Global Design for Secure Socio-Technical Systems

Tong Li
Supervisor: John Mylopoulos, Fabio Massacci
University of Trento
via Sommarive 14, Povo(TN), Italy
tong.li@disi.unitn.it

Abstract.

Socio-Technical Systems (STS) consist of people, software, hardware and organizational units. The pervasiveness and complexity of STSs make security analysis both particularly challenging and especially critical. Traditional security analysis techniques that address security in a piecemeal fashion (e.g. only for software, or only for business processes) are insufficient for addressing global security concerns and have been found often to leave serious STS vulnerabilities untreated. In this proposal, we aim at developing a comprehensive framework that consists of concepts, techniques and tools for designing secure STSs. In our framework, a STS consists of organizational goals and security requirements, businesses and industrial processes through which requirements are satisfied, software applications that support those processes, and system infrastructure that supports both processes and applications. We intend to propose a systematic process to analyze and design each part of the STSs, and finally provide an all-round security design for STSs.

1 Introduction

The rapid developments of software systems, coping with information demands, have automated much of the manual tasks. There is no denying that the software systems are playing an increasingly significant role in large systems. Socio-Technical Systems (STS), as studied by Ropohl [18], introduce concepts which view systems from two perspectives—social and machine, stressing the reciprocal relationships between these perspectives. As STSs consist of many interweaving human and technical components, to design a STS, many complex sub-parts must be analyzed and well designed. For example, system organizational settings, business processes, software and infrastructures. A system can function correctly only if all the sub-components are well designed.

Security is a crucial and difficult issue for almost all systems. Ensuring security in STSs has been proved to be especially complicated [4], as addressing security means not only considering technical aspects of software, but also social aspects, such as the design of a business process. For example, if a business process, which issues customer orders, does not contain a step to check the authenticity of the order, the whole system will be vulnerable regardless of the security of other components. Moreover, the social aspect of a STS also requires to consider how do people interact with a STS. For example, if a security mechanism is too complex, stakeholders may close or bypass it, which leads the system to be vulnerable. In short, security protections for STSs are subject to the Bucket Effect: applying intensive security measures to a single component can not guarantee the security of the whole system.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

Unfortunately, most existing approaches address security requirements in a piecemeal fashion, merely considering security issues for a particular narrow part, i.e. software, physical infrastructure, or business process. We argue that the security designs of different parts in STSs may influence each other in certain ways. Currently, there are no formal approaches to connect all the design concepts in different system components and analyze their mutual impacts. Thus, no systematic, global means are proposed to design fully secure STSs.

To address this problem, we plan to develop a framework which consists of concepts, processes, and tools for designing secure socio-technical system by connecting different components with each other. This framework contributes in following aspects: 1) explicitly capture and represent causal relationships among different parts of a system to support the design in each part; 2) propose a systematic process to produce secure designs for each component — the synthesis of these designs is treated as the system design, which fulfills both organizational objectives and security requirements; 3) provide a systematic means to handle system changes, while continuously satisfy both organizational objectives and security requirements.

In the remaining part of this paper, we summarize the state of the art in Sec. 2, based on which we describe concrete research topics in Sect. 3. Sect. 4 shows our research approach that deals with the identified problems. Finally, the contribution of our work is summarized in Sect. 6.

2 Related Work

In this section, we will summarize related work on the topic of system security analysis and design. We are able to categorize many of these works according to the analytical perspectives they hold. Specifically, we consider following four perspectives: organizational settings, business processes, software applications and physical infrastructure. Work which falls outside of these classifications are discussed in a separate subsection.

There are a number of research proposals which focus on ensuring security of organizational settings through modeling interactions among social actors, as well as some social relationships (e.g. trust, delegation, authorization) [15, 7, 11].

Security analysis which focuses on business processes aims to represent the security requirements in business process models. Analysts can use these requirements to further design secure business processes [17, 9].

Most security efforts have focused on the technical aspects of software. There are many security analysis techniques which can represent either security attacks or security requirements, such as Abuse Case [12], Attack Tree [19]. In addition, there has been much investigation into systematic processes for security requirement engineering. For example, SQUARE [13] and SREP [14]. Moreover, security design methodologies are used to align software developments with security requirements, which capture a lot of security domain knowledge. Examples of security design methodologies include UMLsec [10] and Security Patterns [20].

Currently, there are few security approaches that focus on the infrastructure domain. Jürjens considers secure deployments in UMLsec [10] , which address security issues in the physical layer.

Mouratidis and Jürjens connect security designs between the organization perspective and the software perspective by combining Secure Tropos and UMLsec [16]. They provide a structured process to translate the results of organizational security requirements to pertinent software design elements to support secure software developments. However, they only transfer security requirements from the organization layer to the software layer, and do not provide feedback in the other direction.

Finally, there is another research branch that address human factors in STSs. Tarimo et al. [21] discuss the importance of security culture in the security analysis of STSs. They exploit the capability of organizational behavior theory, and explain how security cultures influence security requirements of STSs. In addition, Beautement et al. [3] argue that system stakeholders do concern with the costs and benefits introduced by security mechanisms, and a lot of security breaches arises due to overlooking the intentional factors of human. Thus, they propose Compliance Budget to understand human behaviors and further address the security balance problem.

3 Research Problem

Compared to technically-focused software systems, the socio-technical systems extend their system boundaries to include several related components, namely organization, business process and infrastructure. Although the proposed components may overlap with each other to some extent, we use them to ensure security designs cover all important aspects of STSs. Our work aims at analyzing all these essential components and investigating connections between them.

Organization component includes high-level system requirements. The concerns about organizational settings (e.g. the organizational hierarchies and policies) are considered in this domain.

Business process component is considered the core part of STSs, as all organizational requirements are met through business processes, and as business processes provide instructions for both humans and software on how to work together to achieve specific goals.

Software component provides concrete implementations for supporting the business process. In the meanwhile, it relies on the deployment of infrastructure.

Physical infrastructure component is the backbone of software applications. The performance of software applications highly depends on related infrastructures. In addition, they may also play a role in some activities in the business process.

Human factors are the major issues in STSs. Instead of addressing human influences in an independent dimension, we consider them together with each of the above components. We aim to consider human reactions together with related system designs, which facilitate understanding human intentions.

Based on the above components, our work will take the organizational goals and security requirements, which are in the organization component, as the inputs. While the outputs are a set of designs in the three left components, which work together to ensure security of the systems. In this sense, we define the designs in the software component as Software Requirement Specifications(SRS); the designs in business process component are Business Requirement Specifications(BRS) and the designs in infrastructure component are Infrastructure Requirement Specifications(IRS). Then the synthesis of all these designs is treated as the system specification, which satisfies both organizational goals and security requirements. Thus, the design of STS is defined as:

$$\text{System Design} = \text{BRS+SRS+IRS}$$

Current studies only take into account one of these parts and leave assumptions on other parts, which may not be held in the target system. We argue that these three parts of system designs are highly involved with each other. Without a global view on all the parts, the system under development will be vulnerable.

As a result, my research problem is summarized as developing a framework of concepts, processes, and tools for globally designing secure socio-technical systems by synthesizing designs in three different components, namely business process, software and physical infrastructure. Specifically, there are four research questions need to be addressed: 1) what are the interrelationships among the above components; 2) how to formally orchestrate analysis in different components to provide global security designs; 3) how to derive secure designs in each component; 4) how to adapt designs to handle system changes.

4 Research Approach

In this section, we propose a research approach, which addresses the security problems in each system component, in order to derive global secure designs for STSs.

4.1 Concepts of Secure STS

To specify system designs, we apply the following three concepts *Requirements (R), Specification (S) and Domain assumption (D)*, proposed by Zave and Jackson [22]. They have been formalized as follows: $D, S \vdash R$. For a specific domain this formula implies that the requirements can be satisfied by the specifications under relevant domain assumptions. A number of existing works make use of this conceptualization and formalism [8], but all of them take the machine domain by default. Our work aims to consider each component of STSs as a single domain. Each of them has its only requirements, domain assumptions, and specifications. In addition, we separate $SecurityRequirement(SR)$ from the concept $Requirement(R)$ in order to stress the security analysis in our research. As a result, the concepts can be represented by the formula $D, S \vdash R, SR$. This formula is further specified for each of the proposed components. For instance, $D_{bp}, S_{bp} \vdash R_{bp}, SR_{bp}$ represent the relevant concepts in the business process component. We structure the three components in a hierarchical way (Fig.1), which answers the research question 1.

Concretely, as shown in Fig.1, the organizational requirements are imported to the business process domain; the design of the business process domains specify the requirements for the software domain, as well as the infrastructure domain; the software domain should support business process, in the meanwhile shed light on the requirements for the infrastructure domain; finally, the infrastructure domain should support the design in other two domains. We use Tropos [5], a goal-based requirement modeling language, to represent requirements in all the layers. The specifications in all three layers are represented by BPMN [2], UML activity diagram and UML

deployment diagram [1] respectively. All of these models will be further extended with related security notations. In order to explicitly represent the semantics of each concepts, as well as their interrelationships, we aim to develop a formal ontology with Description Logic to capture all related knowledge (both system development and security engineering).

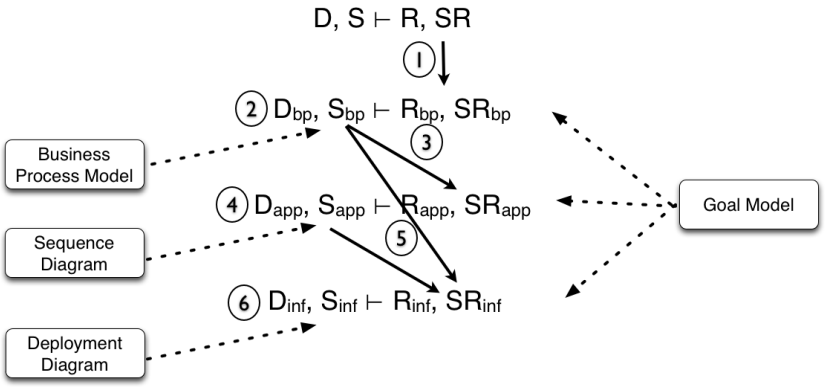

Fig. 1. Process for multi-layer system design

4.2 Secure STS Design Process

Based on the comprehensive ontology we are planning to build, we will further propose an intuitive process to orchestrate design processes in different layers(Fig.1), which address the research question 2. In each layer, its requirements are influenced by designs of other layers. In this process framework, analysis is conducted from the business process layer to the software layer, finally to infrastructure layer. It is worth noting that the proposed process is not like waterfall model, which is analyzed once for all, but an iterative process that incrementally addresses system designs. Through these analytical processes, the traceability links between layers could be derived, which are the key issues for global system designs.

The traceability links are built from specifications in one layer to requirements in other layers, not to specifications in other layers. In this sense, we do not directly impose designs from one layer to another, which indicates designs in each layer are only subject to requirements of that layer. The benefits from this separation include: 1) the requirement model plays as the mediator to connect designs in different do- mains, which clearly separate problems and solutions. By using the intermediary requirement model, the many-to-many mapping relationships between designs that are in different layers could be simplified as one-to-many relationships, which reduce the complexity of the design work and increase the adaptability of the system design. 2) By using the goal models to represent requirements, we can exploit the inherent advantages of the goal-based analytical techniques to facilitate the design work, such as the analysis of requirement satisfaction and alternative selection. As a result, this proposed framework could be used not only for developing new systems, but also for developing systems that are based on legacy systems. If existing designs of a legacy system cannot satisfy requirements imposed by previous layers, the analysis will backtrack to the previous layers and try to find another alternative.

There are six tasks identified in the Fig. 1, which outline the process for deriving an all-round system design. Concretely, the tasks 1,3 and 5 focus on how to derive general requirements and security requirements from upper layers. To this end, mapping mechanisms among layers will be defined to assist the derivation of requirements. The tasks 2,4 and 6 focus on how to derive secure designs that fulfill both general requirements and security requirements. We intend to carry out risk-based security analysis processes to derive security designs by adopting security patterns. This general analytical process will be customized for each layer according to their domain specific features. By introducing such a design process we satisfy the research question 3.

4.3 Evolution of Secure Design

Changes are inevitable for most systems, and these changes must be handled in a way that ensures the continued satisfaction of system requirements. The essence of our work is to combine concepts in multiple layers, based on which we are able to identify designs in different layers which are effected by change requests.

In front of changed requirements, we will incrementally evolve the system rather than re-design from scratch. Thus, appropriate strategies should be applied to promote the incremental design process. Referring to the

work[6]. We consider three kinds of approaches, namely minimal efforts, minimal changes and maximal familiarities.

4.4 Approach Illustration

The expected contribution of this research is providing a systematic means to globally design a secure STS. In this part, we use a simple example to show the ideal system development process. A smart grid real-time pricing scenario is used as an example for illustration. Due to the space limitation, we only focus on a particular part of this example, which is highlighted by dotted rectangles.

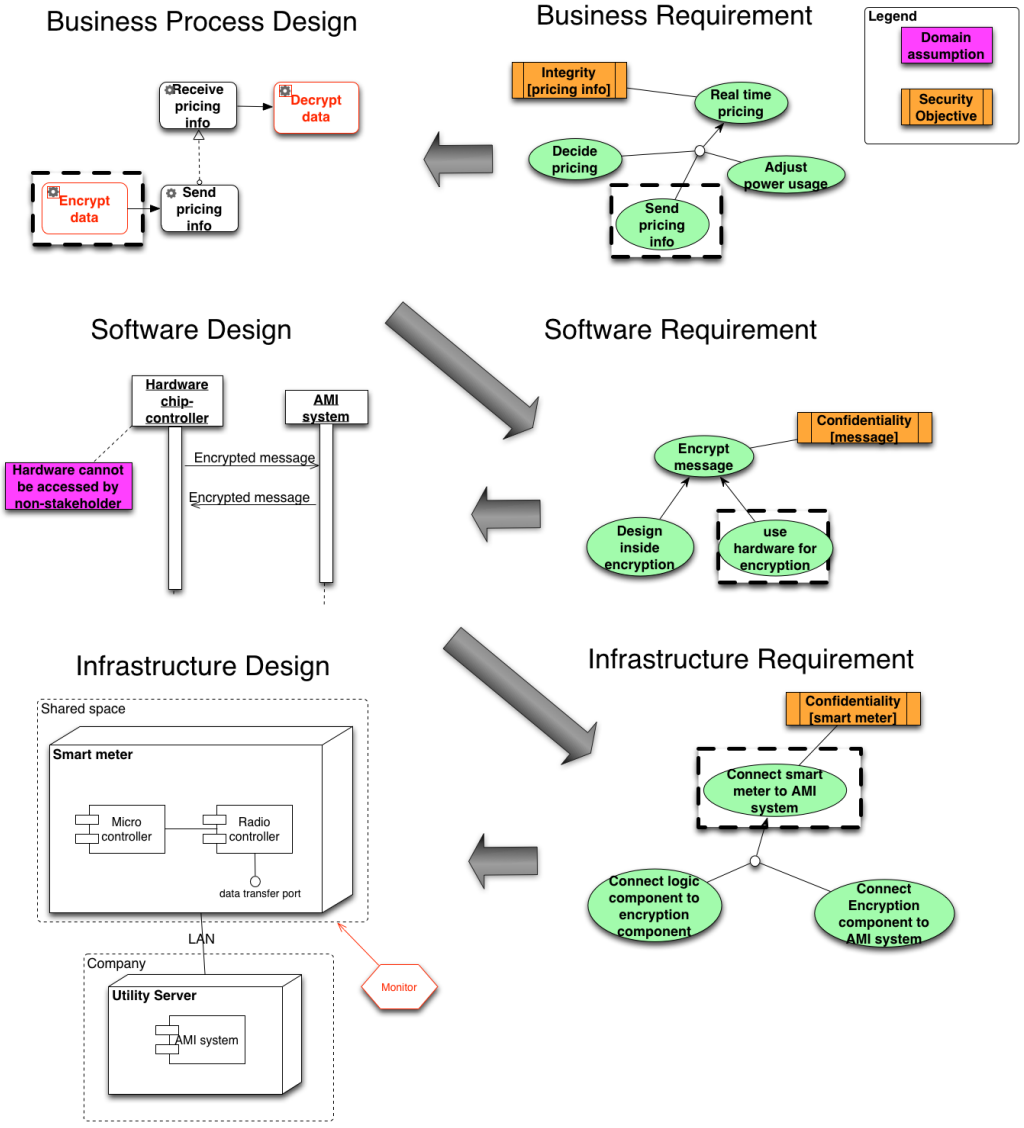

Fig. 2. Illustration of global design for secure STS

The whole design process is shown in Fig.2, which stick to the process described in Fig.1. Except for the notations introduced by existing work(e.g. Tropos, BPMN, UML), we add another two concepts, domain assumption and security objective, into the design process. Domain assumption represents the assumption that are taken during system design, and security objective represents security properties that have to be held during a goal achievement process. Fig.2 shows an intuitive order for system design, which could be conducted backwards and iteratively as necessary.

When there is a change happens at arbitrary part of the system design, the system will evolve accordingly to continuously satisfy the organizational goals. Fig.3 represents a redesign process via traceability links. Concretely, the change first cause a backwards propagation as shown in the upper part of Fig.3. The propagation will be

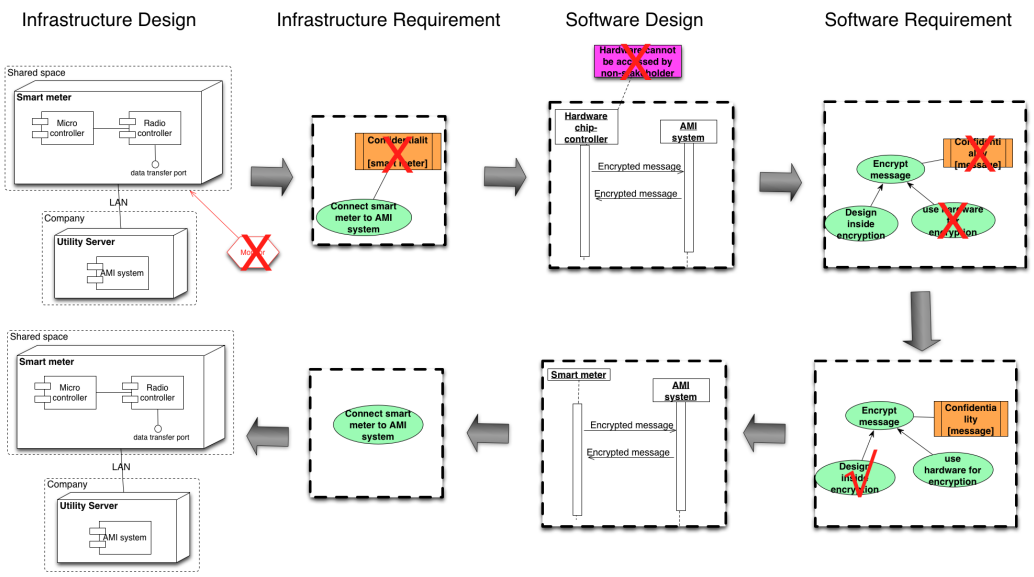

Fig. 3. Illustration of a system evolution according to a change

stopped until the change could be handled, e.g. find an alternative way to replace the unsatisfied requirement. After that, the system will forward the new design to related parts to "implement" it.

5 Evaluation Plan

Our framework aims to support the secure system design for socio-technical systems, and it is supposed to provide an all-around solution within comparatively efficient processes. To ensure the effectiveness and efficiency of the proposed framework, we should follow an evaluation methodology to examine the practical value of our work.

Tool support We are intended to develop a CASE tool, which can support all the steps of our analytical framework. Enhanced by this tool, the system design process could be done in a semi-automatic way. For the task 3 and 5, the tool can analyze the designs in upper layers and generate potential requirements, which will be further confirmed and modified by human. For task 2, 4 and 6, the tool will assist the security analysis processes to derive security design in each layer. In addition, the tool is able to analyze the ripple-effects of specific changes. All the design fragments influenced by the changes will be presented by the tool, and the customer will take the final decisions about which part should be changed.

Case study We will apply our design methods to a real socio-technical system, such as smart grid and electrical healthy system. The key point of the case study is to take out the theoretical method from laboratory to practice in order to check whether it works in reality. To this end, a number of factors need to be considered during the case study, such as the background of the method adopter, the time span for learning the method, the help or guide provided by the method designer. We need to exploit the influential factors as many as possible to provide an in-depth evaluation for our method. In addition, if it is possible we are planning to compare our method with other related work on the same case study to examine the advantages and disadvantages of our method.

6 Contribution

In summary, satisfaction of the proposed research goals via execution of the described plans will contribute to the state of the art by providing:

1. A conceptual multi-layer framework to understand the secure socio-technical system in a comprehensive way, specifically to understand the interactions among different system components.
2. A methodology to globally design secure socio-technical systems, which satisfy both organizational objectives and security requirements, either based on an existing system or for a new system. The methodology is aimed to enable the evolution of system design to manage requested changes, while continuously fulfilling organizational objectives and security requirements.
3. A tool that supports all the analytical steps of the proposed methodology. The tool will allow us to carry out an evaluation to examine the practical value of our work.

References

1. Unified Modeling Language. http://www.omg.org/spec/UML/2.1.2/, 2003.
2. Business Process Modeling Notation. http://www.omg.org/spec/BPMN/2.0/, 2011.
3. A. Beautement, M. A. Sasse, and M. Wonham. The compliance budget: managing security behaviour in organisations. In *Proceedings of the 2008 workshop on New security paradigms*, NSPW '08, pages 47–58, New York, NY, USA, 2008. ACM.
4. R. P. Bostrom and J. S. Heinen. STS Perspective MIS Problems and Failures : A Socio-Technical Perspective PART I : THE CAUSES. *MIS Quarterly*, 1(3):17–32, 1977.
5. P. Bresciani, A. Perini, P. Giorgini, and F. Giunchiglia. Tropos: An agent-oriented software development methodology. *Agents and Multi-Agent*, 2004.
6. N. A. Ernst, A. Borgida, and I. Jureta. Finding incremental solutions for evolving requirements. *Proceedings of 19th IEEE International Requirements Engineering Conference (RE)*, 0:15–24, 2011.
7. P. Giorgini, F. Massacci, and N. Zannone. Security and trust requirements engineering. In *Foundations of Security Analysis and Design III*, volume 3655 of *Lecture Notes in Computer Science*, pages 237–272. 2005.
8. C. Haley, R. Laney, J. Moffett, and B. Nuseibeh. Security requirements engineering: A framework for representation and analysis. *Software Engineering, IEEE Transactions on*, 34(1):133–153, 2008.
9. P. Herrmann and G. Herrmann. Security requirement analysis of business processes. *Electronic Commerce Research*, 6(3-4):305–335, Oct. 2006.
10. J. Jürjens. UMLsec: Extending UML for Secure Systems Development. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, volume 2460 of *Lecture Notes in Computer Science*, pages 412–425. 2002.
11. L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proceedings IEEE international conference on requirements engineering (RE'03)*, volume 3, pages 151–161, Monterey, California, 2003.
12. J. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *15th Annual Computer Security Applications Conference, (ACSAC'99) Proceedings.*, pages 55–64. IEEE, 1999.
13. N. R. Mead and T. Stehney. Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes*, 30(4):1, July 2005.
14. D. Mellado, E. Fernández-Medina, and M. Piattini. A common criteria based security requirements engineering process for the development of secure information systems. *Computer Standards & Interfaces*, 29(2):244–253, Feb. 2007.
15. H. Mouratidis and P. Giorgini. A natural extension of tropos methodology for modelling security. In *the Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002)*, Seattle-USA,, 2002. Citeseer.
16. H. Mouratidis and J. Jurjens. From Goal-Driven Security Requirements Engineering to Secure Design. *International Journal of Intelligent System*, 25(8):813–840, 2010.
17. A. Rodriguez, E. Fernandez-Medina, and M. Piattini. A BPMN Extension for the Modeling of Security Requirements in Business Process. *IEICE transactions*, E90-D(4):745–752, 2007.
18. G. Ropohl. Philosophy of socio-technical systems. *Society for Philosophy and Technology*, 4(3):59–71, 1999.
19. B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.
20. M. Schumacher. Security patterns and security standards. In *Proceedings of the 7th European Conference on Pattern Languages of Programs (EuroPLoP), July*, 2002.
21. C. N. Tarimo, J. K. Bakari, L. Yngström, and S. Kowalski. A social-technical view of ict security issues, trends, and challenges: Towards a culture of ict security - the case of tanzania. In *ISSA*, pages 1–12, 2006.
22. P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, Jan. 1997.