990). *Information technology in developing countries*. Amsterdam: North-Holland.

Bijker, W. E., Hughes, T. P., & Pinch, T. (1987). The social construction of technological systems: New directions. In W. E. Bijker, T. P. Hughes & T. J. Pinch (Eds.), *The Sociology and History of Technology*. Cambridge, Mass.: MIT Press.

Boesen, J., Madsen, B. S., & Moody, T. (1977). *Ujamaa - Socialism from Above*. Uppsala, Sweden: Scandinavian Institute of African Studies.

Bonaccorsi, A., & Rossi, C. (2003). Why open source software can succeed. *Research Policy, 32*, 1243-1258.

Braa, J., & Hedberg, C. (2002). The struggle for District-based Health Information Systems in South Africa. *The Information Society, 18*(2), 113-127.

Braa, J., Macome, E., Costa, J., Mavimbe, J., Nhampossa, J., José, B., et al. (2001). A study of actual and potential usage of information and communication technologies at district and provincial levels in Mozambique with a focus on the health sector. *Electronic Journal of Information System in Developing Countries, 5*(2), 1-29.

Braa, J., Monteiro, E., & Sahay, S. (2004). Networks of action: Sustainability of Health Information Systems across developing countries. *MIS Quarterly, 28*(3), 337-362.

Brock-Utne, B. (2007). Learning through a familiar language versus learning through a foreign language: A look into some secondary school classrooms in Tanzania. *International Journal of Educational development, 27*(5), 487-498.

Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering*. Reading, Mass: Addison-Wesley.

Bryman, A. (2004). *Social Research Methods* (2nd ed.). New York: Oxford University Press.

Callon, M. (1986). Some elements of a sociology of translation: Domestication of the

scallops and the fishermen of St Brieuc Bay. In J. Law (Ed.), *Power, Action and Belief: A new sociology of knowledge?* (pp. 196-223). London: Routledge & Kegan Paul.

Câmara, G., & Fonseca, F. (2007). Information policies and Open Source Software in developing countries. *Journal of the American Society for Information Science and Technology, 58*(1), 121-132.

Castells, M. (1998). *End of Millennium, the information age: Economy Society and Culture*. Oxford: Blackwell.

Chiao, B. H. F. (2003). An economic theory of free and open source software: a tour from lighthouse to chinese-style socialism. Paper presented at the International Conference on Open Source, 26 July 2003, Academia Sinica, Taipei.

Chopra, S., & Dexter, S. D. (2008). *Decoding liberation: the promise of free and open source software*. New York: Taylor & Francis Group.

Ciborra, C. U. (2005). Interpreting e-Government and development efficiency, transparency or governance at a distance? *Information Technology & People, 18*(3), 260-279.

Ciborra, C. U., & Navarra, D. (2005). Good governance, development theory, and aid policy: risks and challenges of e-Government in Jordan. *Information Technology for Development, 11*(2), 141-159.

Ciborra, C. U., Braa, K., Cordella, A., Dahlbom, B., Failla, A., Hanseth, O., et al. (2000). *From Control to Drift: The Dynamics of Corporate Information Infrastructure*. Oxford: Oxford University Press.

Cohen, A., Manion, L., & Morrison, K. (2000). *Research methods in education* (5th ed.). New York: Routledge Farmer.

Coleman, G. (2004a). The political agnosticism of Free and Open Source Software and the inadvertent politics of contrast. *Anthropological Quarterly 77*(3), 507-519.

Coleman, G. (2004b). The politics of open Source Adoption, NGO's in the Developing World. Retrieved 9 May, 2008, from http://www.tacticaltech.org/SSRC_Report

Cook, I., & Horobin, G. (2006). Implementing eGovernment without promoting dependence: Open Source Software in developing countries in Southeast Asia. *Public Administration and Development, 24*(4), 279-289.

Dada, D. (2006). The failure of e-Government in developing countries: a literature review. *The Journal of on Information Systems in Developing Countries, 26*(1), 1-10.

Daly, J. A. (2002). Book review: information technology in context: studies from the perspective of developing countries. *Progress in Development Studies, 2*(3), 235–

268.

De-Villiers, M. R. (2005). Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. In J. Bishop (Ed.), *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries* (pp. 142 - 151). White River, South Africa: South African Institute for Computer Scientists and Information Technologists.

Dedrick, J., & West, J. (2003). *Why firms adopt platform standards: a grounded theory of open Source platforms.* Paper presented at the MISQ Special Issue Workshop on Standard Making: A Critical Research Frontier for Information Systems, Seattle, WA - 13 December 2003.

Dick, B. (2002). *Thesis resource paper: you want to do an action research thesis?* Retrieved 11 November, 2005, from http://www.scu.edu.au/schools/gcm/ar/art/arthesis.html.

Europa. (2003). *Cases of official recognition/adoption of FOSS*. Retrieved 18 February 2008, from http://europa.eu.int/information_society/activities/opensource/cases/index_en.htm

Evers, S. (2000). Development environments for open source software development. Unpublished Diploma Thesis, Technische Universtät Berlin, Berlin.

Feller, J., & Fitzgerald, B. (2000). A framework analysis of the open source software development paradigm. In W. J. Orlikowski, P. Weill, S. Ang & H. C. Krcmar (Eds.), *Proceedings of the Twenty First International Conference on Information Systems* (pp. 58 - 69). Atlanta, GA, USA: Association for Information Systems.

Feller, J., & Fitzgerald, B. (2002). *Understanding Open Source Software Development*. London: Addison-Wesley.

Fitzgerald, B. (2005). Has Open Source Software a future? In J. Feller, B. Fitzgerald, S. A. Hissam & K. R. Lakhani (Eds.), *Perspectives on Free and Open Source Software* (pp. xxxi, 538). London: The MIT Press.

Fitzgerald, B. (2006). The transformation of Open Source Software. *MIS Quarterly, 30*(3), 587-598.

Fogel, K. (2006). *Producing Open Source Software: How to run a successful free software project*. Beijing Sebastopol, Calif.: O'Reilly.

FOSSFA. (2004). *Free and Open Source Software for Africa (FOSSFA) Action Plan 2004 - 2006*. Nairobi: FOSSFA.

FREECODE. (2008). FreeCode International. Retrieved 17 May 2008, from http://www.freecodeint.com/art.html?catid=3

Freshmeat. (2008). Project License Breakdown. Retrieved 3 March 2008, from http://freshmeat.net/stats/#license

Friedmann, D., & McAdam, D. (1992). Collective identity and activism: networks, choices and the life of a social movement. In A. D. Morris & C. McClurg (Eds.), *Frontiers in Social Movements* (pp. 156-173). New Haven: Yale University Press.

FSF. (2006). Free software is a matter of liberty not price. You should think of "free" as in "free speech." Retrieved 10th November 2006, from http://www.fsf.org/

FSF. (2007). GNU General Public License. Retrieved 23 December 2007, from http://www.gnu.org/copyleft/gpl.html

Fulk, J., Flanagin, A. J., Kalman, M. E., Monge, P. R., & Ryan, T. (1996). Connective and Communal Public Goods in Interactive Communication Systems. *Communication Theory, 6*, 60 - 87.

Ghosh, R. A. (1998). Cooking pot markets: An economic model for the trade in free goods and services on the Internet. *First Monday, 3*(3). Retrieved 28 March 2008, from http://www.firstmonday.org/issues/issue3_3/ghosh/.

Ghosh, R. A., Krieger, B., Glott, R., & Robles, G. (2002). *Open Source Software in the Public Sector: Policy within the European Union*. International Institute of Infonomics, University of Maastricht, Netherlands.

Gjerull, N. F. (2006). *Open Source Software Development in Developing Countries: The HISP Case in Ethiopia.* Unpublished Master Thesis, University of Oslo, Oslo.

Glass, R. (2005). Standing in front of the Open Source steamroller. In J. Feller, B. Fitzgerald, S. Hissam & K. Lakhani (Eds.), *Perspectives on Free and Open Source Software* (pp. 81-92). Cambridge: MIT Press.

Goguen, J. (1998). Actor-Network Theory. Retrieved 17 March 2008, from http://www-cse.ucsd.edu/~goguen/courses/268D/6.html

Greenwood, D., & Levin, M. (1998). *Introduction to Action Research: Social research for social change*. Thousand Oaks, CA: Sage.

Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. In N. K. Denzin & Y. Lincoln (Eds.), *Handbook of Qualitative Research* (pp. 105-117). Thousand Oaks, CA: SAGE.

Hansen, M., Köhntopp, K., & Pfitzmann, A. (2002). The Open Source approach — opportunities and limitations with respect to security and privacy. *Computers & Security, 21*(5), 461 - 471.

Heeks, R. (2002). Information systems and developing countries: Failure, success and local improvisations. *The Information Society, 18*(2), 101-112.

Heeks, R. (2003). Most eGovernment-for-Development Projects Fail: How Can Risks be Reduced? In *iGovernment Working Paper Series, Paper No. 14* (pp. 1-17). Manchester, UK: Institute for Development Policy and Management, University of Manchester.

Heeks, R. (2006). *Implementing and managing e-Government: An international text*. London: Sage.

Heeks, R., & Stanforth, C. (2007). Understanding e-Government project trajectories from an Actor-Network perspective. *European Journal of Information Systems, 16*(2), 165-177.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: An Internet-based survey of contributors to the Linux kernel. *Research Policy, 32*, 1159-1177.

HISP. (2007). Health Information System Programme. Retrieved 17 May 2008, from http://www.hisp.org/

HISPINDIA. (2008). HISP India Website. Retrieved 17 May 2008, from http://www.hispindia.org/

HISPNIGERIA. (2008). The official website for HISP Nigeria. Retrieved 17 May 2008, from http://hips.ifi.uio.no:8080/display/HISP/Nigeria+Implementation

Hofstede, G. (2001). *Culture's consequences, comparing values, behaviours, institutions, and organizations across nations*. Thousand Oaks, CA: Sage Publications.

Hoyle, R. H., Harris, M. J., & Charles, M. J. (2002). *Research methods in social relations*. London: Thomson Learning.

Hughes, J. A. (1990). *The philosophy of social research* (2nd ed.). Harlow: Longman.

ITU. (2006). Statistics, International Telecommunication Union. Retrieved 5 May 2008, from http://www.itu.int/ITU-d/ict/statistics/

Kaasbøll, J., Fjuk, A., Karahasanoc, A., & Groven, A.-K. (2006). Improvements of teaching and tools for learning object-orientation. In A. Fjuk, A. Karahasanoc & J. Kaasbøll (Eds.) (pp. 205-220). California: Informing Science Press.

Kelty, C. M. (2000). Anthropology monsters, faces of gift. Paper presented at the American Anthropological Association Conference, 15- 19 November.

Kelty, C. M. (2001). Free Software/Free Science. *First Monday, 6*(12), from http://www.firstmonday.org/issues/issue6_12/kelty/

Keniston, K. (2002). Grassroots ICT Projects in India: Some Preliminary Hypotheses. *ASCI Journal of Management, 31*(1&2), 1-9.

Kimaro, H. (2006a). *Decentralization and sustainability of ICT based health information systems in developing countries: A case study from Tanzania.* Unpublished Ph.D Thesis, University of Oslo.

Kimaro, H. (2006b). *Decentralization and Sustainability of ICT based Health Information Systems in Developing Countries: A Case Study from Tanzania.* Unpublished Ph.D Dissertation, University of Oslo, Oslo.

Klang, M. (2005). Free software and open source: The freedom debate and its consequences. *First Monday, 10*(3), 1-15, from http://firstmonday.org/issues/issue10_3/klang/index.html.

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in Information Systems. *MIS Quarterly, 23*(1), 67-93.

Krishnamurthy, S. (2003). A managerial overview of open source software. *Business Horizons, September-October 2003* Retrieved 27 December 2007, from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=649903

Kshetri, N. (2004). Economics of Linux adoption in Developing Countries. *IEEE Software, 21*(1), 74-81.

Lakhani, K. R., & Wolf, R. G. (2005). Why hackers do what they do: Understanding motivation effort in Free/Open Source Software projects. In J. Feller, B. Fitzgerald, S. Hissam & K. Lakhani (Eds.), *Perspectives on Free and Open Source Software* (Vol. 2007). London: MIT Press.

Latour, B. (1986). The Power of Association. In J. Law (Ed.), *Power, Action and Belief.* London: Routledge and Kegan Paul.

Latour, B. (1987). *Science in Action*. Milton Keynes: Open University Press.

Latour, B. (2005). *Reassembling the social: an introduction to Actor-Network-Theory*. Oxford: Oxford University Press.

Law, J., & Callon, M. (1992). The life and death of an aircraft: A network analysis of technical change. In W. E. Bijker & J. Law (Eds.), *Shaping Technology/Building Society: Studies in Social Technical Change*. Cambridge, MA: MIT Press.

Lee, H., & Oh, S. (2006). A standards war waged by a developing country: Understanding international standard setting from the actor-network perspective. *Journal of Strategic Information Systems, 15*, 177-195

Laosethakul, K., & Boulton, W. (2007). Critical Success Factors for E-commerce in Thailand: Cultural and Infrastructural Influences. *The Electronic Journal on Information Systems in Developing Countries, 30*(2), 1-22.

Lerner, J., & Tirole, J. (2000). The simple economics of open source. Retrieved 12 May

2008, from http://www.nber.org/papers/w7600

Lerner, J., & Tirole, J. (2001). The open source movement: key research questions. *European Economic Review, 45*, 819 - 826.

Lerner, J., & Tirole, J. (2005). The scope of open source licensing. *The Journal of Law, Economics, & Organization, 21*(1), 20-56.

LTK. (2008). Actor-Network Theory (ANT) at Learning-Theories.com. Retrieved 30 March, 2008, from http://www.learning-theories.com/actor-network-theory-ant.html

Loney, M. (2002). Government body says developing countries need open source. Retrieved 3 May 2008, from http://news.zdnet.co.uk/itmanagement/0,1000000308,2122219,00.htm

Lungo, J. H. (2003). *Data Flows in Health Information Systems: An Action Research Study of Reporting Routine Health Delivery Services and Implementation of Computer Databases in Health Information Systems.* Unpublished Master Thesis, University of Oslo, Oslo.

Lyytinen, K., & Damsgaard, J. (2001). *What's wrong with the Diffusion of Innovation Theory. The case of a complex and networked technology.* Paper presented at the Proceedings of the IFIP 8.6. Conference, Banf 8-10 April 2001, Canada.

Madon, S. (2004). Evaluating the developmental impact of e-Governance initiatives: An Exploratory Framework. *The Electronic Journal on Information Systems in Developing Countries, 20*(5), 1-13.

Madon, S., Sahay, S., & Sahay, J. (2004). Implementing property tax reforms in Bangalore: an actor-network perspective. *Information and Organization, 14*(2004), 269–295.

Marwell, G., & Oliver, P. (1993). *The Critical Mass in Collective Action: A Micro-Social Theory*. Cambridge, UK: Cambridge University Press.

May, C. (2006). Escaping the TRIPs' trap: The political economy of Free and Open Source Software in Africa. *Political Studies, 54*, 123-146.

McGrath, K. (2005). Doing critical research in information systems: a case of theory and practice not informing each other. *Info Systems J, 15*, 85-101.

McLean, C., & Hassard, J. (2004). Symmetrical absence/symmetrical absurdity: Critical notes on the production of actor-network accounts. *Journal of Management Studies 41*(3), 493-519.

Meystre, S., & Müller, H. (2005). Open source software in the biomedical domain: Electronic health records and other useful applications. *Swiss Medical*

Informatics, 55, 3-15.

Mockus, A., Fielding, R., & Herbsleb, J. (2000). A case study of open source software development: the apache server. Paper presented at the Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland.

Myers, M., & Avison, D. (2002). An introduction to qualitative research in information systems. In M. D. Myers & D. E. Avison (Eds.), *Qualitative Research in Information Systems: A Reader*. London: Sage.

Myers, M. D. (1997). Qualitative research in information systems. *MIS Discovery* Retrieved 1 February, 2007, from MISQ Discovery, http://www.misq.org/discovery/MISQD_isworld/

Musa, P. F., Mbarika, V. W., & Meso, P. (2005). Calling for programmed technology transfer and adoption strategies for sustainable LDC Growth. *Communications of the ACM, 48*(12), 111-116.

Nfuka, E., & Rusu, L. (2007). Management of IT-enabled change in a public organisation in Tanzania. *International Journal Information Systems and Change Management, 2*(4), 334-349.

Ngotyana, B. (1973). The strategy of rural development. In L. Cliffe, P. Lawrence, W. Lutrel, A. Migot & J. S. Saul (Eds.), *Rural Cooperation in Tanzania*. Dar es Salaam: Tanzania Publishing House.

Nhampossa, J. (2006). *Re-Thinking technology transfer as technology translation: A case study of health information systems in Mozambique.* Unpublished Ph.D Thesis, University of Oslo, Oslo.

Nielsen, K. A., & Svensson, L. (Eds.) (2006). *Action research and interactive research: beyond practice and theory*. Netherlands: Shaker Publishing BV.

Nohria, N. (1995). Note on organization structure. *Harvard Business Review, May - June 1995*.

Nordal, K. (2006). *The challenge of being open - building an open source development network.* Unpublished Master Thesis, University of Oslo, Oslo.

Noronha, F. (2003). *Developing countries gain from Free/Open-Source Software*. Retrieved 16 February 2008, from http://www.linuxjournal.com/article/6884.

Nyerere, J. K. (1962). Ujamaa - The Basis of African Socialism. In J. K. Nyerere (Ed.), *Ujamaa - Essay on Socialism* (pp. 162-171). Dar es Salaam: Oxford University Press.

Nyerere, J. K. (1968). *Freedom and Socialism / Uhuru na Ujamaa - A Selection from the Writings and Speeches 1965-1967*. Dar es Salaam: Oxford University Press.

Orlikowski, W. (1993). CASE tools as organizational change: Investigating incremental and radical changes in system development. *MIS Quarterly, 17*(3), 309-339.

Orlikowski, W., & Baroudi, J. (1991). Studying information technology in organizations: research approaches and assumptions. *Information Systems Research, 2*(19), 1-28.

OSI. (2007). Open Source Licenses. Retrieved 24 November 2007, from http://www.opensource.org/docs/definition.php.

Patton, M. Q. (2002). *Qualitative evaluation and research methods* (3rd ed.). Thousand Oaks, CA: Sage Publications.

Perens, B. (2005). The Open Source Definition. Retrieved 1 November 2005, from http://perens.com/Articles/OSD.html

Popper, K. R. (1945). *The Open Society and its enemies.* London: Routledge.

Ramiller, N. C. (2005). Applying the sociology of translation to a system project in a lagging enterprise. *Journal of Information Technology Theory and Application, 7*(1), 51-76.

Raymond, E. (2001). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolution* (Rev. ed.). Beijing: O'Reilly & Associates Inc.

Raymond, E. (2003). The Jargon File, version 4.4.7. Retrieved 2 March 2008, from http://www.catb.org/jargon/

Reason, P., & Bradbury, H. (2001). *Handbook of Action Research*. London: Sage.

Riehle, D. (2007). The economic motivation of Open Source Software: Stakeholder perspective. *IEEE Computer Society* (April), 25-32.

Rogers, E. M. (2003). *Diffusion of Innovations* (5th ed.). New York: The free press.

Rolfsen, M., & Knutstad, G. (2007). Transforming management fashions into praxis: Action Research Project in AutoParts. *Action Research, 5*(4), 341-357.

Rosen, L. (2005). *Open Source Licensing: Software freedom and Intellectual Property Law*. Upper Saddle River, NJ: Prentice Hall.

Rossi, C. (2006). Comparing motivations of individual programmers and firms to take part in the Open Source Movement: From community to business. *Knowledge, Technology, and Policy, 8*(4), 40-64.

Samuelson, P. (2006). IBM's pragmatic embrace of Open Source. *Communication of the ACM, 49*(10), 21-25.

Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., & Lakhani, K. (2006). Understanding Free/Open Source Software Development Processes. *Software Process*

Improvement and Practice 11(2), 95 -105.

Schmitz, P.-E. (2001). A s*tudy into the use of open source software in the public sector*. Brussels: European Commission, Information and Documentation Centre.

Scott, J. C. (1976). *The moral economy of the peasant*. New Haven, CT and London: Yale University Press.

Selener, D. (1997). *Participatory action research and social change*. Ithaca, NY: Cornell University.

Shakermover. (2008). *Software as a Service*. Retrieved 3 March 2008, from http://en.wikipedia.org/wiki/Software_as_a_service

Sharman, D., & Yassine, A. (2004). Characterizing complex product architectures. *Systems Engineering Journal, 7*(1), 1-35.

Sheriff, S. (2007). *Rural access: Options and challenges for connectivity and energy in Tanzania*. Dar es Salaam: International Institute for Communication and Development (IICD).

Silverman, D. (2006). *Interpreting Qualitative Data* (3rd ed.). London: Sage Publications.

Smith, M., Madon, S., Anifalaje, A., Lazarro-Malecela, M., & Michael, E. (2008). Integrated health information systems in Tanzania: Experience and Challenge. *The Electronic Journal on Information Systems in Developing Countries, 33*(1), 1-21.

Sommerville, I. (2001). *Software Engineering* (6th ed.). Essex: Pearson Education.

Sowe, S. K., Stamelos, I., & Angelis, L. (2007). Understanding knowledge sharing activities in Free/Open Source Software projects: an empirical study. *The Journal of Systems and Software,* 81(3), 431-446.

Stalder, F. (2003). *Open Source as Social Principle*. Retrieved 2 February 2008, from http://felix.openflows.org.

Stallman, R. (2002). *Free software, free society: Selected essays of Richard M. Stallman*. Boston: Free Software Foundation.

Stanforth, C. (2006). Using Actor-Network theory to analyse e-government implementation in developing countries. *Information Technology and International Development, 3*(3), 35-60.

Taylor, M., & Singleton, S. (1993). The communal resource: transaction costs and the solution of collective action problem. *Politics and Society, 21*(2), 195-215.

Thanasankit, T., & Corbit, B. (2000). Cultural Context and its Impact on Requirements Elicitation in Thailand. *The Electronic Journal on Information Systems in Developing Countries, 1*(2), 1-19.

Tsuruta, T. (2006). African imaginations of moral economy: Notes on indigenous economic concepts and practices in Tanzania. *The Online Journal for African Studies, 9*(1 & 2). Retrieved 28 March 2008, from http://web.africa.ufl.edu/asq/v9/v9i1a8.htm.

TTCL. (2007). Tanzania Telecommunication Company Limited Broadband Packages and Pricing. Retrieved 23 November 2007, from http://www.ttcl.co.tz/Broadband_Pricing.asp

UN. (2004). *Road Maps towards an Information Society in Latin America and Caribbean (No. LC/G.2195/Rev.1-P).* Santiago: United Nations Economic Commission for Latin America and the Caribbean.

Villanueva, E. (2002). *Edgar Villanueva, Letter to Microsoft*. Retrieved 16 February 2008, from http://everything2.com/index.pl?node=Peruvian Congressman's Open Letter to Microsoft.

von Hippel, E. (2005). Open source software projects as user innovation networks. In J. Feller, B. Fitzgerald, S. Hissam & K. Lakham (Eds.), *Perspectives on Free and Open Source Software*. Cambridge: Massachusetts Institute of Technology.

von Hippel, E., & von Krogh, G. (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organisation Science, 14*(2), 209-223.

von Hippel, E., & von Krogh, G. (2006). Free revealing and the private collective model for innovation incentives. *R&D Management, 36*(3), 295-306.

Vuzo, M. S. (2007). *Revisiting the Language Policy in Tanzania: A Comparative Study of Geography Classes Taught in Kiswahili and English.* University of Oslo, Oslo.

Walsham, G. (1993). *Interpreting Information Systems in organizations*. Chichester: John Wiley & Sons.

Walsham, G. (1995a). The emergence of interpretivism in IS research. *Information Systems Research, 6*(4), 376-394.

Walsham, G. (1995b). Interpretive case studies in IS research: Nature and method. *European Journal of Information Systems, 4*(3), 74-81.

Walsham, G., & Avgerou, C. (2000). *Information Technology in context: Studies from the perspective of developing countries*. Aldershot: Ashgate.

Walsham, G., & Sahay, S. (2006). Research on Information Systems in Developing Countries: Current Landscape and Future Prospects. *Information Technology for Development, 12*(1), 7-24.

Weber, S. (2003). Open Source Software in developing economies. Retrieved 24 April 2007, from

http://programs.ssrc.org/itic/publications/ITST_materials/webernote2.pdf.

Weber, S. (2004). *The Success of Open Source*. Cambridge, MA: Harvard University Press.

Weerawarana, S., & Weeratunga, J. (2004). *Open Source in developing countries*. Stockholm: Edita Sverige AB.

WHO. (2004). *Developing Health Management Information Systems: A Practical Guide for Developing Countries*. Geneva: World Health Organization.

Whyte, W. F. (1993). *Participatory Action Research*. Newbury Park, CA: Sage.

Whyte, W. F., Greenwood, D. J., & Lazes, P. (1991). Participatory Action Research: Through practice to science in social research. In W. F. Whyte (Ed.), *Participatory Action Research* (pp. 19-55). Newbury Park, CA: Sage.

Wichmann, T. (2002). *Use of Open Source Software in firms and public institutions: Evidence from Germany, Sweden, and UK*. Berlin: International Institute of Infonomics.

Williams, S. (2002). *Free as in freedom: Richard Stallman's crusade for free software*. Cambridge: O'Reilly.

Wong, K., & Sayo, P. (2004). *Free/Open Source Software: A General Introduction*. Kuala Lumpur, Malaysia: UNDP-APDIP.

Wood-Harper, T., & Bell, S. (1990). Information systems development for developing countries. In S. C. Bhatnagar & N. Bjørn-Andersen (Eds.), *Information Technology in developing countries* (pp. 23-39). North-Holland: Elsevier Science Publishers.

ZALONGWA. (2007). Current Customers of Zalongwa Student Academic Register Information System. Retrieved 27 September 2007, from http://www.zalongwa.com/customers/