A framework for usable and secure system design

Shamal Faily

Wolfson College

University of Oxford

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Hilary Term 2011

Abstract

Despite existing work on dealing with security and usability concerns during the early stages of design, there has been little work on synthesising the contributions of these fields into processes for specifying and designing systems. Without a better understanding of how to deal with both concerns at an early stage, the design process risks disenfranchising stakeholders, and resulting systems may not be situated in their contexts of use.

The research problem this thesis addresses is how techniques and tools can be integrated and improved to support the design of usable and secure systems. To develop this understanding, we present IRIS (Integrating Requirements and Information Security) — a framework for specifying usable and secure systems. IRIS considers the system design process from three different perspectives — Usability, Security, and Requirements — and guides the selection of techniques towards integrative Security, Usability, and Requirements Engineering processes.

This thesis claims that IRIS is an exemplar for integrating existing techniques and tools towards the design of usable and secure systems. In particular, IRIS makes three significant contributions towards the stated research problem. First, a conceptual model for usable secure Requirements Engineering is presented, upon which the IRIS framework is founded; this meta-model informs changes to elicitation and specification techniques for improved interoperability in the design process. Second, several characteristics of tool-support needed to elicit and specify usable and secure systems are introduced; the CAIRIS (Computer Aided Integration of Requirements and Information Security) software tool is presented to illustrate how these characteristics can be embodied. Third, we describe how the results of applying IRIS can be used to improve the design of existing User-Centered Design techniques for secure systems design.

We validate the thesis by applying the IRIS framework to three case studies. In the first, IRIS is used to specify requirements for a software repository used by a UK water company. In the second, IRIS is used to specify security requirements for a meta-data repository supporting the sharing of medical research data. In the final case study, IRIS is used to analyse a proposed security policy at a UK water company, and identify missing policy requirements. In each case study, IRIS is applied within the context of an Action Research intervention, where findings and lessons from one case study are fed into the action plan of the next.

Acknowledgements

This doctoral research was funded by EPSRC CASE Studentship R07437/CN001. I am very grateful to EPSRC and QinetiQ Ltd for the sponsorship of this work. I am also grateful to the Department of Computer Science for additional sponsorship of fieldwork and conference-related activities.

I would like to give special thanks to my supervisor, Dr Ivan Fléchais, for his advice and encouragement while carrying out this research. I am also grateful to members of the departmental security reading group for their feedback and comments on early drafts of publications arising from this thesis, and the many reviewers and attendees that provided insightful comments at workshops, conferences, and symposia where IRIS-related papers and posters were presented.

During my time at Oxford, I have been a member of Wolfson College, which has supported me financially in the form of travel funding and book acquisitions; I am proud to say that the Wolfson College library now has, at this point in time, the best collection of scholarly texts on the subject of usable and secure system design in Oxford! I would also like to thank my colleagues on the Entertainment Sub-Committee, Bar Sub-Committee, and Wine Society, as well as my past and present friends at Wolfson for the welcoming distraction their friendship provided.

Finally, I would like to thank my parents for their patience, support, and love.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we motivate and introduce the research problem, and identify the gap in the research this dissertation addresses. We then propose a research question, the answer to which contributes knowledge helping to close this gap. Based on this question, we present the main claim made by this thesis, and justify the research approach to be adopted. We conclude this chapter by presenting an overview of this dissertation, followed by its detailed structure.

1.1 Thesis motivation

The effect of Information Technology on our lives can be seen all around us. In terms of economic growth, a recent report commissioned by Google estimated that the UK's internet industry was worth £100 billion [148]. The increasing ubiquity of technology has also led academic researchers to re-think our interaction with it. In a comparatively recent Communications of the ACM article [221], several leading Human Computer Interaction (HCI) researchers noted that our relationship with computers has changed so radically since the field's inception that even the term *HCI* needs a rethink. In the past, we could reasonably assume that IT involved desktop computers and *users* in commercial organisations. Nowadays, systems are as ubiquitous as the people who use it, who are increasingly connected with different people and other systems. In such a connected world, the users of technology have incalculable opportunities to intentionally or unintentionally interact with a myriad of systems.

One question which has yet to be answered is how much Information Technology has empowered the work of those who build it. Media reports about the growth of high technology industry go almost hand-in-hand with reports about the impact of threats to it. For example, a recent report commissioned by the UK government estimated that the cost of cyber crime to the UK economy is £27 billion per annum [71]. While the methodologies used to devise this figure are debatable, the increased burden of expectation on system designers is not. As consumers, we expect systems to be attuned to the physical and social contexts within which they are used. As a corollary, we would also like our systems to be as secure as they are usable but, as we have discovered, threats to, and vulnerabilities within, this complex network of people and technology make this a challenging task for system builders.

Systems can be made vulnerable through a variety of factors, ranging from the accidental introduction of incorrect code, through to an overly complex user interface which may be misused or circumvented. Those who might take advantage of these vulnerabilities have capabilities and motivations which may be unknown to the designers who inadvertently introduced them, together with different abstractions for what the vulnerabilities are and how they can be exploited. So, while our expectations for technology innovation continue to be exceeded, the quality of these systems' security and usability often falls short.

There is no obvious reason why designing secure and usable systems should be so difficult, especially when guidance on applying Security and Usability Engineering best practice is no longer restricted to the scholarly literature. Nielsen claimed that cost was the principal reason why Usability Engineering techniques are not used in practice [177], but the financial costs of applying such techniques have been reduced by technology advances. Similarly, practical techniques for identifying and mitigating security problems during system design are now available to developers in an easy to digest format, e.g. [212, 238].

Problems arise when considering how to use these approaches as part of an integrated process. Accepted wisdom in Software Engineering states that requirements analysis and specification activities should precede other stages in a project's lifecycle [109]. However, Information Security and HCI proponents argue that their techniques should instead come first. For example, ISO 13407 [137] states that activities focusing on the collection of empirical data about users and their activities should guide early design, but security design methods such as [100, 70] suggest that such stages should be devoted to high-level analysis of the system to be secured. Invariably, the decision of what concern to put first is delegated to the methodology followed by a designer. The designer has many approaches to choose from, some of which include treatment for security or usability concerns. To date, however, no approach treats both security and usability collectively, beyond treating them both as generic qualities contending with functionality.

When weighing up the approaches available, and the effort needed to apply them in their developmental contexts, designers may even choose to simply ignore them. Designers may believe that their knowledge about user goals and expectations negate the need for applying usability design techniques, or their understanding of the system's risks and mitigating controls negates the need for security analysis. In such cases, developers may believe Security and Usability Engineering approaches are useful, but they don't believe the pay-off justifies their cost.

There is growing evidence that the design of usable and secure systems is worthy of specific attention. The US Department of Homeland Security ranked usable security as one of the top cyber-security research topics for governments and the private sector [162], and HCI-Sec (HCI for Security) continues to be an active research topic. Researchers are also beginning to consider how developers should build secure and usable systems [102], yet there remains a lack of guidance available to designers about how to design usable systems at a sufficiently early stage in the design process. Fortunately, despite the lack of guidance, the Security Requirements Engineering and HCI communities have proposed a number of individual techniques forming the basis of integrated design approaches. In theory, specifying and designing secure and usable systems involves carefully selecting the right techniques from each community. In practice, each technique is founded in different, potentially conflicting, conceptual models. The level of tool-support for these techniques also varies considerably, and there has been little work on integrating these tools and the conceptual models which underpin them.

The knowledge gleaned integrating design techniques and tools also leads to research contributions beyond the design of usable and secure systems. While there are academic fora devoted to integrating security and software engineering activities, e.g. [225], and HCI and security activities [252], there has been little work describing how usability design techniques can be usefully applied to designing secure systems. It is, therefore, possible that the results of integrating design techniques and tools may lead to design innovation in this area.

1.2 Research question

The thesis is motivated by the research question: **How can existing techniques and tools be integrated and improved to support the design of usable and secure systems?**

This research question can be broken down to the following questions.

- RQ1: Which concepts from Usability, Security, and Software Engineering can be harmonised to support the design of secure and usable systems?
- RQ2: What are the characteristics of tool-support needed to support the design of secure and usable systems?
- RQ3: How can User-Centered Design techniques be improved to support the design of usable and secure systems?

This dissertation presents the IRIS (Integrating Requirements and Information Security) framework. This framework constitutes a conceptual model for usable secure Requirements Engineering (Chapter 4), a process framework guiding the selection of design techniques for IRIS design processes (Chapter 5), and a software tool for supporting these processes (Chapter 6).

The principal claim made by this thesis is as follows: **The IRIS framework is an exemplar for integrating existing techniques and tools towards the design of usable and secure systems**.

In Software Engineering, the term *system design* encompasses a broad range of activities; these range from scoping an early vision of what a system needs to do, through to developing models of software components and interfaces. We, therefore, limit our focus on design to the early stages of a system's development for two reasons. First, the term *design* often refers to a plan or an outline of work, upon which a structure is built [2]; agreeing and specifying the nature of this plan is both required, and something best carried out as early as possible. Second, as Chapter 3 of the dissertation will discuss, each discipline contributing techniques to the design of usable and secure systems argues for their own approaches preceding all others. Consequently, we believe there is value in exploring how early design techniques interoperate together.

1.3 Dissertation overview

This dissertation is broken down into the subsequent nine chapters. Chapters 2 and 3 situate the dissertation and present the tools used to develop and evaluate the thesis. Chapters 4, 5, and 6 present the IRIS framework, which is validated by the case studies described in Chapters 7, 8, and 9. Finally, in Chapter 10, we review and evaluate the thesis.

Figure 1.1 provides a graphical roadmap of this dissertation, and describes how its structure maps to the contributions, and how these appeal to the research questions posed in Section 1.2.

1.4 Dissertation structure

Chapter 2 reviews the current state-of-the-art in the design of usable and secure systems. We consider existing work from the HCI Security community, and its limitations, before reviewing prevalent HCI concepts relevant to the design of secure systems. Given the dissertation's focus, we review several Requirements Engineering approaches from a security and usability perspective, before reviewing existing design *frameworks* for eliciting security and usability requirements. We conclude the chapter with a brief review of the available tool-support for facilitating Usability and Requirements Engineering activities.

Chapter 3 describes the methodology used for carrying out the research justifying this thesis. We describe the design research approaches taken by the HCI, Information Security, and Requirements Engineering communities before presenting the research methods employed. We conclude this chapter, by describing how these research methods tackle the different contributions associated with this dissertation, and introduce the NeuroGrid specification exemplar; this is used to illustrate the theoretical contributions arising from the thesis.

Chapter 4 presents a conceptual model for usable secure Requirements Engineering. This work builds upon practical work in usability design, and recent research on meta-models for Security Requirements

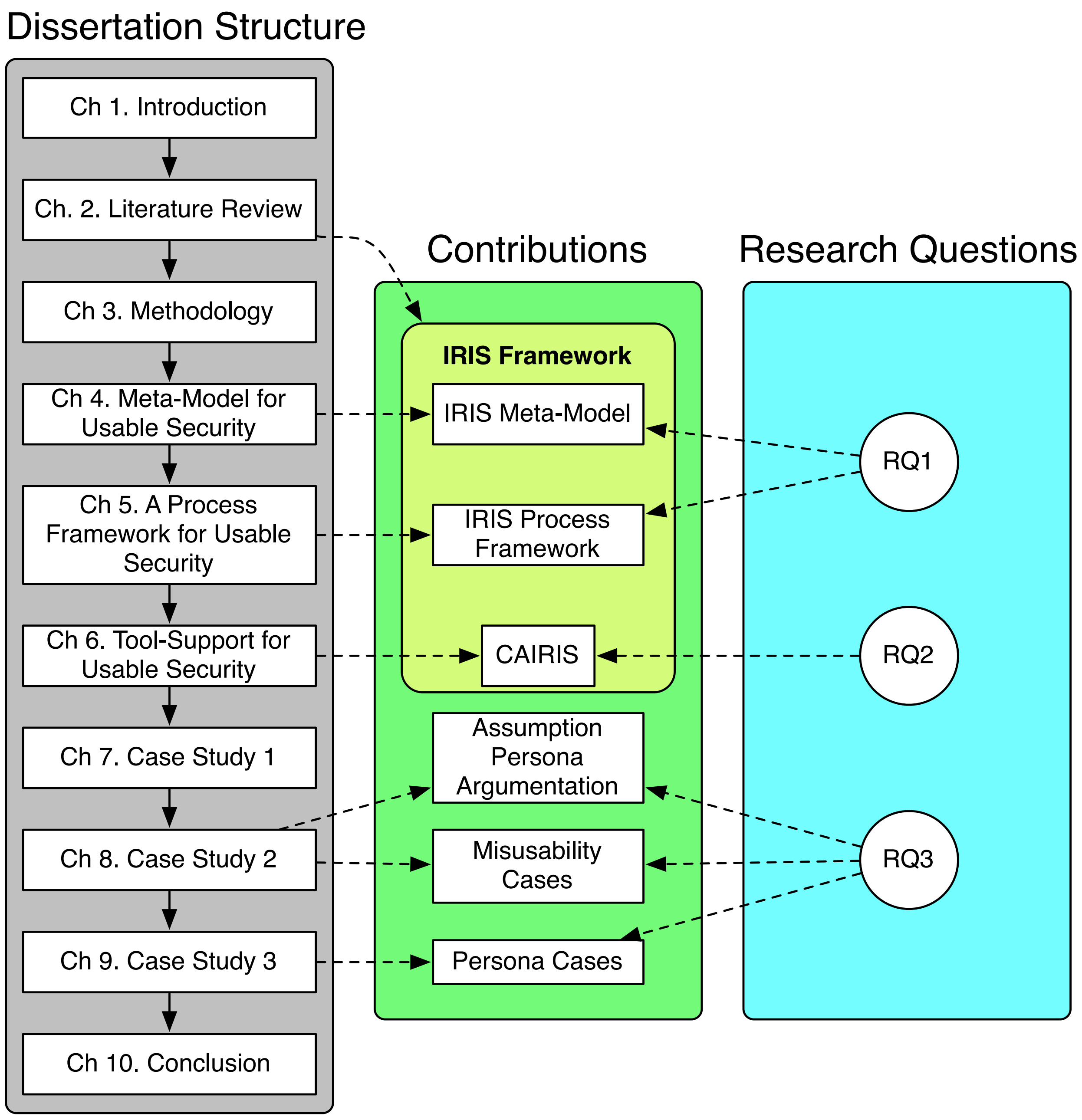


Figure 1.1: Dissertation roadmap

Engineering. Collectively, the meta-model concepts help structure and manage Usability, Security, and Requirements Engineering activities in different contexts. After presenting a brief overview of the conceptual model itself, we sub-divide the model explanation into six different views: Environment, Asset, Task, Goal, Risk, and Responsibility. For each view, we present and justify the related concepts and their relationships.

Chapter 5 introduces a process framework for selecting techniques when specifying usable and secure systems. Building on the meta-model described in Chapter 4, we present and describe the different *perspectives* of IRIS. Finally, we propose a number of exemplar techniques for each perspective.

Chapter 6 presents CAIRIS (Computer Aided Integration of Requirements and Information Security), a software tool designed to embody the characteristics needed to support the IRIS framework. We briefly discuss the principles motivating the design of CAIRIS before presenting its high-level architecture. We then present several characteristics of tool-support for the specification of usable and secure systems, and illustrate how CAIRIS supports these.

In Chapter 7, we present the first of three case studies used to validate the IRIS framework. We report how IRIS was used to specify requirements for a software repository managing control software used by a UK water company. We begin the chapter by introducing the methodology used, before describing how IRIS was applied, and evaluating the results of this application. Finally, we describe learning action points for subsequent case studies.

Chapter 8 reports on the second validating case study, where IRIS was used to specify security requirements for a portal facilitating the sharing of medical study data. Building upon the findings in Chapter 7, we also present a new technique for better aligning personas with the design of secure and usable systems. This technique, *Assumption Persona Argumentation*, describes how structured argumentation can be used to support the development and evolution of assumption personas. Building upon both the IRIS meta-model and this argumentation structure, we present a further technique called *Misusability Cases*, scenarios which describe how design decisions lead to usability problems subsequently leading to system misuse. After motivating and introducing the methodology followed, we describe how IRIS was applied, and evaluate the results of this application. Finally, we describe learning action points for the final case study.

Chapter 9 reports on the final validating case study, where IRIS was used to analyse the security and usability impact of a security policy for control system software at a UK water company. This analysis was used to identify missing security policy requirements. Building upon the findings of Chapters 7 and 8, we also present a new technique — *Persona Cases* — which builds upon ideas developed in Chapter 8 to develop personas both grounded in, and traceable to, their originating empirical data. Like the previous case study chapters, we motivate and describe the approach taken, before evaluating and reflecting on how these results inform future research.

In Chapter 10, we synthesise the findings gleaned from developing and applying the IRIS framework. We highlight specific contributions made by IRIS towards the design of secure and usable systems, before critically analysing the thesis in more detail. This critical analysis involves summarising the case made by the thesis, reviewing whether the research questions posed in Section 1.2 have been properly answered, reflecting on the issues identified while conducting this research, and stating how the research contributions in this dissertation answer the research questions. Finally, we conclude this chapter by proposing future work extending the contributions made.

1.5 Publications arising from thesis work

Figure 1.2 describes the elements of this dissertation which have been published in journals and peer-reviewed workshop and conference proceedings.

Publication	Related Chapter
Faily, S., and Fléchais, I. A Meta-Model for Usable Secure Requirements Engineering. In Proceedings of the 6th International Workshop on Software Engineering for Secure Systems (2010), IEEE Computer Society, pp. 126–135.	4
Faily, S., and Flechais, I. Analysing and Visualising Security and Usability in IRIS. In Proceedings of the 5th International Conference on Availability, Reliability and Security (2010), IEEE Computer Society, pp. 543–548.	6
Faily, S., and Fléchais, I. Towards tool-support for Usable Secure Requirements Engineering with CAIRIS. International Journal of Secure Software Engineering 1, 3 (July-September 2010), 56–70.	6
Faily, S., and Fléchais, I. Barry is not the weakest link: Eliciting Secure System Requirements with Personas. In Proceedings of the 24th British HCI Group Annual Conference on People and Computers: Play is a Serious Business (2010), BCS-HCI '10, British Computer Society, pp. 113–120.	7
Faily, S., and Fléchais, I. The secret lives of assumptions: Developing and refining assumption personas for secure system design. In Proceedings of the 3rd Conference on Human-Centered Software Engineering (2010), vol. LNCS 6409, Springer, pp. 111–118.	8
Faily, S., and Fléchais, I. Eliciting Usable Security Requirements with Misusability Cases. In Proceedings of the 19th IEEE International Requirements Engineering Conference (2011), IEEE Computer Society. To Appear.	8
Faily, S., and Fléchais, I. Persona cases: a technique for grounding personas. In Proceedings of the 29th international conference on Human factors in computing systems (2011), ACM, pp. 2267–2270.	9
Faily, S., and Fléchais, I. User-centered information security policy development in a post-stuxnet world. In Proceedings of the 6th International Conference on Availability, Reliability and Security (2011). To Appear.	9

Figure 1.2: Publications resulting from thesis

Chapter 2

Literature Review

In this chapter, we review the current state-of-the-art in the design of usable and secure systems. The chapter begins by identifying several common themes in the design of effective information security and reviews work by the HCI-Sec community towards designing usable security. Based on limitations in this existing work, we take a step back and review the prevalent HCI concepts available for designing usable and secure systems, including research on integrating these ideas with Software Engineering, and the potential consequences of these approaches to security.

Because the concept of *Requirement* is shared by both the security and usability communities, we review how existing work in Security Requirements Engineering might be cogent to the design of usability. In particular, we review several dominant Requirements Engineering approaches, and consider issues which may arise when viewing them from a usability perspective. We also introduce the concept of a *framework* and illustrate how existing Requirements Engineering frameworks deal with eliciting security and usability concerns. We conclude this chapter with a brief review of the available tool-support for facilitating Usability and Security Requirements Engineering activities.

2.1 Towards usable security design

2.1.1 Themes in Information Security design

Within the context of this thesis, the goal of security is to protect the organisation. Schumacher et al. [214] define security as the totality of all services and mechanisms protecting an enterprise. Information Security is defined by ISO/IEC 27002 [140] as *the protection of information from a wide range of threats in order to ensure business continuity, minimize business risk, and maximize return on investments and business opportunities.* Although this definition appeals to the notion of *business*, it is unfair to typecast businesses as purely commercial operations. We would be hard pressed to find examples of organisations not concerned about their risks, their on-going continuity, and making efficient use of their resources. Nevertheless what the definition does offer is the idea that information is the lifeblood of an organisation, and safeguarding it is tantamount to protecting the organisation.

Even though the literature reports many methods and tools for designing security, a number of themes are pervasive when designing effective information security.

First, security is about the protection of *assets* [112]; these need to be both identified and appraised, and are defined as anything that has value to an organisation [138]. Valuation of assets is particularly difficult as this needs to be expressed in terms of organisation value or, more directly, as the cost paid by the organisation should the asset be compromised. Gollman [112] claims the identification of assets is comparatively easier than its valuation, but it is unwise to trivialise the exercise of identifying *information* assets. Information can be defined as data interpreted in some meaningful context [33], and as knowledge communicated concerning some particular feat, subject, or event [156]. In both cases, the significance and value of information depends on the context where it is found.

Second, because protecting an organisation from all possible threats is untenable, security requirements are often derived by assessing their *risks* [140]. *Risk* is evocative of some form of threat, danger or hazard. A multidisciplinary study commissioned by the Royal Society [206] identified two forms of risk: objective risk and perceived risk. Objective risk defines risk as the probability of harm occurring; this harm can be stated scientifically and is capable of measurement. Perceived risk relates to the subjective assumptions people hold about future events. Given the interpretive nature of information, it is this latter definition which holds sway in the Information Security literature. Closely associated with the definition of *Risk* are the concepts of *Threat* and *Vulnerability.* ISO/IEC 27002 [140] defines a threat as a potential cause of an unwanted incident, and a vulnerability as a weakness of an asset or group of assets that can be exploited by one or more threats.

Third, as succinctly put by Bruce Schneier, security is a *process rather than a product*: processes need to be put in place to recognise inherent insecurity [213]. Understanding how to do this has been a consistent theme in the Information Systems Security literature. Of particular interest to security design has been work on the following:

- Responsibility modelling, e.g. [23, 35, 236]. This technique is concerned with defining what roles are held by responsibility owners and modelling responsibility relationships between agents.
- Writing and promoting Information Security Policies: these are documents that describe, in general terms, the security goals of an organisation [26].
- Fostering security awareness within an organisation [120, 67], and behaviour conducive to an organisation's security policy [244, 245].

Finally, and as a corollary of the previous points, security has a human dimension. Assets have value not only to members of an organisation, but also to the attackers behind organisational threats. Writing easily understandable security policies which balance protection with productivity is a challenge [26]. Recent work on security perceptions held by inter-organisational groupings [88] found that fostering a culture of security involves more than just policy communication; it involves factors such as understanding the values and norms of an organisation's culture and sub-cultures, and an appreciation of the different work contexts [88]. The human and cultural dimension of security has been explored from a design perspective. In particular, James' work on the Orion strategy [145] draws on analogies with Checkland's Soft Systems Methodology approach [47] which treats design as a form of participative enquiry in the real world of a system's problem, and the conceptual world of future systems. The high-level of stakeholder participation associated with this approach was found to raise awareness of security and increase subsequent ownership of selected security measures.

Unfortunately, the prevailing approach to tackling human factors in Information Security is to treat people as the problem. For example, Schneier claims that security is only as good as its weakest link, and that people are the weak link [212]. However, Brostoff and Sasse argue that this position is tantamount to blaming users for a security design they might compromise, and analogous to blaming the cause of safety-critical system failures on *human error* rather than bad design [38].

Therefore, for information to be secure, systems need to be usable.

2.1.2 Usable security

In their seminal paper on Information Security, Saltzer and Schroeder [209] espouse several design principles for protection mechanisms. One of these is the principle of Psychological Acceptability, which states: *The interface to the protection mechanism should be designed for ease of use, so users routinely and automatically apply the mechanism correctly.* In recent years, there has been an enormous amount of research in the area of HCI-Sec; seminal work in this area has looked at the usability of password policies [7] and security controls [260]. The HCI-Sec community has considered the challenge of designing usable security from two different standpoints.

2.1.2.1 Design principles and idioms

The first standpoint is characterised by the application of usable security design principles and idioms. Examples of these include Yee's design guidelines and strategies for secure interaction [264]. These guidelines are written as principles a designer should consider when making decisions about secure interface design; examples of these guidelines include: *Match the most comfortable way to do tasks with the least granting of authority*, and *Indicate clearly the consequences of decisions that the user is expected to make.*

Another example of this standpoint is Garfinkel's patterns for usable security [107]. These patterns are written in the same style as the design patterns in Gamma et al.'s seminal work on Design Patterns [105], and catalogued according to usage. Examples of these patterns include:

- User visibility and sanitization: Explicit user audit, Reset to initialisation, Delay unrecoverable action.
- Identification and key management: Leverage existing identification, Create keys when needed, Distinguish internal senders.
- Patterns for promoting overall secure operation: Distinguish security levels, Disable by default, Install before execute.

Yee and Garfinkel's contributions are both sensible and well meaning, but are not a panacea for designing usable security. For example, one of Yee's Communication principles states that a designer should "Present objects and actions using distinguishable, truthful appearances," yet nothing is said about how this might be done. In the case of Garfinkel's patterns, while the relationship between many of the patterns are described, unlike [105], the consequences (particularly to security) of applying these patterns is not. For example, the *Leverage Existing Identification* pattern describes participants, the organisational implementation, and the successful results of this pattern, but culling the consequences of this pattern from known uses is left as an exercise to the designer; therefore, vulnerabilities may be introduced if certain details of the pattern are misinterpreted.

Despite its faults, the key contribution made by both Yee and Garfinkel is the notion that security and usability *can* be harmonious. Yee's guidelines were designed to appeal to the mental models of users, while Garfinkel's patterns are the results of synthesising known examples of usable security design.

2.1.2.2 User-centered security

The second standpoint is characterised by Sasse et al's work on applying HCI design approaches to the design of security mechanisms [210], and Zurko and Simon's work on user-centered security [271].

Based on the empirical findings from several studies about password usage, Sasse et al. found that framing the design of security controls from a technology, user, goal, task, and context perspective can lead to useful insights. For example, considering security as an enabling task that needs to be completed before a main task helps explain why users choose to circumvent controls which get in the ways of their work.

User-centered security refers to "security models, mechanisms, systems, and software that have usability as a primary motivation or goal". Zurko and Simon claim that secure systems have, traditionally, been indifferent to the needs of users — irrespective of whether these are end-users, developers or administrators. Like Yee [264], Zurko and Simon argue for security models conducive to the mental models

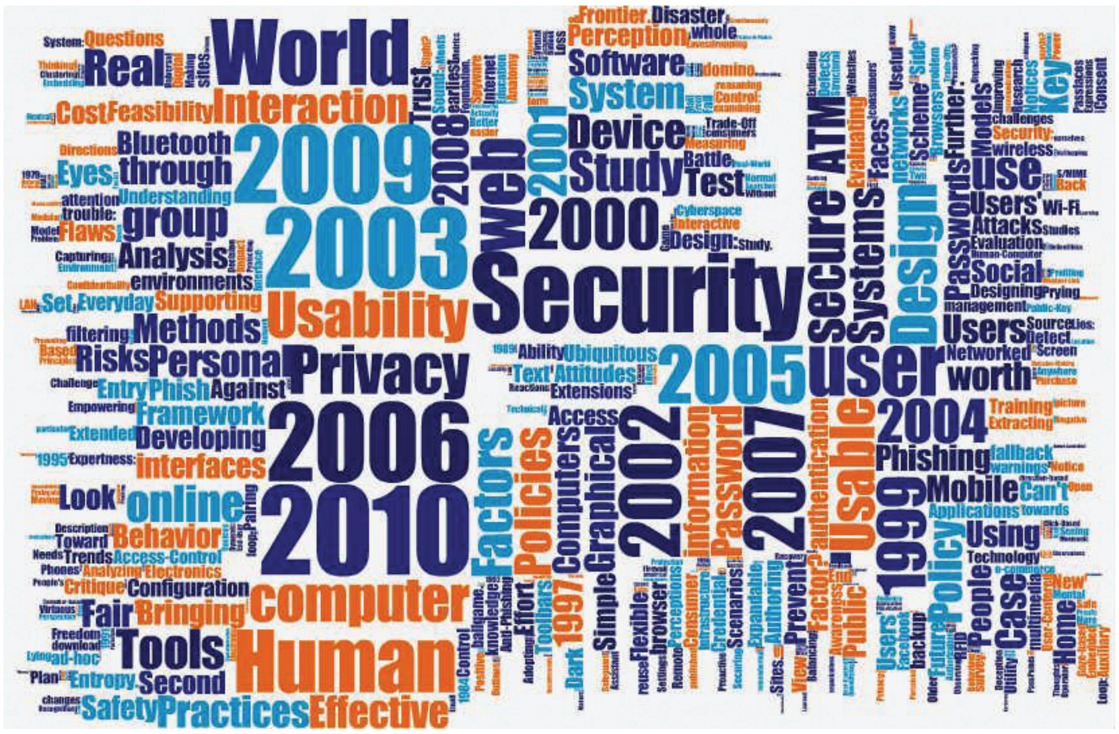


Figure 2.1: Word cloud of HCI-Sec paper topics (taken from [240])

of different types of users, but they also propose synthesising security design techniques with established design techniques from HCI.

Sasse et al.'s and Zurko and Simon's work inspired, and continues to inspire, much of the current research in HCI-Sec. Surprisingly, as the word cloud in Figure 2.1 illustrates, the bulk of work in this community focuses on studying the usability of security controls rather than activities associated with designing systems that are usable and secure. In particular, a recent survey by Birge [34] found that many HCI-Sec papers are divided into studies about usability testing on security controls, mitigating security controls, conceptual investigations about terms such as "trust" and "privacy", experience studies about user attitudes, and models and guidelines demonstrating trusted interactions and trusted user interfaces. Birge also notes that much of this research focuses on the needs of the end-user rather than the needs of the designer, and few studies have attempted to tackle the question about how designers should approach security concerns. Moreover, attempts to synthesise HCI and security techniques continue to raise issues for researchers. To understand why, we examine two particular examples of such follow-on work.

The first example is Gerd tom Markotten's work on User-Centered Software Engineering [246]. Like the User-Centered Security paradigm, this work attempts to synthesise a Security Engineering process with Usability Engineering. The Security Engineering process itself involves defining the characteristics of a candidate architecture, performing threat and risk analysis on this architecture, deriving more detailed architectural components, connectors, and implementation logic, and evaluating the system. This synthesis involves informing the characteristics of the candidate architecture with insights from interviews and observations of target user groups, considering the role of users as inside attackers during threat and risk analysis, iteratively developing the system's user interfaces, and evaluating them using usability inspection methods [178]. This approach was used to support the development of a software tool allowing security novices to align their security goals with the expectations of certain internet-facing applications [146]. Although initially promising, this work remains largely unvalidated; no information about the target user group is provided, nor are any insights into the practical aspects of synthesising usability methods with the methods used to design the security tool.

The second example is the AEGIS (Appropriate and Effective Guidance for Information Security) design method by Fléchais et al. [98]. As a participative design method, AEGIS builds on the ideas of approaches such as the Orion strategy introduced in Section 2.1; the method assumes that secure systems are not merely software systems, but socio-technical systems: systems of technology used within a system of activity. To this end, AEGIS was designed as a lightweight process which augments existing software development methods that provides guidance to developers for designing secure systems. This process is applied within a focus group setting, where stakeholders gather, identify, and model the system's assets in different contexts. These assets are evaluated according to values held by the participants about them. Vulnerabilities, threats, and risks affecting these assets are elicited, before possible security controls mitigating these risks are selected. The costs and benefits of situating these countermeasures for each of the affected contexts is considered and, if unmitigated risks remain, this process is repeated.

In a comparatively recent retrospective of work in this area [270], Zurko highlighted a number of challenges for designers. These include reducing usable security techniques to methods simple enough for most developers to execute effectively, and ensuring that users understand how to use security controls directly related to their tasks and contexts. AEGIS deals quite succinctly with both of these challenges. First, AEGIS assets and their relationships are modelled using UML [208], arguably the *lingua franca* of modelling notations in the Software Engineering community. By modelling the on-going analysis as UML, AEGIS asset models make useful boundary objects: artefacts flexible enough to be used within a conceptual boundary, but robust enough to retain a common identity across boundaries [232]. Second, asset modelling and risk analysis is carried out with respect to different *environments*. These represent physical or cultural contexts meaningful to workshop participants. Therefore, rather than modelling and analysing an arbitrary system, participants can relate to the different contexts the analysis is related to.

At a superficial level, AEGIS appeals to Zurko and Simon's canons for User-Centred Security. A more critical analysis of AEGIS does, however, raise issues; these affect not only AEGIS but the paradigm of User-Centered Security in general.

First, both User-Centered Security Engineering and AEGIS assume that usability will follow by taking a participative approach to design. However, work by Irestig et al. [135] found that while participative practices are effective for eliciting usage culture values that might have otherwise remained tacit, resulting systems tend to be comparatively chaotic, small-scale, and imbued with the power relationships of the organisation.

Second, although conceptually simple, UML class diagrams alone cannot model the many elements which also impact secure systems, such as goals, tasks carried out by users, and other potentially relevant relationships, such as dependencies between different users. Moreover, as analysis progresses, models are likely to grow and become unwieldy without dedicated tool-support; to date, no such tool-support has been presented.

Finally, although treating environments as first-class modelling objects is an important step towards contextualising risk analysis, workshops alone may not be enough to elicit information about threats and vulnerabilities within different environments. Although Zurko and Simon have proposed applying techniques such as Contextual Inquiry [125] to elicit such data, we are unaware of any peer-reviewed literature which has attempted to do this.

2.1.3 A case for better HCI integration?

This brief review of the Information Security and HCI-Sec literature has reinforced the necessity to consider the role of people and their contexts when designing usable security. Although the design artifacts of usable security design are subjective, there is evidence that usability and security do not have to be treated as conflicting qualities. Unfortunately, the evidence in Section 2.1.2.2 suggests that while synthesising techniques from security and HCI is desirable, design issues may arise if applied to cases more significant than those reported by the literature to date.

Based on the strength of these findings, we propose that rather than adopting an ad-hoc approach to synthesising security and usability design techniques, we should carry out a more thorough review of the HCI and related Software and Security Engineering literature to develop a better understanding of the underlying issues.

2.2 Usability design

2.2.1 HCI and usability

Human-Computer Interaction (HCI) is "a discipline concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" [122]. HCI is an interdisciplinary field and, throughout the last several decades, has been influenced by research in fields such as ergonomics, cognitive science, computer science, social sciences, and the arts and humanities. These influences have shaped how the discipline has approached the idea of design. In paradigmic terms, Cockton notes that the discipline has been rebranded from the *system-centered* seventies, through to the *user-centered* eighties, the *context-centered* nineties, and into the opening *value-centered* decade of the new millennium [53].

Usability has been described as "the professional term for HCI" [241], although its definition varies depending on the stake that practitioners have in it. Coutaz and Calvary [65] state that Software Engineers consider Usability as an intrinsic property of a system, in that a system can be intrinsically usable or unusable. For example, Ghezzi et al. [109] consider a software system usable if its human users find it easy to use. Similarly, Lauesen [153] defines Usability as the combination of the following six factors.

- Fit for use: The system can support the user's real life tasks.
- Ease of learning: How easy is the system to learn for various user groups?
- Task efficiency: How efficient is the system for the frequent user?
- Ease of remembering: How easy is the system to remember for the occasional user?
- Subjective satisfaction: How satisfied is the user with the system?
- Understandability: How easy is it to understand what the system does?

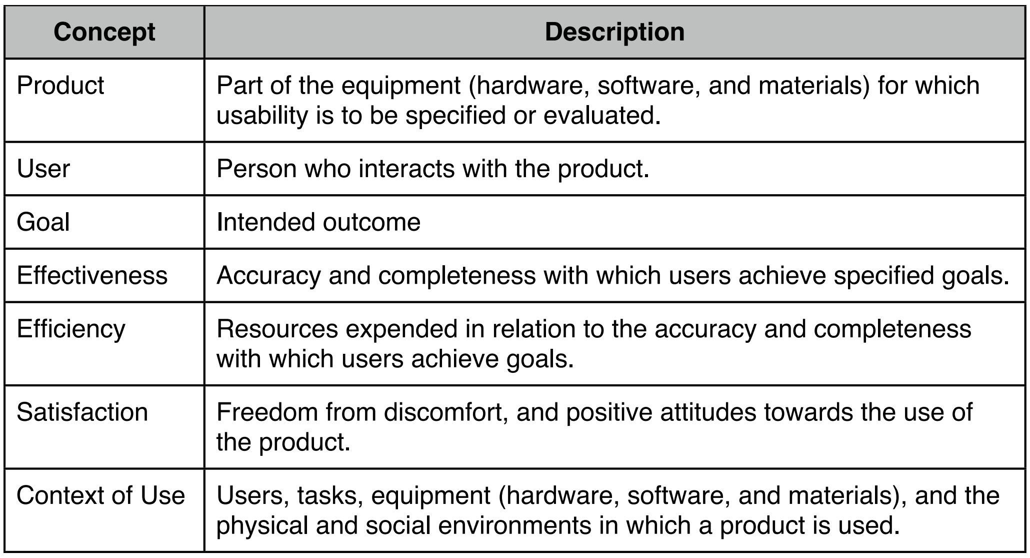
Conversely, the HCI community considers Usability not as an intrinsic system property, but as a contextual concept. In this sense, Usability is relative to a system's contexts of use, rather than a system being intrinsically usable or unusable [65]. ISO 9241-11 encapsulates this contextual nature in its definition of Usability, i.e. the extent to which a *product* can be used by specified *user*s to achieve specified *goal*s with *effectiveness*, *efficiency* and *satisfaction* in a specified *context of use* [136]. Each of the italicised terms are further specified by the ISO standard as shown in Figure 2.2.

Figure 2.3 describes the relationship between these different components of Usability. It represents what the standard calls a *Work System*: a system consisting of users, equipment, tasks and a physical and social environment, for the purpose of achieving particular goals [136]. In effect, a user is carrying out tasks, which are collections of activities, to achieve one or more goals.

Given our interest in exploring HCI, rather than Software Engineering, approaches for usability design, this dissertation adopts the ISO 9241-11 definition of Usability as the basis for further review of the literature. In the following sections, we review design approaches appealing to this definition, together with problems that arise when considering the synthesis of design, security, and usability.

2.2.2 User-centered approaches

The design philosophy espoused by most usability professionals is *User-Centered Design*. The introduction of this term into the design vernacular is attributable to Donald Norman, who defined it as *a*



Concept	Description
Product	Part of the equipment (hardware, software, and materials) for which usability is to be specified or evaluated.
User	Person who interacts with the product.
Goal	Intended outcome
Effectiveness	Accuracy and completeness with which users achieve specified goals.
Efficiency	Resources expended in relation to the accuracy and completeness with which users achieve goals.
Satisfaction	Freedom from discomfort, and positive attitudes towards the use of the product.
Context of Use	Users, tasks, equipment (hardware, software, and materials), and the physical and social environments in which a product is used.

Figure 2.2: ISO 9241 usability definitions

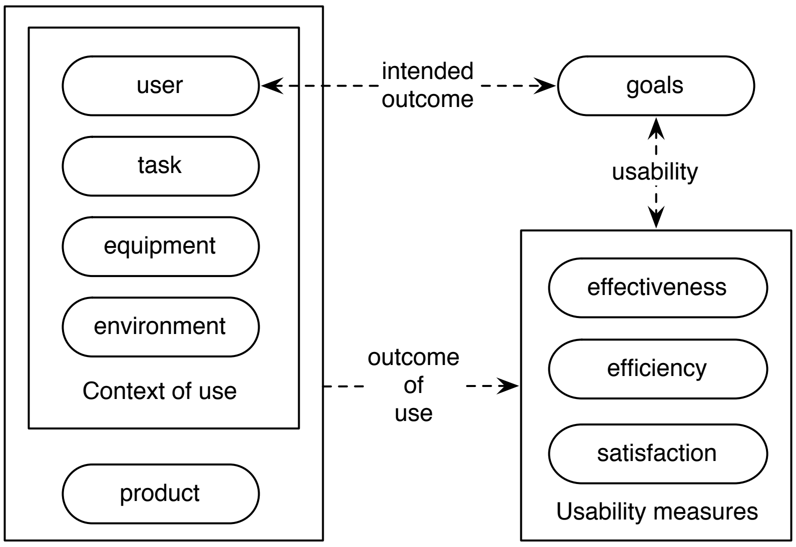


Figure 2.3: ISO 9241-11 usability framework

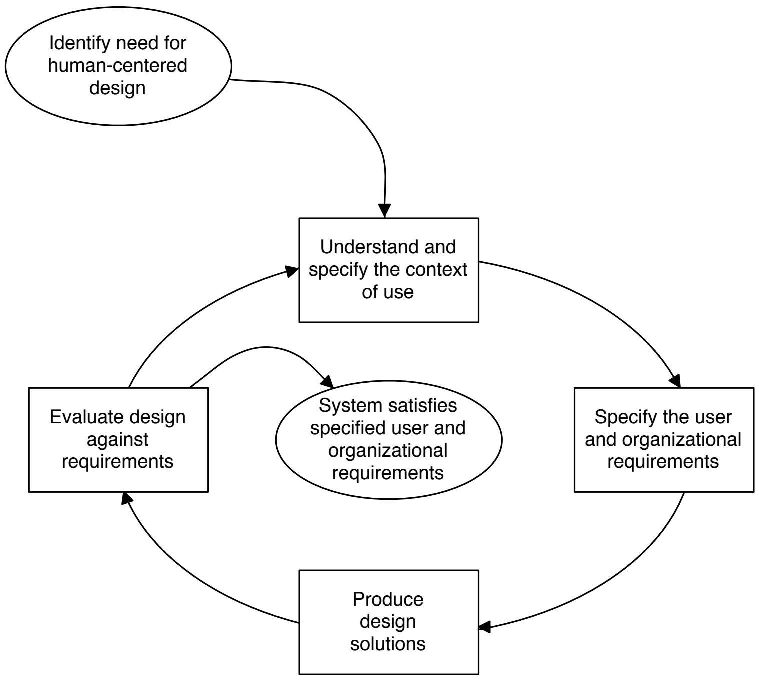

Identify need for human-centered design
Understand and specify the context of use
Specify the user and organizational requirements
Produce design solutions
Evaluate design against requirements
System satisfies specified user and organizational requirements

Figure 2.4: Human-centered design activities

philosophy based on the needs and interests of the user, with an emphasis on making products usable and understandable. The origins of user-centricity can be traced back to the *guided fantasy* technique that Tim Mott and Larry Tesler used at Xerox PARC to develop the first generation of desktop publishing systems [170]. User-Centered Design is now the prevailing paradigm for designing interactions with technology, and not just software systems. This said, when arguing the need for *user-centered* approaches, the work most commonly referred to in the HCI literature is Gould and Lewis' key principles for useful and easy-to-use computer systems [113]. Based on their experiences at IBM, Gould and Lewis recommend the following three principles of design.

- Early focus on users and tasks: designers need to study the behavioural characteristic of users, together with the work they need to accomplish.
- Empirical measurement: the performance and reactions of users should be observed, recorded, and analysed while using prototypes to carry out real work.
- Iterative design: a cycle of design, test, measure, and redesign should be carried out and repeated as necessary.

These principles are similar to the principles of Human-Centered Design stipulated by ISO/IEC 13407 [137], i.e.,

- the active involvement of users and a clear understanding of user and task requirements;
- an appropriate allocation of function between users and technology;
- the iteration of design solutions; and
- multi-disciplinary design.

Several usability design processes appeal to this notion of user-centricity; these include Goal-Directed Design [61], Contextual Design [32], and Usage-Centered Design [57]. These processes are not incompatible with the ISO 13407 Human-Centered Design activities, illustrated in Figure 2.4. Indeed, a number of concepts and techniques are prevalent in these processes. These are described in more detail in the following sections.

2.2.2.1 Goals

A precise definition of what the User-Centered Design community means by a *goal* eludes us. Cooper et al. [61] define a goal as *an expectation of an end condition*; Holtzblatt et al. [126] consider a goal to be an achievement to be worked towards or maintained. In both cases, goals are both personal and motivational.

Cooper et al. [61] claim that a user goal might be an experience goal (expressing how they want to feel), an end goal (expressing the motivation for performing a task), or a life goal (expressing personal aspirations). Other goals, such as customer or business goals also exist, but the authors state that these goals should not be expressed at the expense of the user.

2.2.2.2 Tasks and scenarios

Users carry out tasks: activities required to achieve a goal [136]. To understand the usability impact of tasks, some form of *task analysis* is performed. Task analysis is concerned with the performance of work [74], where *performance* is a synonym for the activities needed to achieve a goal. The outcome of task analysis is one or more models which describe how work is carried out. Examples of task analysis techniques include Hierarchical Task Analysis (HTA) [14], GOMS (Goals, Operations, Methods, and Selection rules) [150], and Scenario-Based Task Analysis [110]. However, of these different approaches, scenarios are, arguably, the most widely used.

Scenarios are stories about people carrying out an activity [203]. The length of this story, the nature of the people, and the types of supplemental information vary based on the developmental context within which scenarios are used. For example, in addition to a narrative describing a user's actions, the User Interaction Scenarios described in [203] include other elements such as scenario-setting, participating actors, goals motivating the actors, and supplemental information about external events. Although such scenarios act as vignettes describing how people might use the designed technology to carry out some activities, these can also be used to provoke discussion around the darker aspects of technology use. An example of such an approach is provided by [176], where scenarios describe functionally valid but dystopian aspects of system design decisions during the early stages of a project, or during policy-making discussions.

A key advantage of scenarios is their allusory nature; they can be used to impart knowledge about tasks at any level of abstraction, provided it can be rendered in a narrative structure. Consequently, scenarios are a universal language in User-Centered Design, and can be used at all stages in usability design, from eliciting data about user tasks, through to storyboarding possible design implementations. Moreover, as Section 2.3.3 will discuss, scenarios are a candidate boundary object that all members of a project team can relate to.

Scenarios are episodic, and symbiotically tied to their contexts of use. As a consequence, they tend to be fragmentary in nature. They are also less structured as a task modelling notation compared to other task analysis techniques, such as Hierarchical Task Analysis [14].

2.2.2.3 Personas and assumption personas

Personas are behavioural specifications of archetypical users. These were first introduced by Cooper [60] as a means of dealing with programmer biases arising from the word *user*. These biases lead to programmers introducing assumptions causing users to bend and stretch to meet these needs; Cooper called this phenomenon *designing for the elastic user*. Cooper's solution was to design for a single user representing the target segment of the system or product being designed. This approach brings two benefits. First, designers only have to focus on those requirements necessary to keep the target persona happy. Second, the idiosyncratic detail associated with personas makes them communicative to a variety of stakeholders. Since their initial proposal over a decade ago, personas have now become a mainstay in User-Centered Design, with articles, book-chapters, and even a book [196] devoted to the subject of developing and applying them to support usability design.

Personas were designed to be data-driven artifacts; they are grounded in empirical data collected about representative users who carry out work within their normal contexts of use. There are cases, however, when personas need to be developed to articulate viewpoints or explore ideas. *Assumption Personas* are persona sketches created to articulate existing assumptions about an organisation's user population [196]. Rather than being based on observed data of prospective users, these are grounded in assumptions that contributors hold about users and the context of investigation. These assumptions may be derived from interpreted or mis-interpreted experiences, and coloured by individual and organisational values. Assumption personas help people to see the value of personas in design, and how different assumptions can shape these. As a result, they guide subsequent analysis or data collection for data-driven personas. Although there are subtle differences in their definition, for practical purposes, assumption personas are synonymous with provisional personas [61] and ad-hoc personas [182].

2.2.2.4 User-centered criticisms

User-centered approaches are popular, but are not without their flaws. From a security design perspective, it is important to understand these weaknesses, as these may lead to the unintentional introduction of vulnerabilities into design.

First, user-centered approaches assume that usability design should precede general system design. Cockton [54] argues that invention will precede innovation, and while concurrent design may be possible, an iterative design process beginning with human factors work is an unrealistic aspiration.

Second, user-centered approaches may be implicitly biased against the software developer community. The premise behind Cooper's argument for personas in [60] is that developers will, given the opportunity, design any given software product for themselves, and that many examples of bad usability can be attributed to software developer indifference to usability. In particular, Cooper's argument is founded on the failure of a particular Microsoft project described in [172], where a culture of usability allegedly clashed with a culture of engineering. However, an alternative reading of [172] also suggests that poor Requirements Engineering practices on the part of the usability designers and a failure to consistently respond to developer requests for a working requirements specification may have been as much to blame. Consequently, it is equally possible that [172] strengthens, rather than rebuts, Cockton's more recent observations. A similar anti-technologist bias has been observed by Thimbleby [242], who claims that many usability professionals believe that technologists are the origin of usability problems, and that these problems can be solved by extolling the virtues of user-centricity and acting as user proxies.

Third, user-centered approaches may, in some cases, be methodologically weak. Chapman and Milham [46] report little peer-reviewed discussion of the Personas technique and, as a result, it may be impossible to verify their accuracy. Moreover, because personas are fictional representations, there is no easy way to falsify them. Consequently, if a persona is developed using questionable methods, or less than accurate empirical data, it is difficult to disprove a persona's validity. Further concerns and practical limitations are described in [46].

Fourth, focusing primarily on user goals and the tasks necessary to achieve them may be making sweeping generalisations about the subtleties of interaction design. In his argument for activity, rather than user-centered design, Norman [181] claims that many examples of successful technology are designed around activities rather than users that carry them out. In this sense, we are required to adapt to technology rather than the other way around. Norman argues that user-centered approaches may be harmful because focusing on users detracts from the importance of the activity. Moreover, acquiescing to user requests may lead to overly complex designs, whereas a well-articulated design model is likely to follow if design instead focuses on activities.

A further danger on focusing on the user is that it potentially blinds the designer to the social context within which the user is situated. Bannon [24] noted nearly two decades ago that HCI research has tended to focus on the needs of individual users, and has neglected the importance of co-ordination and co-operation between processes. Since then, the Computer Supported Cooperative Work (CSCW) community has noted that the social activities technology needs to support are nuanced. Moreover, the shared understanding of systems assumed by designers do not hold, especially when users have hidden or conflicting goals that they resist articulating [5]. While attempts have been made to make design techniques available for eliciting such insights, e.g. [32], Randall et al. claim that tensions exist when adapting them because engineers are concerned with synthesis rather than analysis, and their activities need to yield information comparatively quickly [197]. Recent work has started to adapt user-centered design artifacts for collaborative activities, e.g. [160], but the resulting impact of scaling individualist design artifacts to support collaborative activities remains unknown.

A final point made by Thimbleby [243] is that knowing about usability problems is different from being willing and able to fix them. Knowing that there are usability problems is, however, important, but while user-centered approaches are necessary and useful, they are not in themselves sufficient. This is because user-centered design techniques appear to focus solely on the world external to the user interface without considering the usefulness of formal specifications for modelling complexity and ambiguity; such specifications can, potentially, highlight the causes of poor user interface design. For this reason, certain HCI proponents propose synthesising Software Engineering practices with HCI.

2.2.3 Human-centered software engineering

The need to develop engineering principles for human factors was proposed over two decades ago by Dowell and Long [76]. Dowell and Long argue that because usability design is poorly integrated into Software Engineering processes, developers make decisions which impact the usability of a system without the involvement of usability specialists. They consider work as a work-system activity which changes the *application domain*: the part of the world relevant to the functioning of the work-system. As such, tasks are the means by which the work system changes the application domain, and goals are desired future states the application domain work system should achieve.

Making the distinction between user goals and system goals (or product goals as [76] calls them) is analogous to drawing a line between HCI and Software Engineering. There has been work in the intersection of these disciplines suggesting that each has strengths and weaknesses. As such, an integrated approach incorporating design, development, and evaluation principles from both fields may lead to more effective use of usability design practices within Software Engineering [218]. Research in this area has been the subject of a number of recent conferences, such as the IFIP Human-Centered Software Engineering conference series, books [217, 220], and articles in mainstream journals [219].

One way of tackling the problem of integrating HCI with Software Engineering is to adopt common techniques and modelling notations. For example, Usage-Centered Design [57, 56] models the interaction of user roles (abstractions of interaction design relationships) and the system as structured scenarios called *essential use cases*. Usage-Centered Design also incorporates several different modelling notations related to UML, e.g. [81]. There is, however, a danger that focusing on the essentials of the problem and abstracting away details about users will also lead to details about their contexts being abstracted away as well. Moreover, Gulliksen et al. [114] argue that adopting such a model-driven approach might disenfranchise the users, relegating them from the role of co-designer to informant. Similarly, Thimbleby's emphasis on the difference between the professional skills associated with User-Centered Design and other engineering approaches, and a perceived research and industry dichotomy [242] risks polarising interaction designers and engineers in much the same way as Cooper's argument for personas in Section 2.2.2.4.

From a security perspective, both the context and stakeholders may be important sources of data about possible threats and vulnerabilities, as well as usability and functionality. Consequently, scenarios and modelling notations alone may not be enough to synthesise usability and security for the purposes of design.

Sutcliffe [237] has compared and contrasted the traditions in both HCI and Software Engineering with regards to design and its theoretical underpinnings. Sutcliffe concluded that scenarios are indeed boundary objects between these disciplines, but that both occasionally carry out research within each other's bailiwick. In one sense, this might be seen as competition, but it could also be viewed as convergence between disciplines. Both disciplines do appear to converge when dealing with interaction concerns, although the viewpoints of each discipline differ in both cases. Sutcliffe argues that synthesis may be possible if instead of focusing on heavyweight methods, a spectrum of complementary approaches is adopted. Sutcliffe identifies scenarios as an integrating technique, however *requirements* may be at least as powerful an integrating technique, especially when considering the synthesis of the additional elements of security.

2.3 Specifying Security

Zave defines Requirements Engineering as the branch of Software Engineering concerned with real-world goals for functions of, and constraints on, software systems [268]. These *real-world goals* motivate *why* a system needs to be built, as well as *what* the system will be [184]. In particular, a *Requirement* is defined by the IEEE [131] as one of:

1. a condition or capability needed by a user to solve a problem or achieve an objective;

2. a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; or

3. a documented representation of a condition or capability as in (1) or (2).

Requirements are typically classified according to whether they are *Functional* or *Non-Functional*. There is no accepted definition for each of these classifications, although it is generally accepted that functional requirements define *what* aspects of what the condition or capability will do, while non-functional requirements define the *how well* aspects; for this reason, non-functional requirements are often referred to as *quality requirements*.

Texts such as [228] and [13] characterise best practice in Requirements Engineering. In particular, the Volere Requirements Specification template [200] has been widely adopted in both industry and Requirements Engineering programmes taught on university undergraduate and graduate courses. Augmenting this best practice is tool-support, e.g. [129], and guidelines for tool-developers for developing tool-support for Requirements Engineering [123]. Given industry's interest in Requirements Engineering, and a growing acknowledgement of the need to address non-functional concerns at the early stages of design, the dearth of case studies reporting the results of contemporary Requirements Engineering techniques with regards to security and usability is surprising.

Security requirements are often considered as non-functional requirements, although there has been debate about whether this might be harmful. Fléchais et al. [99] argue that treating security as a Non-Functional Requirement may encourage designers to defer security requirements engineering and design until a design is mature because functionality should take precedence over other concerns. Conversely, however, Cockton's argument about functionality taking natural precedence over usability in Section 2.2.2.4 appears to be no less valid if we consider both security and usability as quality concerns. Unfortunately, the definition of what a requirement is from a security perspective is equally unclear. The remit of *security* requirements appears to be wide, and includes factoring education and training [119], conforming to external audit and governmental requirements [120], and incorporating the need for employees to believe in the security of their organisation [207].

Given this remit, it appears that the distinction between goals, policies, and requirements is blurred. This lack of concrete guidance about what security requirements are is problematic for Software Engineering. A survey of Security Requirements Engineering approaches undertaken by Tøndel et al. [247] concluded that approaches for security requirements elicitation are not prescriptive enough to be usable by software developers. The survey results concur with previous observations by Firesmith [96] and Haley et al. [117] that engineers confuse security requirements with other developmental artefacts. It also argues that no clear definition for security requirements exists, due to disagreement on the extent to which security measures should be stated.

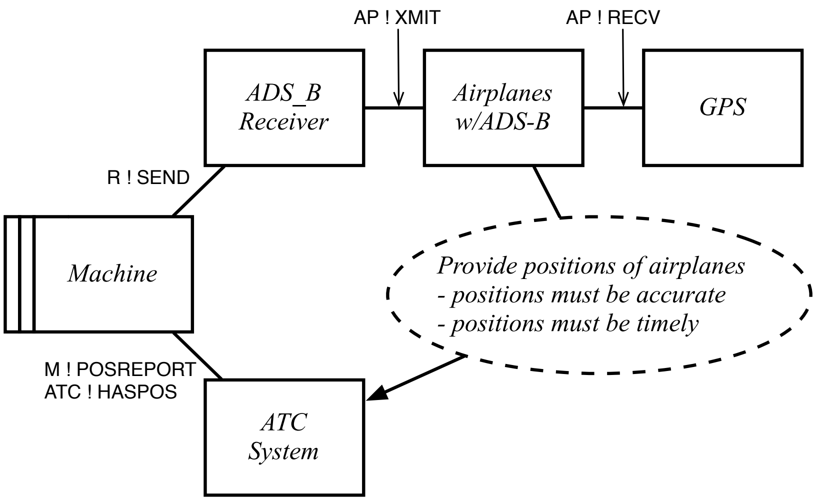


Figure 2.5: Example of a security requirement (modelled as an ellipse) constraining a context, modelled as a problem frame diagram

What is clear, however, is that Requirements Engineering as a discipline appears to be a research nexus between HCI and Information Security. Moreover, as Sutcliffe [237] alludes to, requirements are also a boundary object between HCI and Software Engineering research.

In the following sections, we review three particular Requirements Engineering approaches; these were selected because (a) they consider the elicitation and specification of requirements, and (b) these techniques have published security extensions. The approaches to be reviewed are Problem Frames, Goal-Oriented approaches, and Use Cases. We will briefly review characteristic techniques of these approaches, together with their security extensions before discussing the techniques applicability and consequences from a usability perspective.

2.3.1 Problem frames

Problem Frames are a tool for classifying, analysing, and structuring software development problems [143]. Problem Frame diagrams model tasks which need to be accomplished by a *machine* being specified within the real world. An example of a Problem Frame diagram, taken from [117], is illustrated in Figure 2.5. The diagram models the problem associated with a machine for passive surveillance of aircraft.

These diagrams model phenomena: observable entities, such as events, objects, and values. These can be grouped into *domains* represented as boxes. These domains may be *causal* and contain predictable phenomena, *biddable* if they contain unpredictable elements like people, or *lexical* if they contain representations of data. The boxes in Figure 2.5 represent different domains, but no supplemental information is provided about what domains these are. Consequently, the domain types are not important for investigating the problem. For example, at the current stage of our analysis, the Air Traffic Control (ATC) System could be an entire socio-technical system, or simply a collection of causal events.

Between each domain, there may be *shared phenomena*; these are represented by lines between the domains. These lines may also be labelled to give an indication about what sort of phenomena is being shared. Between the machine and the ATC System, two sets of phenomena are shared. The first represents position reports from the machine to the ATC System (M ! POSREPORT); the ! symbol represents phenomena output from a domain. The second represents a variable transmitted from the ATC System to indicate whether the ATC System has a fix on an aircraft. A requirement is represented using a dashed ellipse, which references one or more domains. If the reference association contains an arrow then the corresponding domain is constrained by a requirement, rather than simply referenced by it. In this example, the requirement constrains the ATC System.

2.3.1.1 Problem frames security extensions

Haley et al. [117] have proposed a Security Requirements Engineering approach which builds upon Problem Frames. This approach involves modelling the system's context using problem diagrams and eliciting system functional requirements, eliciting security goals and threats which violate them, and identifying security requirements. These requirements are modelled as constraints on the functional requirements. The system is then verified by constructing logical *outer arguments* to verify claims about system behaviour, and demonstrate that security requirements are satisfied in a system conforming to this specification. These arguments are augmented by informal *inner arguments*. Finally, to justify these arguments, it may be necessary to add detail to the system context. This, in turn, leads to these activities being repeated for this revised context.

The main contribution of Haley et al.'s framework is the recognition of the symbiotic nature of security requirements with other *core artifacts* of design, such as models and other requirements. In the case study described in [117], they describe how requirements were situated and, after analysing the requirements in context, rebuttals to security arguments could be identified. These rebuttals lead to further rounds of requirements analysis. As such, both the requirements and an understanding of the environment evolved as analysis progressed.

Another example of a specification process using Problem Frames is SEPP (Security Engineering Process using Patterns) [118]; this bears similarities to Haley et al.'s framework, but rather than refining requirements from goals, security goals are represented as individual machines; these become the focus of a problem diagram. An additional problem frame diagram, a Concretized Security Problem Frame (CSPF), is used to model the security mechanisms safeguarding the actual machine being specified. With the CSPF, a number of *necessary conditions* are specified; these represent assumptions needing to be met by the environment, and are analogous to the trust assumptions used in [117].

Recent work by Schmidt [211] aligns SEPP with risk analysis activities. Specifically, threats and attackers are treated as a biddable domain with a shared phenomena with an exposed domain. If threats are likely to exploit any necessary conditions in the CSPF, and the likelihood of this threat is considered to be sufficiently high, then this threat is mitigated via an auxiliary mechanism or via some form of organisational control.

2.3.1.2 Problem frame criticisms

From a Requirements Engineering perspective, Problem Frame approaches are elegant. These approaches do, however, incorporate a number of weaknesses when viewed through a usability lens.

First, assuming that security requirements are always preventative restricts their expressive power. In practice, a security goal can also be satisfied by deterring, detecting, or reacting to a particular threat; requirements for such threats cannot be easily modelled as constraints. Moreover, treating security as a constraint can put security requirements on a confrontational footing with usability; rather than considering security as an enabler, these approaches consider security as a constrainer.

Second, Problem Frames assume the existence of a single physical environment. The notion of context in problem frames is specific to the view of the problem world currently being modelled. However, this approach is insufficient for modelling problems in social contexts. For example, in [117] it is argued that if sensitive material is used on an executive's workstation then a confidentiality goal is applicable to that workstation asset. However, for a different social context, the value of that goal might vary, or an integrity or availability goal may be more relevant.

Third, Problem Frame approaches are agnostic of the techniques used to elicit data. Although [117] proposes using security argumentation to justify the assumptions underpinning security decisions, knowledge about vulnerabilities within the context may be lost in the process of preparing a succinct problem diagram.

Finally, Problem Frames have not been well-received outside of the Requirements Engineering research community. A study comparing the comparative usage of UML and Problem Frames [258] concluded that non-IT professionals find Problem Frames more difficult to grasp. Moreover, IT professionals felt uncomfortable presenting Problem Frame diagrams to non-IT stakeholders.

A valid counterargument to the final point is that *most* of the approaches in the section have yet to find usage outside of the research community. Problem Frames, however, suffer from a particular irony in that its biggest strength may also be its biggest weakness. The Problem Frames approach allows analysts to model complex domains using succinct, elegant models. These models do, however, compress a large amount of domain knowledge which makes interpreting a model challenging without a working knowledge of both the approach and the problem domain. In comparison, UML class diagrams are, in their most abstract sense, a model of classifiers (things) and associations between them. Consequently,

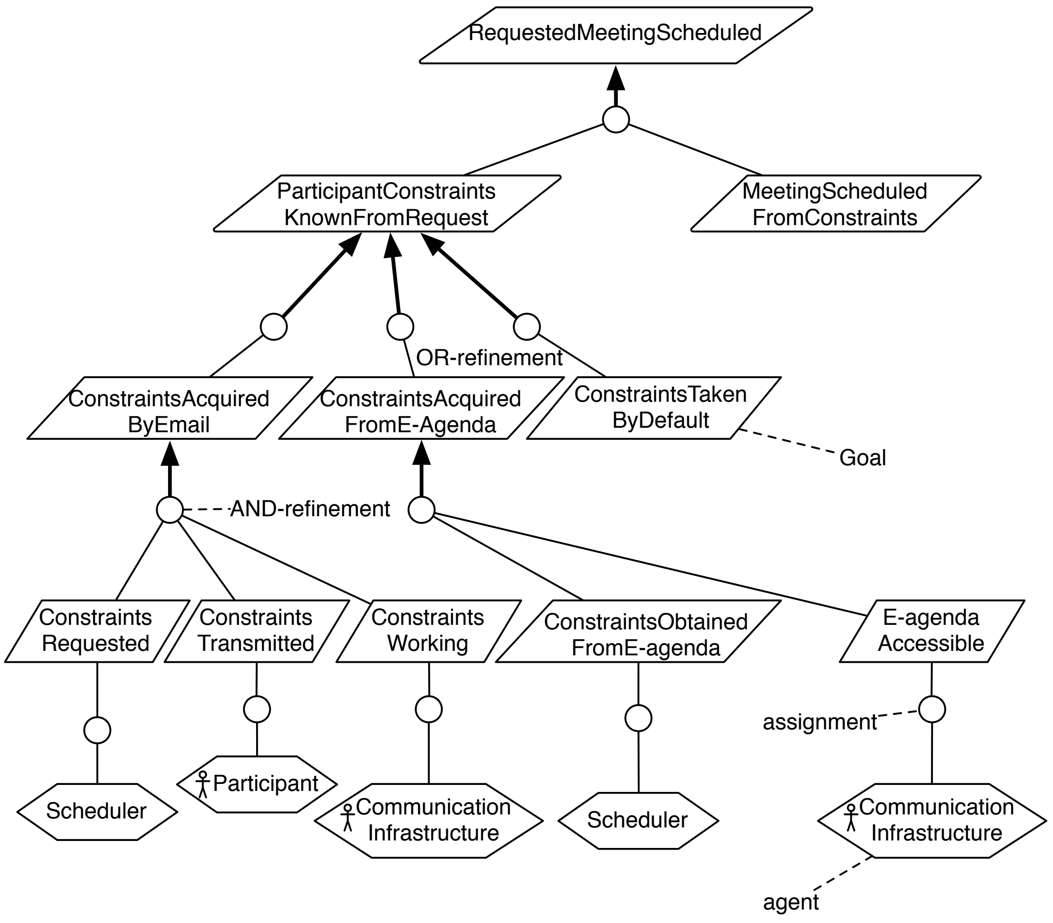

RequestedMeetingScheduled
ParticipantConstraints KnownFromRequest
MeetingScheduled FromConstraints
OR-refinement
ConstraintsAcquired ByEmail
ConstraintsAcquired FromE-Agenda
ConstraintsTaken ByDefault
Goal
AND-refinement
Constraints Requested
Constraints Transmitted
Constraints Working
ConstraintsObtained FromE-agenda
E-agenda Accessible
assignment
Scheduler
Participant
Communication Infrastructure
Scheduler
Communication Infrastructure
agent

Figure 2.6: KAOS goal model example

reading a UML class diagram requires significantly less cognitive effort than a Problem Frame diagram.

2.3.2 Goal-oriented approaches

Goal-Oriented Requirements Engineering (GORE) refers to the use of goals for requirements elicitation, evaluation, negotiation, elaboration, structuring, documentation, analysis, and evolution [255]. In this section, we review the two dominant paradigms for GORE. The first of these is focused on discovering *what* goals and requirements are; this paradigm is characterised by the KAOS framework. The second is characterised by understanding not only what goals are, but *why* they are necessary. Both approaches are concerned with *how* goals are operationalised into more granular goals and operations.

2.3.2.1 KAOS

The KAOS (Knowledge Acquisition in autOmated Specification) method [68] is a systematic approach for analysing, specifying, and structuring goals and requirements. This approach involves modelling goals from both a top-down and a bottom-up basis.

KAOS defines a goal as a prescriptive statement of intent that a system should satisfy through the cooperation of its agents [255]. The agents are components of a larger system, and may be human or machine. In this nomenclature, a requirement is defined as a goal under the responsibility of a single agent of the software-to-be. Consequently, a requirement is a refinement of a goal. Both goals and requirements may have attributes associated with them, such as a name, a formal definition, priority, or source.

Goals can be behavioural or soft. Behavioural goals prescribe intended system behaviours and may be *Achieve* or *Maintain* goals; achieve goals prescribe intended behaviours which must hold sooner or later, while maintain goals prescribe intended behaviours which must always hold. Soft goals prescribe preferences among possible system behaviours, and are usually prefixed with keywords such as *increase* or *reduce*, rather than *achieve* or *maintain*.

KAOS places emphasis on semi-formal and formal reasoning, and goal satisfaction. A KAOS goal model is an annotated graph of goals linked via AND/OR links, as illustrated in Figure 2.6. For example, in the case of AND links, all sub-goals linked to a parent goal need to be satisfied before the parent goal is itself satisfied. Goal associations are not only restricted to other goals. Goals may be refined by Domain Properties: descriptive statements about the problem world. Goals can also be operationalised by system operations, which describe changes to the system's state; operations in KAOS fulfil a similar role to operational schemas in Z [230]. Agents can also be assigned responsibility to goals, as illustrated in Figure 2.6.

In addition to the goal and obstacle models, several other model types are supported by KAOS. These include a UML class-diagram for modelling domain-specific concepts, and a KAOS notation based agent model, representing agents and their assigned goal or object responsibilities.

Obstacles and anti-goals In addition to being refined to sub-goals, goals in KAOS may conflict with obstacles: conditions representing undesired behaviour and preventing an associated goal from being achieved [256]. Obstacles can be modelled using an AND/OR refinement tree, where the root of the tree is an obstacle node associated with the goal it obstructs. Letier [154] proposes identifying obstacles by applying HAZOP-like guidewords to goal definitions; for example, this may include replacing an instance of *more* in a goal definition with an alternative like *less*.

A variant of this approach has been proposed by van Lamsweerde [254] for dealing with intentional, rather than accidental, obstacles. These obstacles are described as *anti-goals*, and represent threats.

KAOS criticisms f KAOS shares two weaknesses with the Problems Frames approach, specifically that KAOS is agnostic to the process of eliciting requirements, and the modelling context is assumed to be a single, operating environment. The weakness of *elicitation agnosticism* is more profound with KAOS, however, because the use of security argumentation with [117] and risk analysis with [211] goes some way to sanity checking requirements data before modelling. With KAOS, however, validating the process of modelling the goals provides no assurance that the goals themselves are valid.

KAOS goals contain a number of different attributes, including the potential to formally specify the goal itself. Nevertheless, compared to the techniques described in Section 2.3.2.2, the KAOS modelling notation is not very expressive. As such, the human impact of design decisions may not be immediately apparent using models which appeal to system, rather than user, goals.

A final criticism of KAOS as a Goal-Oriented Requirements Engineering technique is an appeal to Richard Gabriel's *worse-is-better* philosophy. Despite its flaws, the agent-oriented goal modelling community is, arguably, larger than the comparative KAOS community. As a result, despite their lack of comparative elegance, the design flaws or use issues may be addressed as time progresses, thereby leading to improved approaches.

2.3.2.2 Agent-oriented approaches

The genesis of the agent-oriented approaches described in this section was Chung et al.'s work on the Non-Functional Requirements (NFR) Framework [51]. The underpinning premise of the NFR Framework is that the design of systems is often contingent on non-functional requirements such as security, accuracy, and performance. It is, therefore, sensible to design with these concerns in mind as early as possible.

Chung et al. proposed augmenting the elicitation of functional requirements with activities for identifying, decomposing, and operationalising goals reflecting these non-functional concerns, where *goals* correspond to obligations to be accomplished or satisfied. However, because non-functional requirements are often subjective and may lead to side-effects as a result of their implementation, these are best reasoned about dialectically with respect to other goals. As such, goals are *satisficed*, i.e. satisfied at some level, rather than satisfied, and are considered to be *soft-goals*: goals with no clear-cut definition and/or criteria as to whether it has been satisfied or not.

i* The i* (Intention STrategic Actor Relations) framework is an agent-oriented approach to Requirements Engineering centred on the intentional characteristics of an agent [188]. i* was devised to deal with the early stages of Requirements Engineering, preceding the initial specification of requirements. The objective of i* is to model and analyse stakeholder interests, and how they might be addressed or compromised in various system and environment alternatives [267]. To this end, certain concepts are key to i*:

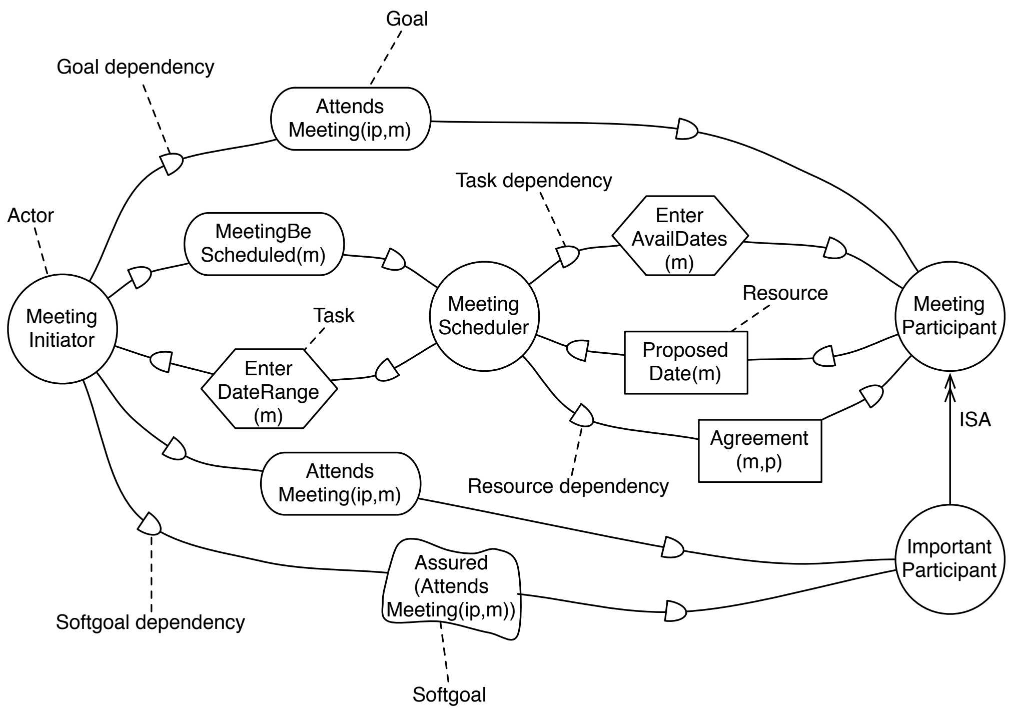


Figure 2.7: i* strategic dependency model example

- *Goals* are conditions in the world that stakeholders would like to achieve.
- *Actors* represent autonomous agents (people, hardware, or software) with intentional goals.
- *Tasks* represent specific procedures performed by actors.
- *Resources* are physical or informational entities.

If an actor intends to achieve a goal then this can be achieved by either carrying out a task, or by relying on another actor for the achievement of that goal, or a resource or task necessary for an actor to achieve the goal. These can be modelled using *Strategic Dependency* models, an example of which is provided in Figure 2.7. This example models meeting scheduling with a computer-based scheduler, and illustrates how a meeting initiator depends on meeting participants to achieve an AttendsMeeting goal, and a MeetingScheduler for a meeting schedule to be initiated. In turn, the meeting scheduler depends on the initiator carrying out the task of entering a date range.

i* also supports a *Strategic Rationale* model, which explains the internal linkages connecting the strategic dependencies. In the example in Figure 2.8, the rationale for a manual meeting scheduling system is described. The bounded areas represent the internal rationale model for the meeting initiator (left) and the meeting participant (right). The model shows how tasks can be decomposed into sub-tasks,

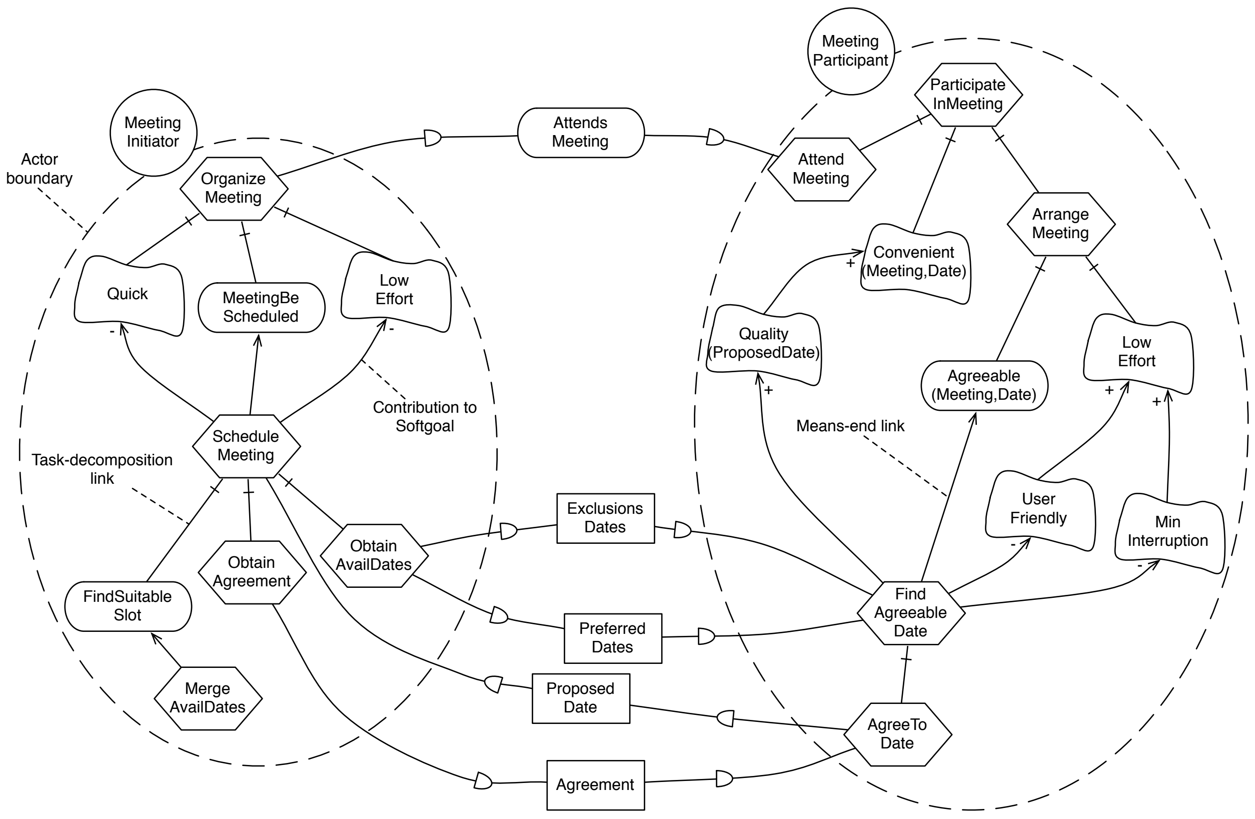


Figure 2.8: i* strategic rationale model example

e.g. ParticipateInMeeting, and how tasks can contribute to goals. For example, from the perspective of a Meeting Participant, the Find Agreeable Date goal contributes to the achievement of arranging an agreeable meeting, and positively contributes to the ProposedData quality goal, but negatively contributes to a User Friendly soft-goal.

Although devised as a modelling framework rather than a design method, i* allows high-level design decisions to be modelled and discussed at an early stage. Once any issues raised are settled, a prescriptive requirements specification can be developed for systems development [267].

Tropos Tropos [37] is an agent-oriented design process designed to minimise the gaps between goal modelling, architectural design, and systems implementation. Minimising these gaps is made possible by applying the core concepts of actors, goals, tasks, and related concepts across all stages of the design process.

Tropos is characterised by the following activities.

- Early Requirements Analysis. Stakeholders are identified and modelled as social actors who depend on each other for goals to be achieved, plans to be performed, and resources to be made available.

- Late Requirements Analysis. The model developed in the first phase is extended, and an actor representing the system-to-be is introduced, together with dependencies between these actors and the system actor; these dependencies represent the system's functional and non-functional requirements.
- Architectural Design. A system architecture is defined with inter-connected sub-systems. In this architecture, system actors map to software agents, and each agent is characterised by specific capabilities.
- Detailed Design. Specific agent capabilities and interactions are modelled, and detailed design is mapped directly to code.

A number of different models are associated with Tropos; the notation for these models is, in several cases, based on i*. *Actor Models* describe the relationships between different concepts, and *Dependency Models* describe the dependencies between different social actors; these models are analogous to Strategic Rationale and Strategic Dependency models respectively. During later phases, UML activity and interaction diagrams are used to model specific plans of actors and their interactions.

Security extensions Both i* and Tropos were designed to be agnostic of any particular type of non-functional requirement. Recently, however, security extensions for both approaches have become active areas of research.

Elahi and Yu [80] argue that actors, resources, and tasks are synonymous to assets because they hold value to an organisation, and can be targeted by attackers. Equally, the semantics of resources and tasks do not rule them out from also being represented as countermeasures to possible attacks. Elahi and Yu propose the addition of new nodes for malicious actors (attackers), tasks, and goals to aid clarity. In addition to these nodes, additional concepts are proposed to deal with vulnerabilities and attacks. A vulnerability is modelled as a black, filled circle: a metaphor for a hole; the vulnerability elements adorn vulnerable tasks or resources. Attacks are modelled by introducing an *exploit* relation between malicious actions and vulnerabilities. A modelling process for modelling these nodes has also been proposed. This entails eliciting stakeholders and system actors, together with their goals, tasks, resources and dependencies, and adding vulnerabilities to certain tasks and resources. Threat modelling is then used to determine how attackers exploit the identified vulnerabilities. Following a risk assessment, the models are augmented with details of countermeasures mitigating these attacks, before the model is re-evaluated.

In more recent work, Elahi and Yu [79] have proposed annotating i* dependency relations with information about trust relationships, trust rationale augmenting these dependency relations with information about trust, and annotating the relations with rationale about the trust relations; this annotation involves re-using i* "belief" elements.

Secure Tropos [173] extends the Tropos method in several ways. First, several concepts are added to the modelling notation. These include Secure Constraints which restrict possible solutions, Secure Dependencies for adorning dependency relations with security constraints, and Secure Goals for achieving a security constraint. Security Constraints are modelled by re-using i* belief nodes, whereas Secure Goals re-use the respective model nodes for goals, but their node names are prefixed with the text *(S)*. In addition to these new concepts, a number of new modelling activities are added to support the different phases of Tropos. For example, Security Reference Modelling involves modelling security objectives for the system, together with possible threats and countermeasures to the threats. A security reference model re-uses the i* nodes for soft-goals, goals, and tasks to represent security goals, protection objectives, and security mechanisms respectively. A further additional model element, a pentagon, is added to represent threats, which may be associated with threatened security goals.

Criticisms Although people may have security goals, these are supplemental to their primary task goals. As such, it may not be appropriate to treat them both on an equal footing. Moreover, the notion of discrete security tasks may also be inaccurate because, again, security related actions may be interleaved with normal tasks. From a usability perspective, considering security as just another non-functional goal to be traded-off also stymies the idea that security and usability can be aligned.

A recent evaluation of the visual effectiveness of i* [171] found that its graphical complexity, the number of different elements used in the visual notation, is, at 17 node and edge types, nearly three times greater than a human's standard limit for distinguishing alternatives. Moody reports earlier work [180], which suggests that this complexity may significantly reduce the understanding of Requirements Engineering diagrams especially by novices. Worryingly, [80] expands the notation for security elements, while [173] overloads existing elements. Given the early stage at which i* is used, an expansive notation may hinder the adoption of these approaches.

The task of accurately modelling the social context with sufficient depth to address usability concerns may be unsustainable with i* derived approaches. Yu [267] observes that each i* model admits of only one perspective on an actor's reasoning, but each actor has its own model of each other's rationale. Consequently, a complete model would need as many SR models as there are actors, each from each other's viewpoints.

Yu [267] suggests that, if interviews are carried out, such models could be constructed from the perspective of each actor, however this leads to significant model management problems given the size and quantity of different models and the inevitable need to reconcile them. It is, however, accepted that the process of merging these different i* / Tropos models may yield useful knowledge about the domain and the social contexts of use [78].

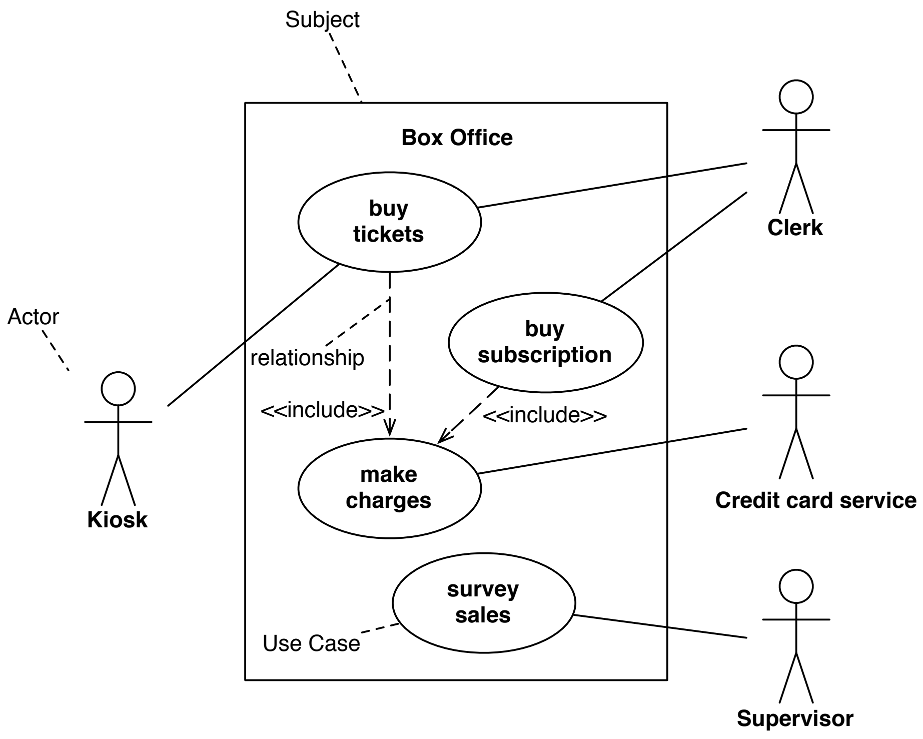


Figure 2.9: Use case model example

Finally, these approaches continue to lack the maturity afforded by industry take-up despite these approaches being the subject of active research for over 10 years. For example, recent work applying i* [158] found problems reconciling models in line with stakeholder views, difficulties identifying the most appropriate place to start modelling, and, ironically, the technique was less effective at eliciting dependencies between actors compared to the use of simple tables.

2.3.3 Use cases

Scenario-based approaches are popular in Requirements Engineering for activities such as eliciting, specifying, validating, and even documenting requirements. Arguably, the most well-known of these approaches are Jacobsen's Use Cases [144].

Use cases are sequences of actions a system performs yielding an observable result of value to a particular *Actor* : someone or something outside the system that interacts with the system [152]. These actions are considered to be algorithmic steps, which, together, constitute a specific flow of events through the system; this sequence of actions is described as an interaction between the system and the actor. Essentially, use cases are nothing more than scenarios, although a more formalised template for completing use cases is proposed by Cockburn [52]; this template includes the use case's goals, steps for the normal *happy-day* scenario, variations scenarios, which branch from the main scenario, exceptions, pre- and post-conditions, and particular constraints on the use case.

Use case diagrams can be used to graphically model use cases; an example of such a diagram is given in Figure 2.9, which models the use cases associated with a Box Office's booking system. In this example, arcs represent the bi-directional, interactional relationship between actors and use cases. Figure 2.9 also illustrates how a use case can make use of other use cases, i.e. the use cases for buying tickets and subscriptions can make use of the activities described in the *make changes* use case. In addition to including other use cases, the semantics for use case diagrams are also additional classes of relationships between use cases. To indicate the system's scope of analysis, a rectangle representing the scope's boundary may also be added.

2.3.3.1 Misuse and abuse cases

Sindre and Opdahl [223] have proposed extending the notion of use cases to support unwanted system behaviour. Such behaviour can be encapsulated by a *Misuse Case*: a sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity, and causing harm to stakeholders if the sequence is allowed to complete. To support the analysis of the described threat, Sindre and Opdahl propose adopting a detailed template similar to that prescribed by [52].

The simplest way of creating misuse cases is informed brainstorming, guided by a security expert asking questions about the system likely to have weaknesses; this activity mirrors the way attackers might think [127].

Misuse cases extend the use case diagram notation in the following ways. First, the actor node is represented as a stick figure with a black, rather than white, head. Second, additional relations between use cases and misuse cases are added, specifically a misuse case can *threaten* a use case, although a use case describing the application of a mitigating countermeasure may *mitigate* a misuse case. Røstad [205] proposed augmenting this notation further by adding an actor type to represent insider attackers, and a grey — rather than black — ellipse to represent vulnerabilities; the black ellipses are then used to represent threats carried out by either inside or outside attackers. An additional *exploit* relation is also proposed between threat and vulnerability ellipses to indicate that a threat exploits a particular vulnerability.

A similar negative extension to use cases is the Abuse Case: a specification of a type of complete interaction between a system and one or more actors, where the interaction results are harmful to the system, actors, or other stakeholders [166]. Like misuse cases, abuse cases are scenario-based and can be graphically represented using use case diagrams. The distinction between misuse cases and abuse cases is very subtle. Sindre and Opdahl claim that the textual templates used by both are different, abuse cases deal with families of misuse rather than specific threats, and abuse cases are not modelled on the same use case diagram as use cases. Because of this subtlety, we follow the example set by much of the literature and consider both as synonymous.

2.3.3.2 Criticisms

Despite its popularity in both academia and industry, the relationship between use cases and requirements is not entirely clear. The Inquiry Cycle Model [193] is an example of using use cases to envision how requirements are realised, although recent work on RESCUE (Section 2.4.1) uses use cases to elicit requirements. The situation becomes more confusing when considering that the Unified Process [152] represents requirements as use cases; this is also the case for agile software development methods such as Extreme Programming (XP) [29], which rely on lightweight scenarios called *User Stories* as proxy requirements [55].

One concern raised by [223] is the possible scaleability issue when dealing with a large number of threats, and that the underlying problem may relate to the process used to elicit security requirements and the need for better guidelines for prioritising requirements, rather than with misuse cases per se. However, what remains unclear is the usability impact of a countermeasure use case *mitigating* a misuse case, and what new vulnerabilities may be introduced as a result. It is possible that this impact may only be felt by examining the text of the respective use cases, but without a way of highlighting this, usability concerns may not contribute to the design process.

2.3.4 Other related security and usability approaches

2.3.4.1 Risk analysis

Mayer's work on the Information System Security Risk Management (ISSRM) modelling language [165] examines all the stages of information systems development. This has included work on aligning ISSRM concepts with techniques such as scenario-based support for risk analysis [161], and goal-modelling approaches [161]. This work has also attempted to integrate existing efforts on meta-modelling for risk analysis.

On first inspection, this approach helps capture contextual information—such as scenarios, and how stakeholder beliefs and goals contribute to specification and design decisions—implicitly supporting the capture of usability as well as risk and requirements data. Closer inspection, however, indicates that concepts are missing and others need re-evaluating. For example, usability concepts are not included within ISSRM, nor is any particular consideration given to the need for designing for different contexts of use. Moreover, ISSRM attempts to align concepts from its own conceptual model with those of other modelling approaches, rather than treating different approaches as orthogonal. For example, Mayer proposes aligning ISSRM asset concepts with the KAOS concepts of goals and requirements. The KAOS meta-model does, however, support concern links between different model types, which means that ISSRM assets are likely to be better aligned with concepts in an associated class model, which better reflect assets of value to a system.

2.3.4.2 Personas

Although there has been an implicit assumption that following Requirements Engineering practices, especially goal and scenario-based approaches, will lead to more usable systems, there has also been specific work on explicitly aligning Usability and Requirements Engineering practices.

Castro et al. [44] have used personas and scenarios to supplement Requirements Engineering activities. Advocates of personas propose complementing them with scenarios, which describe system behaviour from a persona's standpoint [61, 196]. Scenarios describe expected persona behaviours, making them an effective way of validating assumptions underpinning both the persona and the context in general. Participants are also encouraged to think in terms of scenarios at an early stage; this reduces the possibility of participants treating personas as elastic users. Castro and his colleagues also align concepts from Cooper's method for developing personas [61] with those used for eliciting, analysing, specifying, and validating requirements. Although many of the techniques used for developing personas appear to align with techniques for eliciting and analysing requirements, Castro is less prescriptive on how personas contribute to other stages.

Another persona-related contribution is Aoyama's method, which combines personas, scenarios, and goals [20]. The method involves developing personas using market research statistical techniques, supplemented with focus groups, to identify personas, identify scenarios and goals, evaluating scenarios against the primary personas perspective and his or her goals, and eliciting requirements from these scenarios. Aoyama's method facilitates the *plug-in* of goal models, but the approach is problematic for several reasons.

First, Aoyama does not prescribe how goal-modelling should be integrated into his method. Aoyama proposes specialising the concept of *Goal* with a *Goal of Use*, yet he does not describe how a goal is defined. In KAOS, a goal is a prescriptive statement of intent that a system must satisfy, but in i*, a goal is defined as the intentional desire of an actor. Because the definition of goal is ambiguous then, by extension, so is the definition of a *Goal of Use*.

Second, Aoyama indicates that requirements analysis is driven from the viewpoint of the persona. However, Pruitt and Adlin [196] argue that personas are best applied when they supplement other analysis rather than replace it. By driving the requirements analysis process from the perspective of a persona, rather than viewpoints more closely aligned to the problem domain, important requirements may be missed.

2.4 Specification frameworks

The advantages and disadvantages of the techniques in Section 2.3 highlight the need for process level guidance to integrate them. The need for such guidance has long been recognised by the Information Systems Development community, which distinguishes between a *method* and a *methodology.* A method is a concrete procedure for getting something done, whereas a methodology is a higher level construct motivating the need for choosing between different methods [133].

A paradigmic example of a methodology is Multiview [22], where information system development is carried out in five sequential stages: the analysis of human activity, the analysis of information, analysis and design of socio-technical aspects, design of the human-computer interface, and design of the technical aspects. Each stage is informed by a different view of the design process. For example, the first stage is concerned with the process for formulating a world view from a problem situation; this relies on techniques from Checkland's Soft Systems Methodology introduced in Section 2.1.1. The third stage is concerned with the impact of different system options from a social as well as a technical perspective; as such, the results of the first stage, and the systems analysis in the second stage provide input to participative design techniques [175] to explore these options. Although this pluralistic approach to design promises to take the best that each contributing perspective has to offer, Avison and Wood-Harper also acknowledge that the methodology is heavily contingent on the abilities of analysts to interpret the situation about Multiview's role in the intervention. While some element of contingency is always necessary, over reliance on it can be problematic from a security and usability standpoint. For example, the analysis of usability concerns is justified as a comparatively late activity because of the time spent reconciling issues in the first, second, and third stages in a Multiview method. However, the lack of specific guidance about using techniques as part of a larger method may lead to inappropriate application of techniques; this may sully the data contributing to usability analysis.

Outside of the Information Systems Development community, the terms method and methodology are used interchangeably. However, the principles of information system methodologies have been encapsulated in the several process *frameworks* that have, in recent years, emerged in Software, Security, and Usability Engineering. A framework can be defined as a structure composed of parts framed together [3]. Haley [116] succinctly defines a *framework* as a set of milestones indicating when artifacts should be produced, as opposed to a *process* describing the steps to be carried out to produce the artifacts.

The usefulness of frameworks for processes was alluded to by Parnas and Clement [192], who argue that prescriptive processes are inevitably *faked* by producing a set of artifacts. These manage what stakeholders might expect as an output of such processes, so processes are situated around the development of these artifacts. These process frameworks are also known as *software process models* or *process paradigms* [227]. The Rational Unified Process [152] (RUP) is an example of such a framework. RUP is organised around a process model encapsulating *best-practice* principles such as developing software iteratively, managing requirements, and visually modelling software. RUP prescribes a number of pre-defined configurations, but designers are encouraged to modify it to fit their own organisational needs. RUP is also supported by software *tool-mentors*, which provide guidance on performing steps of an activity which certain IBM software tools support.

We are unaware of existing frameworks dealing with both usability and security from a requirements perspective. There have, however, been processes and frameworks purporting to deal with each.

2.4.1 RESCUE

RESCUE (REquirements with SCenarios for a User-centered Environment) is a user-centered Requirements Engineering process [157]. Although not explicitly defined as a framework, the earlier phases of RESCUE afford leeway in technique application. RESCUE consists of the following four concurrent system engineering streams: Human Activity Modelling, i* system modelling, Use Case and Scenario Analysis, and Requirements Management.

Human Activity Modelling involves analysing the way work is carried out, and partitioning the analysis of the problem domain into different aspects, such as the work domain, control task, and social organisation. Based on empirical data elicited via observations or semi-structured interviews, a structured activity modelling template is created which details actors, goals, normal courses, and variations for different work activities. When the system boundary has been agreed, i* Strategic Dependency models are used to map the dependency network between actors, and Strategic Rationale models are used to describe the intentional description of activities, and the rationale relationships between actors and related resources, goals, and tasks. Use cases are then identified based on actor or system objectives, which are graphically represented in a use case model, and authored using a template based on style guidelines prescribed by [52]. These are presented to stakeholders in facilitated walk-through workshops; these are supported by ScenarioPresenter, an interactive, web-based tool for storing and guiding scenario walkthroughs [163]. Requirements elicited as part of RESCUE are specified using a template based on the Volere Requirements shell [200] and managed using commercial requirements management tools.

RESCUE implicitly assumes that a secure and usable system will result by following its prescribed guidelines. Security and usability are both considered as non-functional requirements, and while the activity-centric nature of RESCUE means it is likely that the system's core functionality will be situated for its actors, the results of specifying security requirements may circumvent these activities. As it stands, security requirements may lead to use case changes, but it remains the task of the analyst to spot use cases which may become unusable as a result of security constraints. Moreover, when potential problems are identified, the analyst needs to manually modify and maintain the traceability relations between the i* models and downstream artifacts. RESCUE also fails to stipulate specific techniques for dealing with security concerns, making it an exercise for the analyst to select both the appropriate techniques and points to apply them.

2.4.2 SQUARE

SQUARE (System QUAlity Requirements Engineering) [168] is a method which aims to build security into the early stages of a project lifecycle. The method is carried out within the context of a project, and applied by carrying out the following steps.

- Agree definitions: this involves agreeing a consistent set of security terms and their definitions, where possible from existing standards.
- Identify security goals: this involves agreeing project security goals in participatory workshops.
- Develop artifacts to support security requirements definition: these artifacts may include architectural models, use cases and use case models, and misuse cases, and are created by analysts knowledgeable with these techniques and/or the problem domain.
- Perform risk assessment: this involves selecting a risk analysis method, before identifying and categorising the threats security requirements need to mitigate.
- Select elicitation techniques: the most suitable security requirements elicitation techniques are selected based on the different project and domain specific factors.
- Elicit security requirements: this entails applying the selected techniques and eliciting verifiable requirements; this may involve face-to-face interaction with stakeholders.
- Categorise requirements: requirements are categorised as essential or non-essential, and of system, software, or architectural significance.
- Prioritise requirements: this involves selecting and applying a prioritisation technique with stakeholders, e.g. [149], and may be informed by the risk assessment previously carried out.
- Requirements inspection: this involves selecting and applying a requirements inspection technique.

Although the selection of elicitation techniques is described as contextual, the selection of techniques for developing artifacts is not. Moreover, while several artifacts are recommended, no explicit guidance is afforded on how different techniques might contribute to each other besides reports of SQUARE applications.

Recent work acknowledges that the term *security requirement* is often used [167], but these requirements could be any form of requirement provided it helps build security concepts into a system. Nevertheless, while requirements are derived from a project's high-level goals, and artifacts may cross-cut with non-security concerns, it is assumed that specifying requirements for secure systems is the sole reason for a SQUARE intervention. In reality, SQUARE is likely to be used as part of a larger project where security and non-security requirements, arising from security and non-security goals, are inter-related.

2.5 Tool-support

One of the consequences of inter-relating ideas is that the results of applying techniques from these areas have to be managed, analysed, and applied as a means to the desired ends, i.e. a usable and secure system design. In this section, we review the current tools for managing usability, requirements, and security concerns.

2.5.1 Usability design tools

Although designing usable systems requires an early focus on users and their goals, to many engineers, usability is synonymous with user interface design [219]. However, as Section 2.2.1 reported, usability is about more than just designing interfaces. Unfortunately, we currently lack tool-support allowing analysts and developers to inform security design with the kind of usability insights previously discussed.

Singh and Bartolo argue that narrative scenarios can convey the results of qualitative data analysis [224], and qualitative data can also be represented as personas. By describing how personas carry out these scenarios, we can represent usability data in a meaningful way to stakeholders and inform the subsequent analysis accordingly. Although there has been some related work on the use of personas and scenarios to support requirements analysis activities [44], and reliability analysis [36], there is no indication of how these artifacts are managed.

A step towards managing personas and the scenarios they participate in involves devising a suitable means of structuring and categorising this interaction. Such categorisations might also help measure the impact to usability of security design decisions, and vice-versa.

2.5.2 Security Requirements Engineering tools

Many tools for Security Requirements Engineering are general requirements management tools which have been augmented for security. Such tools are often based on the spreadsheet metaphor, where a table is used to enter the attributes of a natural language requirement. By applying this metaphor, requirement attributes, such as its description, type, and rationale, can be quickly specified. Certain requirements management tools, such as DOORS [129], contain scripting frameworks allowing augmentation with additional functionality. For example, using its DXL scripting framework, Alexander has extended DOORS to support use case [10] and misuse case [9] specifications. Unfortunately, the generic strength of a requirements management tool is also its weakness; the lack of distinct semantics means an analyst must manually maintain traceability links between requirements and non-requirements artifacts.

Model-based approaches support traceability between different artifacts. If a tool conforms to the requisite meta-model then, as data is entered into it, the data can be structured in a manner that facilitates automated analysis. Such approaches also help visualise this analysis and, frameworks allowing, can be combined to form approaches, which integrate different techniques. Existing tool-support for model-based approaches exist for risk analysis [70, 169], goal modelling [199, 186, 189], and analysing refined representations of security requirements [265]. The task of integrating different model-based approaches for Security Requirements Engineering is, however, non-trivial. The first problem is the model-driven paradigm itself. A given approach may choose to reduce visible complexity by simply moving it elsewhere [115]. For example, UMLSec [147] depicts a security requirement as a stereotype, and is concerned with the requirement's deployment rather than its specification. Another problem is the diversity of the models to be integrated; tools are often based on different meta-models, making understanding and agreeing interfaces difficult. If we assert that a misuse case *threatens* a use case, do we agree what it means for the use case to be threatened? Does the misuse case threaten the work carried out by the use case, or the assets associated with it?

Houmb et al. [128] examined some of the challenges faced when integrating tools based on different techniques. As part of their SecReq method, Houmb et al. integrated their HeRA tool [151] with the Common Criteria [139] and UMLSec. The Common Criteria is a standard which forms the basis of evaluating the security of an agreed product, known as a *Target of Evaluation*, according to certain levels of assurance. Houmb's approach involved using pre-defined functional requirements from the Common Criteria [141] to formulate security requirements. The process of eliciting and refining these requirements was supported by HeRA by suggesting heuristics based on UMLSec stereotypes. These requirements were subsequently modelled as UMLSec stereotyped use case diagrams, and elaborated as activity diagrams. Some of the integration problems found by Houmb et al. include:

- Expertise in different areas such as Requirements Engineering and security was needed to refine useful requirements from the abstract Common Criteria security goals. Heuristics on their own were not enough.
- Although UMLSec tool-support could verify whether requirements (modelled in UMLSec) are verified by a design, the approach does not provide any analysis or modelling support for the elicitation of requirements. As such, although requirements can be verified against design, they could not be validated against a system's goals as easily.

Recent work on conceptual models for Security Requirements Engineering, e.g. [165], are a step towards resolving this integration problem, but these still lack treatment for usability concepts such as tasks and personas. This resulting lack of tool support is a major obstacle for integrating Software Engineering approaches with HCI [219].

One strategy for integrating these approaches is to consider how Security Requirements Engineering and HCI might complement each other. This section has already discussed how approaches from HCI can inform Security Requirements Engineering. Thimbleby [242] argues that usability approaches are necessary, but far from sufficient for critical systems. The sheer size of the state space associated with interactive devices is so big that empirical evaluation on its own is unsustainable. It is, however, possible, to supplement usability analysis with basic technical methods.

2.5.3 Visualising design

Before stakeholders can measure the impact of usability of secure system design decisions, they need to understand the rationale contributing to them. Previous work has illustrated how different visualisation techniques can both explain the results of analysis, and explore the resulting impact. While we are unaware of work purporting to visualise the analysis and resulting impact of usability and security, there has been work on independently visualising analysis in each area.

Techniques from information visualisation have been used to support security risk analysis [124, 106] and risk analysis as part of the design process [94]. Hogganvik's thesis on the subject of visualising risk analysis concludes that parsimony is important with respect to the number of model symbols and colours employed, and that colour may be a useful means of distinguishing the value of different risks.

Gandhi and Lee used visual metaphors to visualise the results of requirements-driven risk assessment [106]. *Cohesive Bar Charts* are used to represent requirement categories and risk components, where bars represent different requirement categories. The bar colour is used to model compliance, its height represents the correlation of a category with other requirements, and its width represents the number of risk components related to a category; bars are also ordered to prioritise categories based on the analysis data. Feather et al. [94] also uses bar charts to portray information from Defect Detection and Prevention (DDP) models [64] to make problematic areas more evident.

Given the contextual nature of security design and the limited space available to visualise a potentially vast amount of data, there is a need to visualise only the information which might be relevant to a particular context. Consequently, there is a need to look at the tool generating the model, rather than simply relying on the model itself. In his seminal paper on the elements of interactive information visualisation systems [222], Shneiderman presents his Visual Information Seeking Mantra: overview first, zoom and filter, and then details-on demand. In particular, this mantra entails:

- Overview: this involves obtaining an overview of the big-picture in a model.
- Zoom: after identifying regions of interest, the focus moves onto a particular item for a closer inspection.
- Filter: while zooming in, uninteresting items are filtered out.

- Details on demand: when a specific item of information is pinpointed, further details can be made available.

Although there is a growing body of work in security data visualisation, e.g. [58], and visualisation for Requirements Engineering activities, e.g. [62], we still lack practical tools that designers can use, which both reason about security and requirements concerns while, simultaneously, respecting Shneiderman's Visual Information Seeking Mantra.

2.6 Chapter summary

In this chapter, we have reviewed the current state-of-the art in related work contributing to the design of usable and secure systems. Summary points from this review are as follows.

- The need for usable security has been long recognised by the Information Security community, however previous work has adopted an ad-hoc approach to synthesising design techniques from security and HCI.
- The User-Centered Design community has considerable expertise in fostering user experiences which are situated for users, their goals, tasks, and contexts of use, yet these approaches are largely craft-based. Although work in Human-Centered Software Engineering is gradually making this knowledge available to software engineers, these approaches, if not managed carefully, risk disenfranchising stakeholders and losing important information about the security context.
- Requirements are a nexus between HCI and Information Security, yet confusion is evident about what security requirements are, how they should be expressed, and how they should be best elicited and analysed. There is also a dichotomy evident in these approaches, with regards to how user-centric or system-centric they should be. System-centric approaches, such as Problem Frames and KAOS, can lead to clear, unambiguous requirements but are agnostic to the problem of eliciting requirements, and any context beyond a single, operating environment. On the other hand, user-centric approaches, such as agent-oriented approaches and use cases, focus on the problem of eliciting requirements data from users, but suffer from scaleability problems in large social contexts, and — in extending the approaches for security — may also suffer from technique usability problems.
- We lack techniques and frameworks for jointly dealing with usability, requirements, and security concerns. Existing tool-support for usability engineering is weak, and existing tool-support for security engineering suffers from scaleability problems when integrated with complementary approaches. There has been previous work in visualising security design, and a growing body of related work

by the Requirements Engineering Visualisation community, however current tool-support fails to satisfy the core principles for interactive information visualisation systems.

Chapter 3

Methodology

In this chapter, we introduce the methodologies used for evaluating the contributions of this dissertation. We examine how the HCI, Information Security, and Requirements Engineering research communities treat the concept of design as a research topic. Informed by these different approaches, we present the research methodologies used within this dissertation.

3.1 Research approaches

3.1.1 HCI research approaches

Research methodologies in HCI tend to focus on evaluating the usability of a design artifact, rather than researching the intrinsic nature of the design activity itself. This position is founded on the assumption that usability in a system design will follow if Usability Engineering techniques are applied, and the appropriate evaluation techniques are used to provide feedback to Software Engineering activities. Two evaluation approaches have been adopted by the HCI community for this purpose: Formative and Summative Evaluation [216]. Formative Evaluation is carried out during design, and evaluates whether a design meets the expectations of users; an example of a Formative Evaluation technique is prototyping. Summative Evaluation is used to assess usability quality, often against a usability specification, once design activities have been concluded. Summative Evaluation is usually synonymous with tests in a controlled usability laboratory, although questionnaires might also be construed as a summative evaluation technique if used after, rather than during, the design process.

There is, however, a recognition that making the jump from design data to design is a difficult problem, although some argue that this is a creativity and sense-making issue rather than a research problem per se [32]. Nonetheless, the HCI research community agrees that methodology research in design is immature [43]. Recent work by Zimmerman et al. [269] has attempted to draw together the

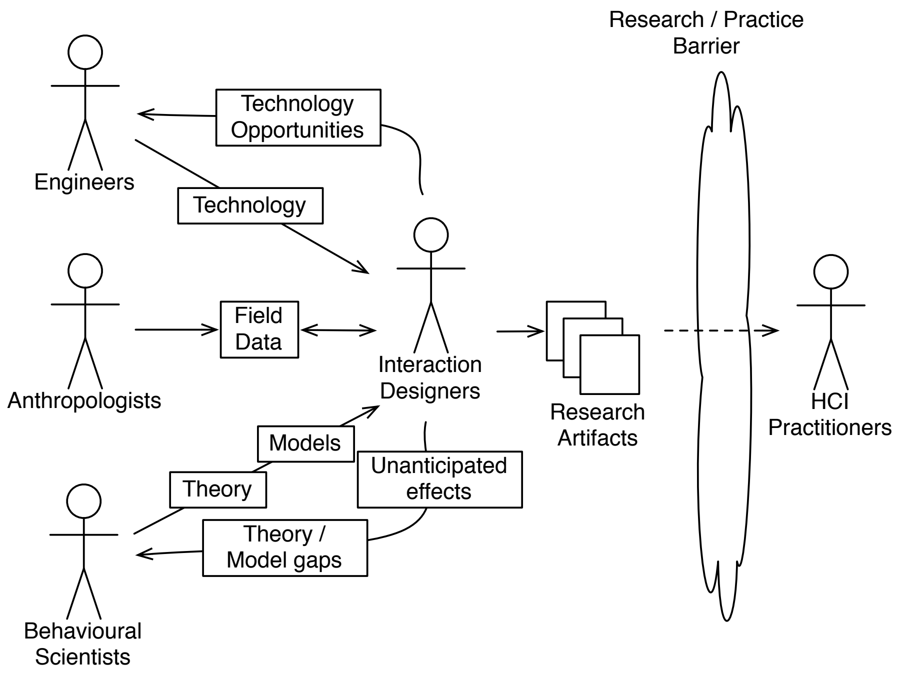


Figure 3.1: Model of interaction design research

different perceptions of design held by the HCI community. They propose a model of research through design, where interaction designers integrate *true* models and theories from behavioural science, with the *how* knowledge afforded by engineering innovations, with all explorations are grounded in *real* knowledge produced by field researchers. This model is illustrated in Figure 3.1.

3.1.2 Information Security research methods

The security research community has, traditionally, inherited many ideas from the safety community. Many aspects associated with the design of a safety-critical systems are applicable to secure systems. Hazards — states or conditions leading to an accident or loss — are conceptually similar to threats [31]. Berry has also observed that the difficulty in identifying threats is identical to that of identifying hazards. Similarly, Fault Tree Analysis [1] is frequently used in safety engineering to identify possible causes of particular hazard; these are analogous to attack trees [212] where the root of a tree is a security threat rather than a safety hazard. From a broader design perspective, Brostoff and Sasse [38] argue that similarities between safety and security domains, such as their consideration as secondary goals and the attribution of problems to operators, motivate the use of safety models for security design. However, Brostoff and Sasse also acknowledge that one feature distinguishing security from other disciplines is its adversarial nature. Unlike, for example, Safety Engineering, which is concerned with unintentional system failures and the resulting hazards to human life, Information Security is concerned with intentional attacks by a myriad of possible attackers. Although the importance of security knowledge is universally agreed, the topic of how this knowledge should be generated is divisive.

In a recent panel on this subject [164], Maxion et al. divided security research approaches into two separate areas: theoretical and experimental. Theoretical cyber-security research is based on eliciting and validating formal, mathematical models, while experimental cyber-security relies on the use of scientific method to demonstrate causal relationships. However, current approaches for experimental security research do not follow what should be an acceptable model. Arguably, current research "best practice" involves devising an idea for a new tool, building the tool, attacking a system, demonstrating that the tool repels the attack, and writing up the results. However, having an idea does not constitute forming a hypothesis, building and deploying the tool does not constitute experimentation, and demonstrating that a tool repels a particular attack does not draw conclusions about the results. For this reason, Maxion et al. believe a return to classic scientific method is necessary for experimental research, as the body of scientific security knowledge can only be obtained by eliciting causal relationships.

Maxion et al.'s approach suggests that the necessary causal relationships can be elicited from the context of experimentation using a linear model of scientific method, however Butler [41] argues that eliciting this research data is difficult; the reasons include limited access to people and data in organisations, and the management perception that the benefits of research do not outweigh its costs.

Fléchais [97] has proposed using social-science research methodologies for developing and evaluating security design methods. These methodologies assume socially complex organisational relationships between people and technology, and attempt to align research with organisational benefit. In [97], Fléchais demonstrated how Action Research [155] was used to evaluate the AEGIS design method, and how Grounded Theory [235] was used to elicit general findings about security design based on the empirical data collected during several case study interventions.

3.1.3 Requirements Engineering research methods

Because Requirements Engineering's concerns are in both the problem and the solution domains, it is a methodologically broad area. In their roadmap paper on Requirements Engineering, Chen and Atlee [48] characterised the research strategies currently adopted by Requirements Engineering. From a design perspective, the strategy of most relevance to this dissertation is *Engineering*: the development of processes for efficiently applying research solutions in practice. A similar, but more general, taxonomy of Requirements Engineering research practice [262] was proposed to classify Requirements Engineering research papers; this contains a *solution of proposal* category, which describes a new technique, its intended use, and an illustrative exemplar. This approach is not, however, considered a research activity.

Reflecting on this past work, Akkermans and Gordijn [8] criticise this apparent trivialisation of design technique research. They argue that treating engineering as an application of solutions to practical problems fails to deal with the difficulty of framing a problem in context, and the common practice of fitting known solutions to the problem at hand, e.g. the application of Design Patterns that codify solutions for known structures of problem [105]. Akkermans and Gordijn and other Requirements Engineering researchers [261] have considered research into design in Requirements Engineering as a *Design Science* problem. Design science recognises that the science of design is a problem-solving process. Hevner et al. [121] propose the following several guidelines when carrying out design science research. These include the necessity for research to produce a viable artifact, research must yield solutions to important and relevant problems, and research must contribute to knowledge in the area of the artifact, design foundations, or design methodologies.

3.2 Proposed research

From this review of methodological perspectives, we can conclude that our research approach needs to deal with the following considerations:

- The research needs to be used to solve real problems and, where possible, should be carried out in the context within which the validating artifact will be situated.
- The research output needs to contribute knowledge back to the theory and models applied in the design process.

The proposed research approach adopted in this dissertation is described in more detail in the following sections.

3.2.1 Research methods

In this section, we describe the research methods used as part of this thesis.

3.2.1.1 Sketches and high-fidelity prototypes

Because they elicit feedback about the design process, sketches and high-fidelity prototypes are both formative evaluation techniques. However, Zimmerman et al. [269] have observed that these can also be used to clearly communicate research contributions.

Although sketches have been interpreted as *low-fidelity* paper prototypes by some texts, e.g. [32, 194], Buxton [42] argues that each focuses on different activities. The low investment needed for sketches affords many opportunities for exploring, learning, and developing a deep understanding of an undertaking. Sketches are also disposable because the investment is in its concept rather than the execution.

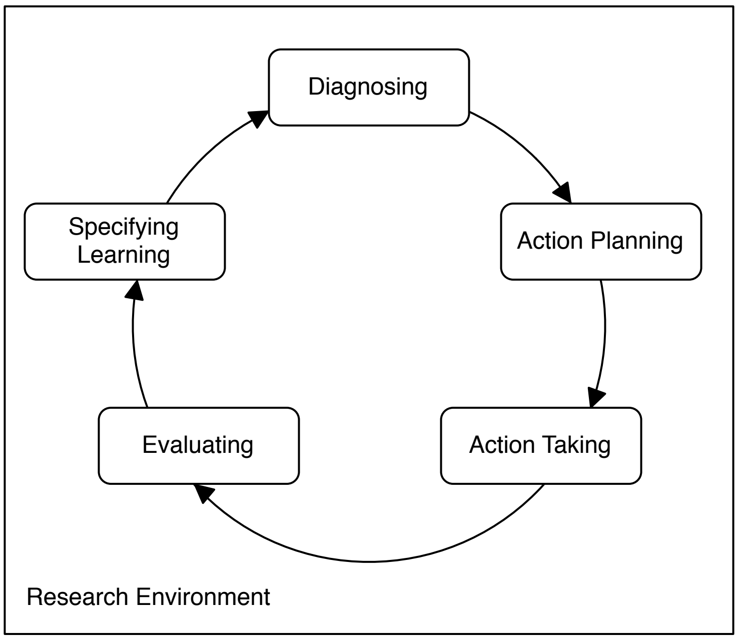


Figure 3.2: Baskerville's action research cycle

In contrast to sketches, *high-fidelity* prototypes are constructed to resemble a product rather than a concept. Unlike a real product, however, they are constructed using rapid development environments, such as scripting languages. These prototypes are also known as *evolutionary* prototypes because they evolve towards a final system as the design activity progresses.

3.2.1.2 Action research

The term *Action Research* was coined by Lewin [155] who describes it as "comparative research on the conditions and effects of various forms of social action, and research leading to social action." As an interventionist approach, Action Research is a self-reflective form of inquiry in that the researcher gleans knowledge about his or her role in the *social action*, as well as knowledge about how valid the social action might be. Action Research is an iterative research approach involving the planning of an intervention, carrying it out, analysing the results of the intervention, and reflecting on the lessons learned; these lessons contribute to the re-design of the social action and the planning of a new intervention.

Although primarily used in social science research, Action Research has also been used to validate security design methods, such as ORION [145] and AEGIS [97]. The Action Research methodology used in this dissertation is based on that proposed by Baskerville [27], who breaks an intervention into the research cycle illustrated in Figure 3.2.

The Action Research intervention is situated within a specified and agreed *Research Environment.*

This environment describes the rules of the intervention, including the entry and exit criteria, confidentiality agreements, ethical approval, and the responsibilities shared between the client and the researcher.

The *Diagnosing* phase involves identifying the problems motivating the intervention. The client may stipulate their interpretation of these problems, but the researcher also needs to undertake some form of empirical or conceptual investigation to develop his or her own assumptions.

The *Action Planning* phase involves devising the nature of the intervention that will relieve or improve the identified problems. This involves agreeing the desired future state, and the changes necessary to achieve this.

The planned intervention takes place during the *Action Taking* phase, according to some form of intervention strategy. This strategy may involve the researcher being an active participant in the intervention; alternatively, the researcher may provide explicit or implicit guidance to other participants and observe the outcome. Irrespective of the strategy, the researcher collects data about the intervention for subsequent analysis.

Once the intervention is complete, the researchers and practitioners evaluate the outcome as part of the *Evaluating* phase; this involves questioning whether the intervention was the sole cause of success (or failure).

Although cyclically the final phase, the *Specifying Learning* phase takes place on an on-going basis, and involves re-applying lessons learned during the intervention. This may involve recommending organisational changes, using the knowledge to inform the approach to take for future interventions, and reporting general insights to the scientific community.

3.2.2 Approach

The research approach adopted by this dissertation is split into an approach for the framework contributions (Chapters 4, 5, and 6), followed by an approach for the practical contributions which validate the framework (Chapters 7, 8, and 9).

3.2.2.1 NeuroGrid specification exemplar

To support the research approach for the IRIS framework contributions, we present a specification exemplar to illustrate the framework. Specification exemplars are self-contained, informal descriptions of a problem in some application domain, and are designed to capture the harshness of reality [95]. These are designed to advance a single research effort, promote research and understanding among multiple researchers, and contribute to the advancement of software development practices. To provide an initial validation of the theoretical research contributions, and to put these contributions in context, we have prepared a specification exemplar based on the UK Medical Research Council funded NeuroGrid project.

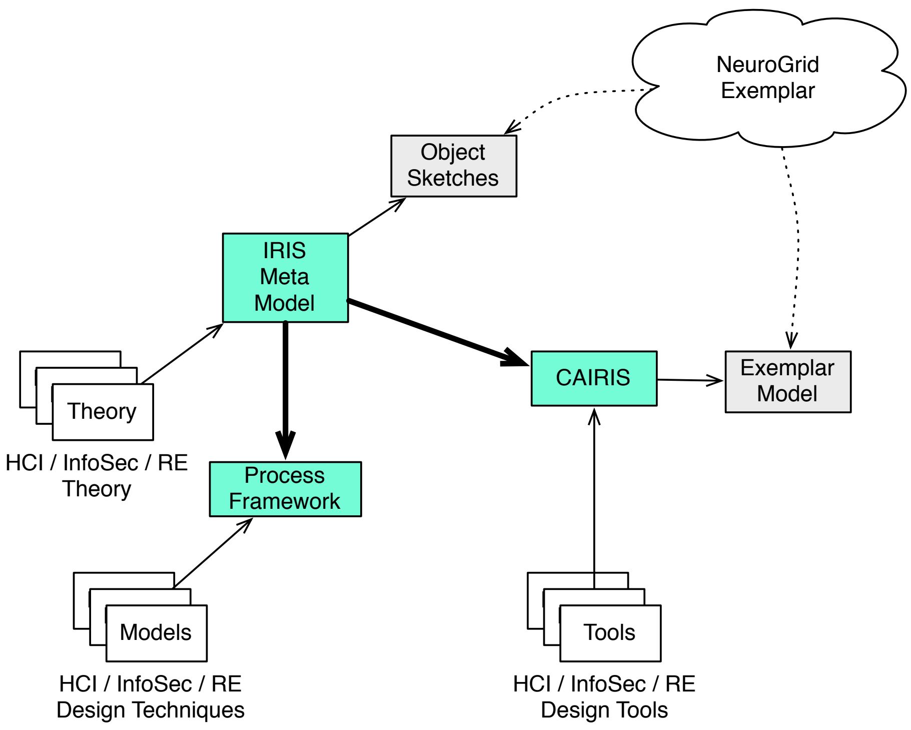


Figure 3.3: Research approach for eliciting and validating IRIS framework contributions

The aim of the NeuroGrid project was to develop a Grid-based collaborative research environment to enhance clinical researcher collaboration [108, 253]. NeuroGrid was used by three clinical exemplar projects: Stroke, Dementia, and Psychosis. The sensitivity of this data and its distributed nature drove the need to find secure and effective ways of accessing and managing this data. Access to the NeuroGrid web service interfaces was controlled by X.509 digital certificates [134]. Data access was facilitated by OASIS XACML-based access control policies [111], specified by the data owners within each exemplar, and manually hand-crafted by the NeuroGrid security team.

Although presented as an exemplar, empirical data was collected from participants in the NeuroGrid project as part of a qualitative data analysis carried out by the author [88].

3.2.2.2 Framework contributions

The research approach taken for developing and validating the theoretical contributions of this dissertation is illustrated in Figure 3.3.

The literature review is used to guide the selection of relevant theory from HCI, Information Security, and Requirements Engineering. Concepts from this theory are mapped to the IRIS meta-model. To illustrate the contribution made by this meta-model, meta-model concepts are sketched as instantiated objects; these objects are based on domain objects from the NeuroGrid exemplar.

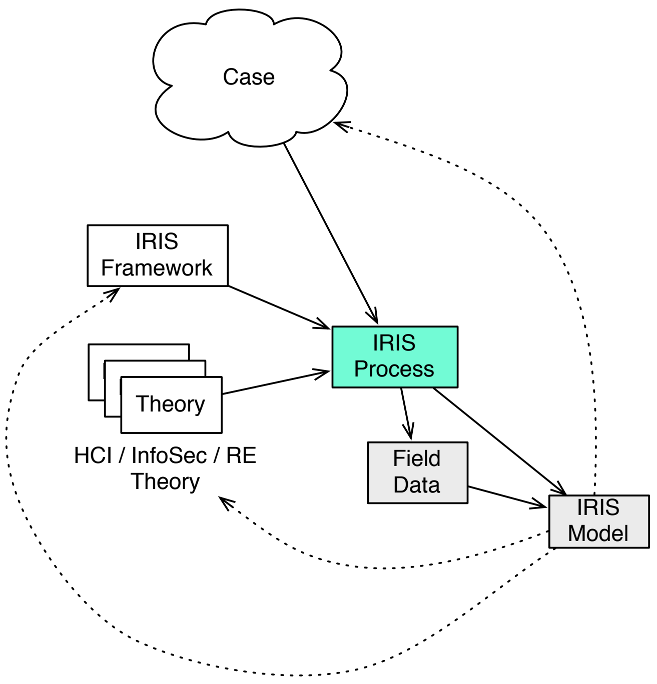


Figure 3.4: Research approach for validating IRIS framework in case studies

The IRIS meta-model provides information about the concepts needing to be collected as part of a usable and secure design process. The meta-model does not, however, provide guidance on how these concepts should be elicited. Techniques associated with the meta-model concepts are presented, together with guidance on how these need to be modified for inter-operability with the IRIS framework.

Informed by the meta-model, and additional information and tool-support from the literature, the software architecture and design principles for tool-support are elicited. This architecture is validated by developing a software prototype which implements this and, using the tool, building a model based on the NeuroGrid exemplar.

3.2.2.3 Practical contributions

To validate the IRIS framework, three validating *confirmatory* case studies are carried out. These are empirical inquiries within a real-life context that are used to test existing theories, and differ from *exploratory* case studies which investigate phenomena to derive new hypothesis and build theories [266]. In each case study, the *case* is the researcher and team of stakeholders developing the specification, and the *theory* being confirmed is the IRIS framework. The approach taken for each case study is illustrated in Figure 3.4.

Each case study is carried out in the context of an Action Research intervention, where IRIS is used to elicit and specify requirements for a real-world problem that needs to be addressed. A synopsis of

Chapter	Case Name	Synopsis
7	Control Software Repository	Software requirements for a software repository used to store instrumentation software. This software controls the assets in a water utility company.
8	MDR Security	Security Requirements for a Meta-Data Repository (MDR) associated with a portal for sharing longitudinal study data.
9	Plant Operations Security Policy	An Information Security Policy for treatment plants in a water utility company.

Figure 3.5: Case study chapters

these cases is provided in Figure 3.5:

For each intervention, the characteristics of the organisational context are used to inform the selection of design techniques which form part of an IRIS process. This process yields field data in various forms, such as documentation artifacts about the case study organisation, observational data, and interview transcripts. Collectively, this data and the IRIS process derives an IRIS model representing the requirements and supporting artifacts for the case. The insights from the intervention re-inform both the IRIS framework, and the case study context. In some cases, the intervention also leads to insights into the design techniques forming part of the IRIS process. These are described in more detail in the case study chapters.

3.3 Conclusion

In this chapter, we have reviewed the research approaches in the HCI, Information Security, and Requirements Engineering literature and, based on this, identified criteria that can usefully be used for selecting research methods. Based on this, we have described the research methods used in this dissertation and, building on these, presented an approach for validating the IRIS framework and other practical contributions.

Chapter 4

Meta-Model for Usable Secure Requirements Engineering

This chapter presents a conceptual model for usable secure Requirements Engineering. This work builds upon practical work in usability design, and recent research on meta-models for Security Requirements Engineering, to help structure and manage Usability, Security, and Requirements Engineering activities in different contexts.

We present an overview of the conceptual model itself, before describing each view of the meta-model: Environment, Asset, Task, Goal, Risk, and Responsibility. For each view, we present and justify the related concepts and their relationships. We illustrate each aspect of the meta-model using examples from the NeuroGrid specification exemplar described in Section 3.2.2.1.

4.1 Overview

Based on our review in Chapter 2, we have identified a number of core concepts within the Security, Usability, and Requirements Engineering literature. Defining these concepts and their relationships in a single diagram would be difficult to view and explain. Therefore, for clarity, we have sub-divided the IRIS meta-model into six views: Environment, Asset, Task, Goal, Risk, and Responsibility; these correspond to different perspectives of a secure system's context of use. Each view is modelled using a UML class diagram.

Although each core concept is defined and motivated in the sections which follow, these are summarised for convenience in Figure 4.1.

Concept	Synopsis	Meta-model
Asset	Objects to be safeguarded by the system being specified.	• All
Attacker	A human agent fulfilling one or more malicious roles in an environment.	• Environment • Risk • Responsibility
Countermeasure	A security control which mitigates one or more risks.	• Environment • Risk • Responsibility
Domain Property	A descriptive statement about the problem world.	• Goal
Environment	The physical, social, or cultural environs within which a system is situated.	• Environment • Goal
Goal	A prescriptive statement of intent that the system should satisfy through the co-operation of its defined roles in a particular environment.	• Environment • Goal • Risk • Responsibility
Obstacle	A condition representing undesired behaviour that prevents an associated goal from being achieved.	• Environment • Goal
Misuse Case	A scenario that describes how an attacker realises a risk within a given environment.	• Environment • Goal • Risk
Persona	A behavioural specification of archetypical users.	• Environment • Task • Responsibility
Requirement	An optative condition or capability that must be satisfied by the system by a single role.	• Goal • Risk
Response	An action or goal which, if satisfied, leads to the acceptance, transferral, or mitigation of a risk.	• Environment • Risk • Responsibility
Risk	A single event where an attacker carries out a threat and, in the process of doing so, exploits a vulnerability, where both the threat and vulnerability are in the same environment or environments.	• Environment • Risk
Role	A subject that directly or indirectly interacts with the system.	• Responsibility
Scenario	A story about people or a person carrying out an activity.	• Task • Risk
Task	The activities required to achieve a goal.	• Environment • Task • Goal • Responsibility
Threat	A potential cause of an unwanted incident, which may result in harm to a system.	• Environment • Goal • Risk
Vulnerability	A system weakness.	• Environment • Goal • Risk

Figure 4.1: IRIS meta-model core concepts

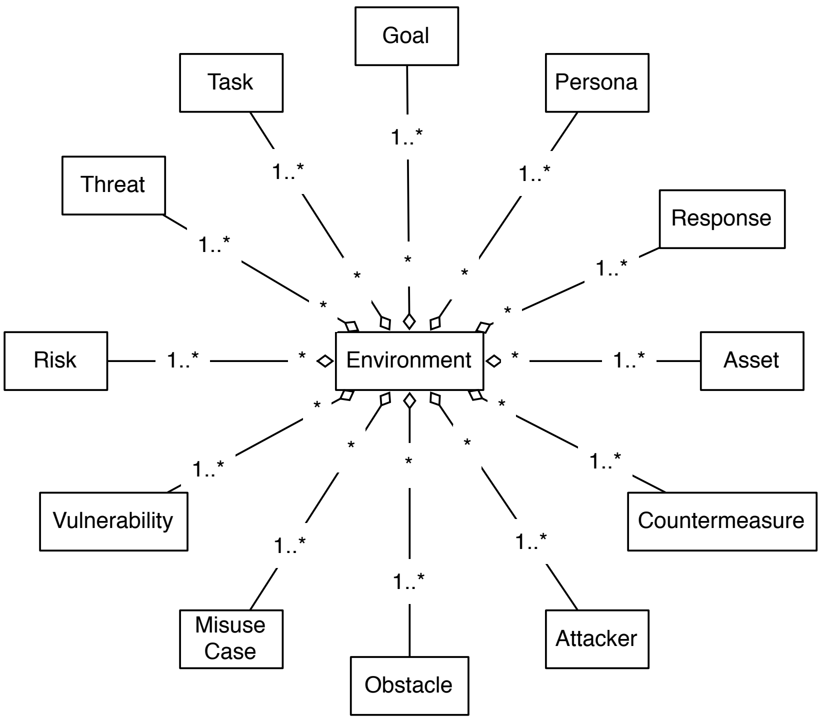


Figure 4.2: Environment meta-model

4.2 Environment meta-model

Systems can be situated in many different contexts, but even though the term *context* is tacitly understood, it is difficult to explicate [73]. For this reason, we have decided to model this concept using the term *Environment*, which we define as the physical, social, or cultural environs within which the system is situated. This term is based on the Oxford English Dictionary definition of Environment, i.e. the object or region surrounding anything, where *anything* is the system being specified, and the *object or region* is the context of interest to the system stakeholders. In Section 7.5.1, a technique for eliciting roughly what contexts would be of stakeholder interest is introduced. This use of the term Environment is implicitly aligned with the term's conceptualisation in ISO 9241 [136] and AEGIS [97].

As Figure 4.2 illustrates, many concepts specified in the IRIS meta-model are situated within one or more environments. Together, IRIS meta-model concepts represent the context of use elements defined in Figure 2.2. Some of these concepts are defined explicitly within an environment; an asset may have different security attributes based on the prevailing environment, and some goals may exist in one environment and not another. Other concepts are implicitly situated within an environment by virtue of their dependencies. For example, a risk may only be defined if the contributing threat and vulnerability exist in the same environment.

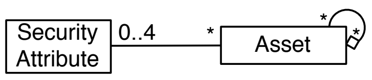


Figure 4.3: Asset meta-model

Some concepts are not situated within an environment. A system is specified by a single set of requirements irrespective of the environments the system must operate in. Similarly, we do not assume a role fulfilled by a human agent will vary by environment. While the role may not vary, the behaviours of the human fulfilling it can; this explains why, in Section 4.4, a persona exists within an environment, but a role does not.

In the NeuroGrid example, three environments were specified. Two of these, *Psychosis* and *Stroke*, were based on the contexts of use associated with the respective clinical researcher communities. The third, *Core Technology* related to the context of use of the NeuroGrid infrastructure development and support team.

4.3 Asset meta-model

Cross-cutting each meta-model view is the concept of Asset, which IRIS defines as objects to be safeguarded by the system being specified; these may also represent components forming part of the system. This definition is not identical to the ISO 27001 definition of Asset described in Section 2.1.1 because we want to indicate the tangibility of objects being safeguarded, rather than intangible concepts like reputation. Even though they are tangible, stakeholders may still value some properties of an asset over others. To reflect this, we associate one or more Security Attributes to each asset. These attributes reflect the security properties stakeholders wish to preserve in the system being specified.

Although we define Information Security as the protection of information from a wide range of threats in order to ensure business continuity, minimise business risk, and maximise return on investments and business opportunities in Section 2.1.1, it can also be defined as *the preservation of Confidentiality, Integrity and Availability of information, and also involve other properties such as Authenticity, Accountability, Non-Repudiation, and Reliability* [138]. With this in mind, we base three security attributes on the former properties stipulated in [138]. These attributes are defined in Figure 4.4.

We have also chosen to include a fourth property: Accountability, which we define as the property that ensures that the actions of an entity may be traced uniquely to the entity [231]. We include this as a property over others such as Authentication and Non-Repudiation for two reasons. First, given the human dimension of security reported in Section 2.1.1, accountability enables us to glean whether or not stakeholders feel that keeping track of asset accountability is important. Second, from a security

Attribute	ISO/IEC 27001 Definition
Confidentiality	The property that information is not made available or disclosed to unauthorised individuals, entities, or processes.
Integrity	The property of safeguarding the accuracy or completeness of assets.
Availability	The property of being accessible and usable upon demand by an authorised entity.

Figure 4.4: ISO 27001 definitions for Confidentiality, Integrity, and Availability

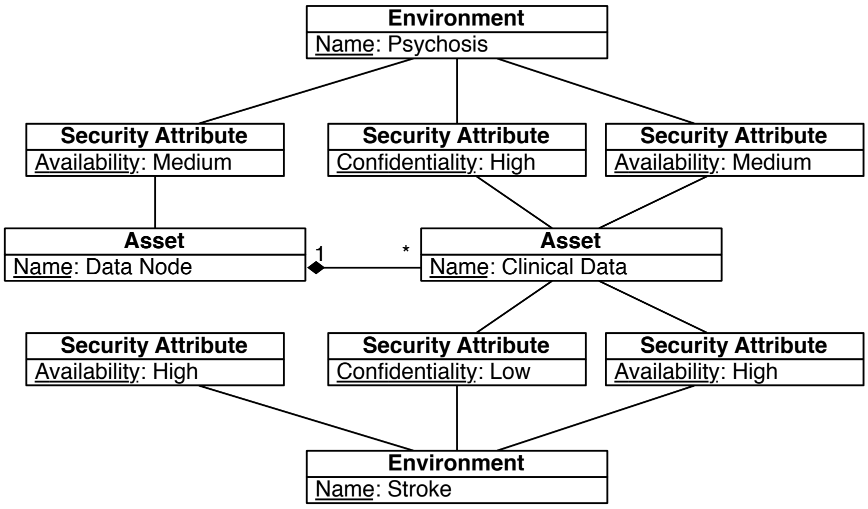


Figure 4.5: Comparative asset values across environments

perspective, accountability is virtuous; thinking about this value encourages stakeholders to think about what it means to be answerable for a particular asset [179].

Figure 4.5 shows a fragment of the asset model for the Psychosis and Stroke environments. In each environment, a data node, which corresponds to a network server, may be associated with zero or more items of clinical data. Although the assets and associations are identical for each environment, the security properties vary. In the Psychosis environment, clinical data is both highly sensitive and partially anonymised; data owners are prepared to sacrifice availability of the NeuroGrid service to safeguard this. In the Stroke environment, however, clinical data is highly anonymised, leading to a lower confidentiality attribute. The availability property associated with the data node asset in this environment is also comparatively higher due to the importance of NeuroGrid's accessibility during clinical trials, as opposed to ad-hoc research in the Psychosis environment.

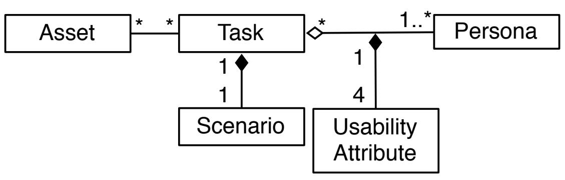


Figure 4.6: Task meta-model

4.4 Task meta-model

The Task meta-model view, illustrated in Figure 4.6, captures elements describing how people carry out work in the system being specified. Rather than a description of a system operative like *user* and *administrator*, IRIS characterises people using personas. We define these as behavioural specifications of archetypical users, as indicated in Section 2.2.2.3. We treat a persona as a descriptive model of how indicative users behave, think, what they wish to accomplish, and why [61]. Personas sensitise stakeholders to the context of use and, through the interplay between personas and their tasks, help identify assets, threats, and vulnerabilities which would otherwise be missed by thinking about each in isolation. Personas are developed from data about users and their contexts; this means few assumptions need to be made about how people fulfil different roles, thereby reducing the risk of ambiguity about operatives being introduced into the analysis process.

As well as an informal objective the persona wants to meet, a task is composed of a textual scenario describing how the persona carries out some work associated with the system. An IRIS Scenario is defined as a story about people or a person carrying out an activity. This definition is based on [204], where the people may be personas we wish to design the system for, although, as Section 4.6 highlights, these people may also be attackers that we want to defend the system against.

The IRIS definition of Task, the activities required to achieve a goal, is based on [136]; however, the goal is assumed to be an IRIS goal rather than a personal or occupational goal associated with a persona. The significance of this will become apparent in Section 4.5. Each task aggregates one or more personas and, for each associated persona, Usability Attributes are specified. These attributes are defined from the perspective of the persona, and describe how well the work in the task meets implicit usability goals of efficiency, effectiveness, and satisfaction. These attributes are elicited by assigning categories relating to task duration, task frequency, task demand, and the task's support for the persona's intentions. When describing tasks, it may become apparent that certain assets use or constrain them. To reflect this, the meta-model allows assets to be associated with tasks, enabling traceability between tasks and models where these assets are present.

Figure 4.7 describes how these meta-model concepts and relationships structure the information about how a clinical researcher persona, Claire, uploads data to NeuroGrid. The examples show the asset

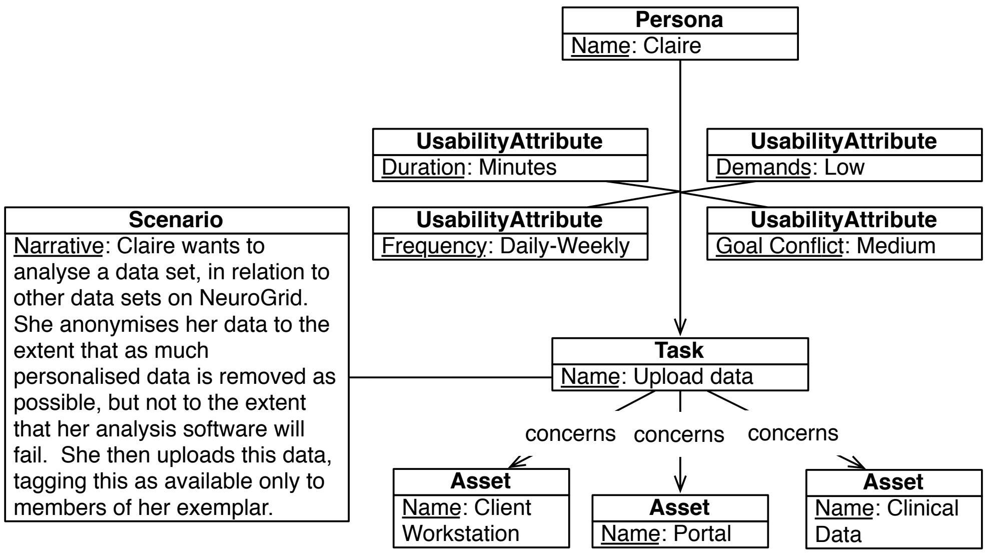

Persona
Name: Claire
UsabilityAttribute
Duration: Minutes
UsabilityAttribute
Demands: Low
UsabilityAttribute
Frequency: Daily-Weekly
UsabilityAttribute
Goal Conflict: Medium
Scenario
Narrative: Claire wants to analyse a data set, in relation to other data sets on NeuroGrid. She anonymises her data to the extent that as much personalised data is removed as possible, but not to the extent that her analysis software will fail. She then uploads this data, tagging this as available only to members of her exemplar.
Task
Name: Upload data
concerns
concerns
concerns
Asset
Name: Client Workstation
Asset
Name: Portal
Asset
Name: Clinical Data

Figure 4.7: Task meta-model example

concerns associated with this task, the narrative of the task itself, and the usability attributes describing how long it takes Claire to carry out this task, how frequently she carries it out, how demanding she finds it, and how much it obstructs her personal or work goals.

In Section 6.4.1, we return to this particular example to illustrate how the usability attributes described form the basis of automating usability analysis.

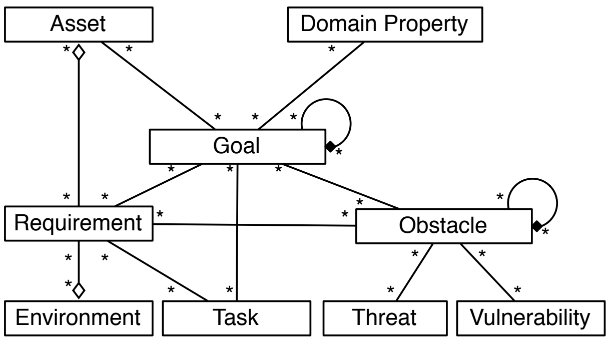


Figure 4.8: Goal meta-model

4.5 Goal meta-model

Many concepts in the Goal meta-model view, illustrated in Figure 4.8, are based on the KAOS method introduced in Section 5.4.5. KAOS was chosen over alternative goal-oriented notations for two reasons. First, unlike agent-oriented approaches described in Section 2.3.2.2, KAOS defines a goal as a prescriptive statement of system intent. Because goal and requirement analysis is already carried out in the context of personas and tasks, we use goals as a vehicle for refining requirements, rather than understanding the intent of actors. Second, as van Lamsweerde has recently reported [255], the KAOS modelling notation is compliant with UML and, by extension, compatible with modelling notations commonly used in Software Engineering.

In IRIS, our definition of Goal is broadly in line with that prescribed by KAOS: a prescriptive statement of intent that the system should satisfy through the co-operation of its roles in a given environment. This definition of a goal appeals to the objectivity of the specification being developed as opposed to the subjectivity of analysis which informs usability attributes (Section 4.4) and security attributes (Section 4.6). The definition differs from van Lamsweerde's definition in that we use the term *roles* rather than agents; this is to reinforce the fact that these tend to be human, rather than system, agents. The definition also binds a goal to a particular environment, because goals and their attributes might be defined in slightly different ways for each; they also may not be present in all specified environments.

In KAOS, requirements are refined goals under the responsibility of a single agent. In IRIS, a requirement is defined as an optative condition or capability that must be satisfied by the system by a single role. This definition is based on the IEEE definition described in Section 2.3, as well as a refined version of the van Lamsweerde's goal definition [255]. As the definition suggests, IRIS chooses to make the distinction between goal and requirement explicit by treating the latter as an independent concept. This leads to a number of benefits. First, requirements may arise during analysis independent of any goal refinement or task discussion. Second, although not recommended, requirements and risk analysis can be divorced from goal modelling; this may be useful if neither analysts nor developers have any knowledge of goal-oriented techniques, or this analysis has been carried out by a different team and is being judiciously re-used.

Domain Properties are descriptive statements about the problem world; this definition is inline with van Lansweerde's definition for the term in KAOS [255]. In IRIS, we use these to capture assumptions satisfied by the system, or properties which must hold to achieve requirements, but are outside of the scope of the system being specified. Specifications derived from IRIS are based on a fixed scope, but goal modelling may lead to the elicitation of important out-of-scope requirements. If these requirements are not specified elsewhere, assuming these requirements must hold by defining them as domain properties is useful for ensuring they are not forgotten.

Assets or environments may be associated with zero or more requirements. When associated with

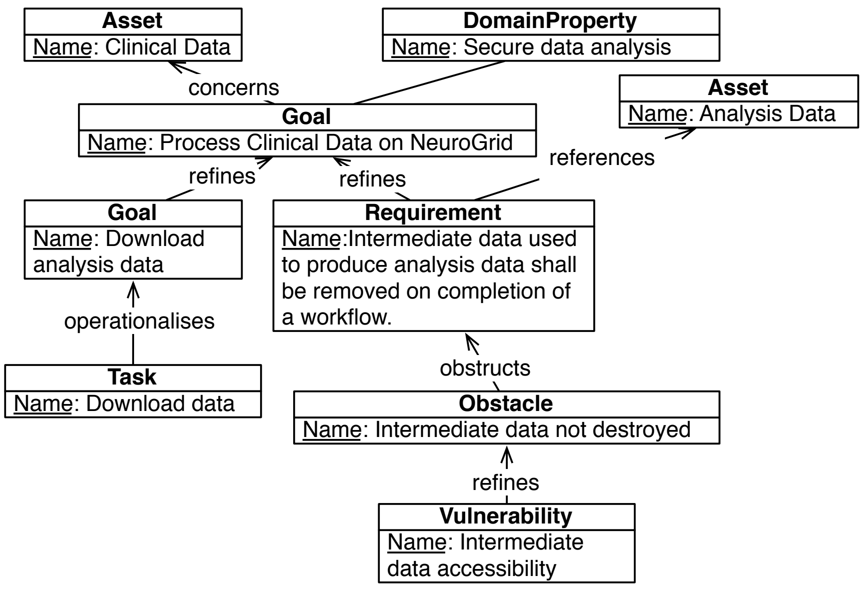

Asset
Name: Clinical Data
DomainProperty
Name: Secure data analysis
concerns
Asset
Name: Analysis Data
Goal
Name: Process Clinical Data on NeuroGrid
references
refines
refines
Goal
Name: Download analysis data
Requirement
Name:Intermediate data used to produce analysis data shall be removed on completion of a workflow.
operationalises
obstructs
Task
Name: Download data
Obstacle
Name: Intermediate data not destroyed
refines
Vulnerability
Name: Intermediate data accessibility

Figure 4.9: Goal model example

assets, the requirements reference or constrain an asset; this is typically the case when a security requirement acts as a constraint. This is similar to the approach taken by Problem Frames (Section 2.3.1), where requirements reference certain phenomena in problem diagrams; this reference indicates that a requirement references or constrains the related domain in the context being modelled. In certain cases, however, requirements may be elicited which do not relate to assets. These may be requirements not refined from a goal, non-functional requirements with a broad system impact, or a requirement implying functionality yet to be conceptualised. In such cases, associating a requirement with the environment stimulating its elicitation may be useful if the requirement needs to be categorised for some reason, e.g. a requirement specification is generated and requirements need to be grouped based on the environments they relate to. By defining these associations as aggregations, we indicate that tool-support implementing this meta-model should not remove these requirements if the associated asset or environment is removed; these requirements may be associated with other assets or environments.

In addition to being refined to sub-goals, requirements, and operationalised as tasks, KAOS goals may conflict with obstacles as indicated in Section 2.3.2.1. Like domain properties, the IRIS definition of Obstacle is identical to that used by KAOS in Section 2.3.2.1. By refining obstacles, candidate threats and vulnerabilities may be defined. We have chosen to model IRIS goal obstruction with obstacles rather than anti-goals introduced in Section 2.3.2.1. While anti-goals arise exclusively from malicious intent, obstacles may arise from accidental error as well.

Figure 4.9 illustrates how these different concepts come together. The scope of analysis for the NeuroGrid specification exemplar is the transfer of data; as such, the high-level goal of processing clinical

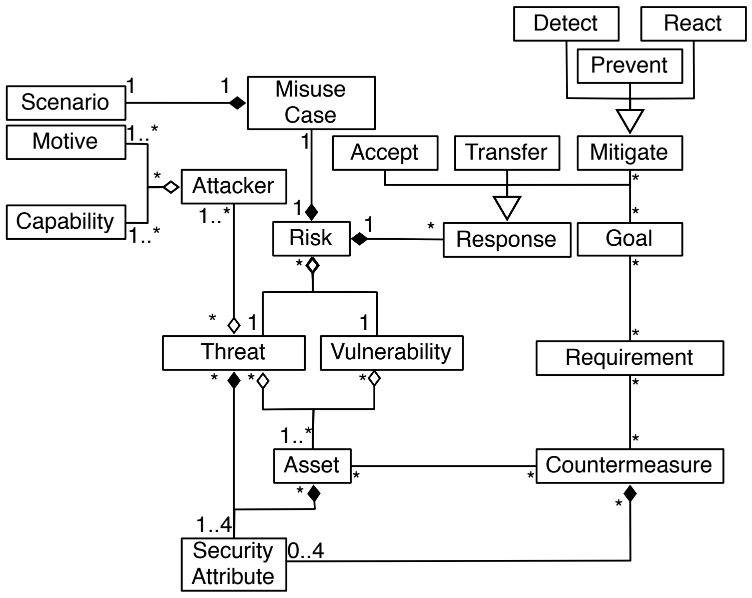


Figure 4.10: Risk meta-model

data assumes that the process of analysing clinical data on data nodes is secure. This goal also *concerns* clinical data, which suggests a traceability association between the goal and this asset.

A number of other goals and requirements need to be satisfied in order for this goal to be satisfied. One of these is the *Download Analysis Data* goal. In this example, we indicate that the *Download data* task illustrates the satisfaction of this goal. Because this is a system, rather than a persona, goal, the task scenario is written in such a way that a reader can understand precisely how the persona carries out the task to achieve this. If achieving this goal conflicts with the persona's own personal goals then these are reflected in both the scenario and the usability attributes associated with the task.

Another requirement following from the high-level goal indicates that any intermediate data created during the process needs to be destroyed when no longer needed. This requirement references the analysis data asset by stipulating a constraint which needs to hold in its construction. An obstacle to this goal might be this data not being destroyed, leading to a vulnerability because sensitive information might be encapsulated within this transient data.

4.6 Risk meta-model

The Risk meta-model view, illustrated in Figure 4.10, models the elements contributing to the definition of risks, and responses and mitigations to them.

In IRIS, risks arise when a number of instantiated concepts come together within one or more environments.

The first of these is a Vulnerability, which we define as a system weakness. This definition is based on Mayer's definition of the term [165], but we restrict its usage to system, rather than organisational, weaknesses. This is because an IRIS scope of analysis is based on a specific system scope and not the larger organisational context it is situated in.

The second of these is an Attacker: a human agent fulfilling one or more malicious roles in an environment. This definition is based on Mayer's definition of an Attacker, i.e. a malicious agent in an environment [165]. In the IRIS definition, we have chosen to explicate that this agent is human, and the attacker is fulfilling malicious roles, rather than indicating the person is intrinsically malicious.

For a risk to be realised, an attacker needs to carry out a Threat: a potential cause of an unwanted incident, which may result in harm to a system. This definition is based on the ISO 27001 definition of the term [138], however the IRIS definition alludes to harm to a system or an organisation. This definition of threat differs from Mayer's [165]: that which exploits a vulnerability and causes an unwanted incident. Mayer's use of the term assumes that threats will always exploit a vulnerability, however threats in IRIS are independent of exploitable vulnerabilities. An IRIS threat may be benign until circumstances arise such that a vulnerability can be exploited.

Based on these definitions, IRIS defines a Risk as a single event where an attacker carries out a threat and, in the process of doing so, exploits a vulnerability, where both the threat and the vulnerability are in the same environment or environments. This definition is similar, but not identical to, Mayer's definition of a threat.

Both threats and vulnerabilities are concerned with assets; threats target assets, and vulnerabilities expose their weaknesses. Threats arise from the motives and capabilities defined for the attackers behind them. Many of these possible motives and capabilities are known from reports in the literature and media. The attackers behind a threat target assets with respect to the security values of interest to them. Therefore, when a threat is defined, security attributes are associated with it depending on the assets the attacker wants to exploit, and how much they want to exploit them by.

For each defined risk, a Misuse Case must also be specified; we define this as a scenario that describes how an attacker realises a risk within a given environment. This definition differs from Sindre and Opdahl's definition in several ways. First, we describe the risk's realisation as a narrative rather than a discrete set of steps. Second, variants are not included; these should be reflected using an alternative misuse case, or a different scenario for another environment associated with the misuse case's risk. Third, we make explicit the notion that misuse cases should validate rather than elicit risks. While the traditional view of misuse cases is that they threaten use cases, this view contests the idea that threats target assets, and not just contexts of use in general. By narrating a risk's impact using a scenario, misuse cases not only place the risk in a human context, they also sanity check the analysis contributing to the definition of a risk, justifying its threat and vulnerability, the related likelihood and severity values, and the security attributes for these assets.

Risks may be treated in different ways. This treatment is defined using the IRIS concept of a Response: an action or goal which, if satisfied, leads to the acceptance, transferral, or mitigation of a risk. This definition is analogous to *Risk Reduction* in ISO 27005 [142], i.e. actions taken to lessen the probability, negative consequences, or both, associated with a risk. Although we may choose to accept the consequence of its impact, or transfer if the responsibility for dealing with it is out of scope, we also choose to mitigate the risk if it has a bearing on the system specification. Consequently, although the former two options are actions, the latter is a goal. The strategy for mitigating a risk may involve preventing or avoiding it, detecting it, or reacting to it. Choosing to mitigate a response is synonymous with intentionally specifying that the system shall manage the risk as part of its design. Consequently, we can associate goals with mitigating responses. As Section 4.5 indicates, these goals can be analysed and refined to requirements mitigating this goal.

Countermeasures can be defined to meet these aforementioned requirements; these are defined as security controls which mitigate one or more risks. This definition is similar to the ISO 27002 definition of a countermeasure, i.e. a means of managing risk. The IRIS definition is, however, more situated to the meta-model in that we explicitly define it to be a control, which might be technical, procedural, or social. Moreover, a countermeasure mitigates a risk rather than passively managing it. Should we choose to incorporate them in the specification, assets can be associated with these countermeasures. If countermeasures target threats, and these countermeasures are considered effective at reducing the value of the security attributes an attacker might hold about the threatened assets, these can also be defined at the countermeasure level.

The example in Figure 4.11 models the risk of unauthorised access to user certificates. The antagonist in this example is Carol, a journalist with a specific set of motivations and capabilities. Carol intends to carry out a social engineering attack to obtain access to a user certificate; her stake in this threat is in compromising the confidentiality of this asset. A risk is realised when Carol carries out this threat and exploits the propensity of users to share certificates with each other. A scenario describing how this risk is realised is described in the misuse case associated with this risk. In this example, we have decided to prevent the occurrence of this risk, which eventually leads to the specification of a goal to this effect. After a certain amount of goal refinement, a requirement stating that user certificates shall only be usable at designated network addresses is elicited. A countermeasure implementing this requirement is specified; this safeguards the user certificate by targeting the certificate-sharing vulnerability by tying the use of user certificates to specific locations.

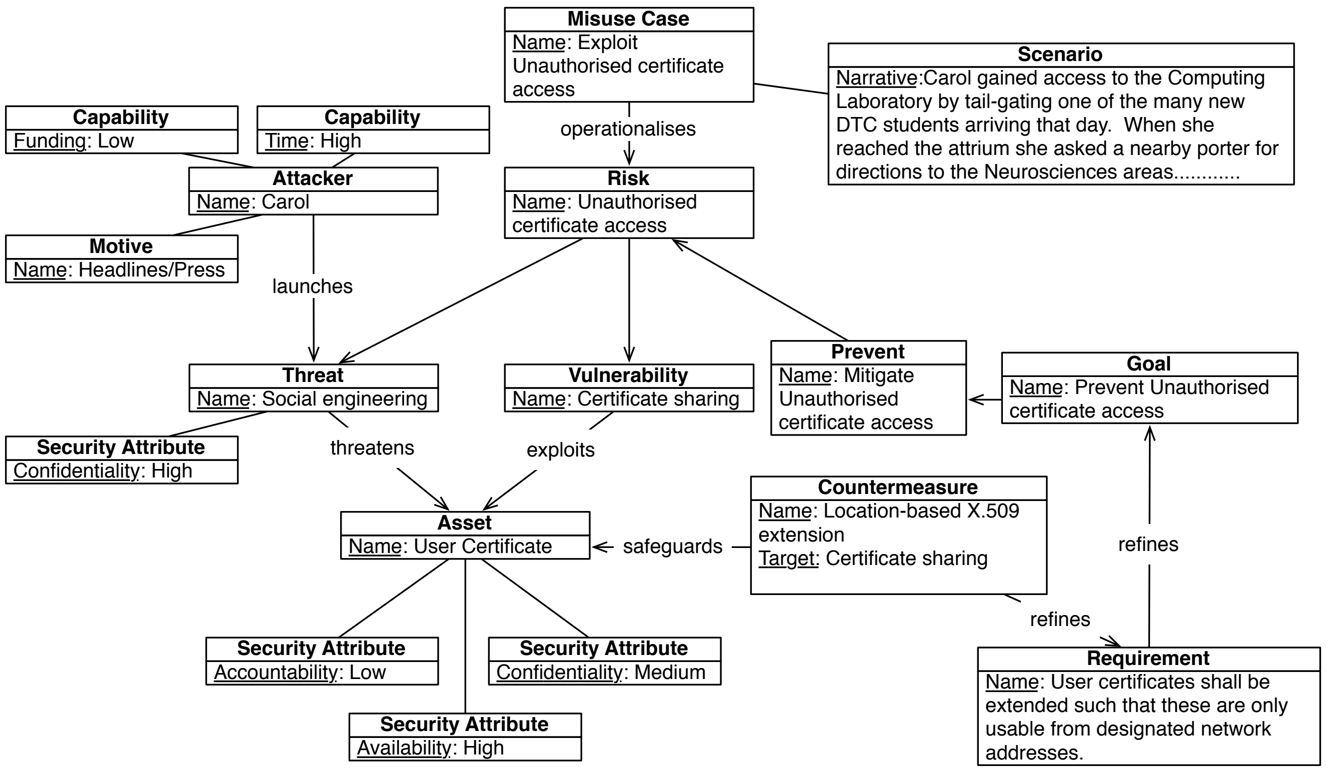

Misuse Case
Name: Exploit Unauthorised certificate access
Scenario
Narrative:Carol gained access to the Computing Laboratory by tail-gating one of the many new DTC students arriving that day. When she reached the attrium she asked a nearby porter for directions to the Neurosciences areas............
Capability
Funding: Low
Capability
Time: High
operationalises
Attacker
Name: Carol
Risk
Name: Unauthorised certificate access
Motive
Name: Headlines/Press
launches
Threat
Name: Social engineering
Vulnerability
Name: Certificate sharing
Prevent
Name: Mitigate Unauthorised certificate access
Goal
Name: Prevent Unauthorised certificate access
Security Attribute
Confidentiality: High
threatens
exploits
Countermeasure
Name: Location-based X.509 extension
Target: Certificate sharing
Asset
Name: User Certificate
safeguards
refines
refines
Security Attribute
Accountability: Low
Security Attribute
Confidentiality: Medium
Requirement
Name: User certificates shall be extended such that these are only usable from designated network addresses.
Security Attribute
Availability: High

Figure 4.11: Risk model example

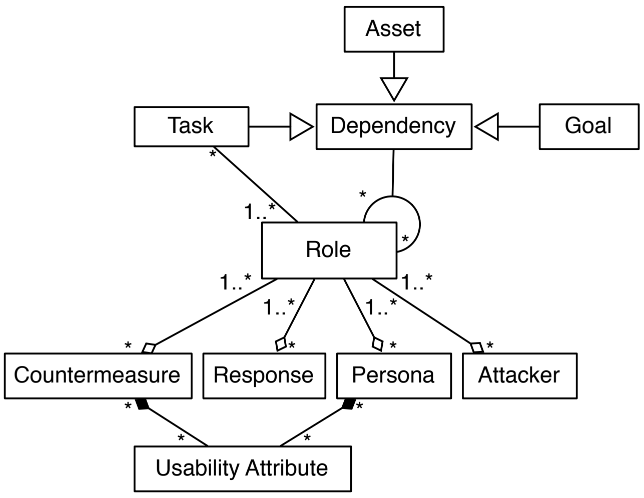


Figure 4.12: Responsibility meta-model

4.7 Responsibility meta-model

Roles are subjects directly or indirectly interacting with the system. The definition is based on the definition for a UML Actor [208].

Attackers and personas are associated with one or more roles. Roles may also become responsible for risk responses transferred to them, or countermeasures they are required to maintain. These responsibility relationships are captured by the Responsibility view of the IRIS meta-model, illustrated in Figure 4.12.

IRIS distinguishes between representations of human personas, attackers, and roles for two reasons. First, by allocating responsibilities to roles, we can identify personas that may become overloaded with roles, or roles which may be dispersed among several personas; these cases can lead to security responsibilities becoming neglected. Second, attackers and personas may share certain roles, allowing us to consider the evolution of a lawful stakeholder to an inside attacker.

Because countermeasures are assigned to roles, roles are associated with personas, and personas participate in tasks, the meta-model facilitates exploring the usability impact of countermeasure design decisions. This is possible if a countermeasure assigned to a role within a given environment is shared by a persona participating in one or more tasks in the same environment. Where this occurs, usability attributes may be associated with the persona-task pairing to indicate how much the countermeasure helps or hinders the persona in the associated task. This is described in more detail in Section 6.4.1.

Although goals define how the intent of a system is refined to a requirements specification, it is also important to understand how these goals laterally relate to other concepts; this knowledge is captured using dependency relationships. Ternary associations describe how one role (the depender), relates to another (the dependee), for a task, goal, or asset (the dependum). The dependency relationships in IRIS are based on the Strategic Dependency links used by i*, described in Section 2.3.2.2; van Lamsweerde [255] has described how these relationships can also be used in KAOS to supplement agent responsibility modelling.

Figure 4.13 shows how, when the data provider role is associated with a countermeasure, usability information about the impact of the countermeasure design to Claire — who fulfils the data provider role — can be recorded. The example also illustrates how data producers are responsible for access control decisions, but rely on a certificate authority for this responsibility to be discharged. This is necessary because, in this specification exemplar, certificate authorities modify access control policies on behalf of data providers because they are too cumbersome for end-users.

4.8 Conclusion

This chapter presented the IRIS meta-model. We described how the meta-model is centred around the concept of Environment before detailing the component parts of this meta-model. We illustrated each

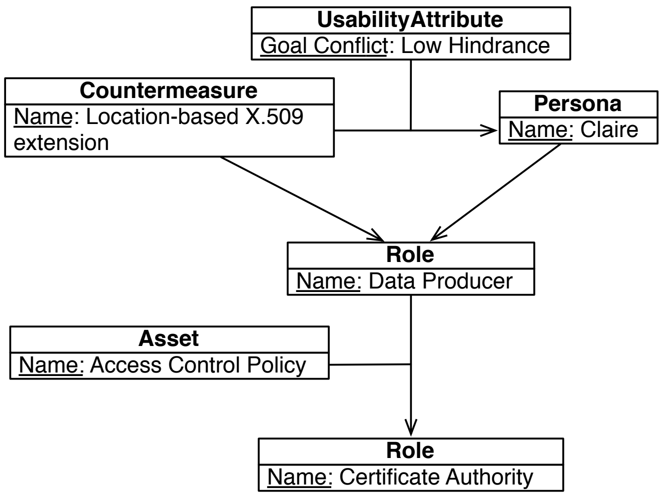

UsabilityAttribute
Goal Conflict: Low Hindrance
Countermeasure
Name: Location-based X.509 extension
Persona
Name: Claire
Role
Name: Data Producer
Asset
Name: Access Control Policy
Role
Name: Certificate Authority

Figure 4.13: Responsibility model example

part of the meta-model using examples from the NeuroGrid exemplar.

In the following chapters, we describe how a process framework can build upon this meta-model, and how tool-support can employ this meta-model to structure data elicited in support of IRIS processes.

Chapter 5

A Process Framework for Usable Security

This chapter presents a process framework for specifying usable and secure systems. Building on the meta-model described in Chapter 4, we describe the different *perspectives* of IRIS, and their make up. Finally, we propose a number of exemplar techniques for each perspective.

5.1 Overview

The meta-model in Chapter 4 facilitates the specification of requirements for usable and secure systems by stipulating concepts needing to be elicited. The meta-model is, however, agnostic about how these concepts are elicited; this makes it necessary to provide guidance on how these concepts should be elicited and specified. To this end, we have broken up the meta-model into three intersecting groups as illustrated by Figure 5.1, as follows:

- Usability
- Requirements
- Security

We call these groups *perspectives* because each views the specification process subjectively through a lens coloured by its related concepts. Each perspective corresponds to different activities carried out as part of a design effort. The premise underpinning this framework is that the specification development process is hermeneutic rather than iterative. Nuseibeh alludes to this in his twin-peaks model [183], which talks about the dialogue between requirements and architectural activities. We assert that when

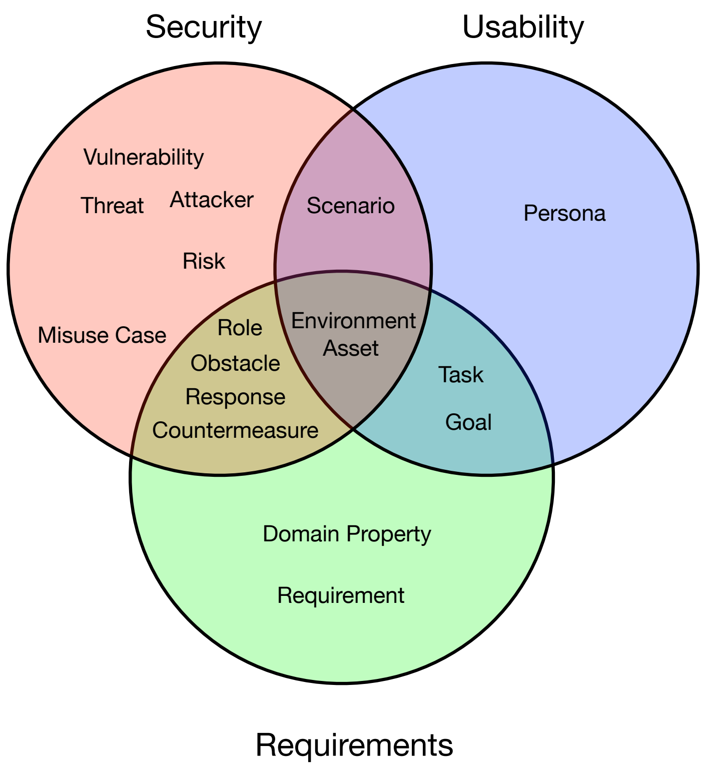


Figure 5.1: IRIS perspective concepts

specifying requirements, insights can occur at any time. While guidance is needed in a design process, the techniques adopted need to be agile enough to switch from one perspective to another should new insights dictate.

5.2 Perspectives

Each perspective views the design process in a different way, and techniques situated within them share certain characteristics.

The Usability perspective views the design process as a means of understanding how a system can be situated in its contexts of use. Consequently, the techniques situated by this perspective aim to model this understanding. The techniques associated with this perspective are often described as *user-centered*, but this is a slight misnomer. Instead, these techniques centre on one or more concepts within a context of use. For example, personas are centred on users and their goals, tasks are centred on user goals and how they achieve them, while scenarios are centred on the intrinsic detail of human activities within different environments. In each case, the techniques focus on specific concepts, but not to the exclusion of other elements of the context of use. Techniques within this perspective are also sensitive to human values. IRIS does not state what human *values* are conceptually, nor does it explicitly describe how these are portrayed from a stakeholder perspective. Instead, values are unpacked and elucidated by the usability techniques that elicit and analyse empirical data.

The Requirement perspective views the design process as a means of specifying the system being built. As such, the techniques situated by this perspective aim either to objectively specify the system being designed, or elicit models of how inter-related concepts situated in their contexts contribute to an objective specification. The techniques associated with this perspective can be described as *requirement-centered.* The ultimate end of techniques situated in this perspective are prescriptive, objective, unambiguous, and bounded statements of system intent. For example, goal-oriented approaches are based on progressive refinement of goals to requirements, while the primary objective of rich picture diagrams, which we introduce in Section 7.5.1, is delimiting the specification space, rather than exploring it.

The Security perspective views the design process as a means of understanding how a system can be securely designed; specifically, how the design can make a system more secure. The techniques associated with this perspective can be described as *risk-centered*, because these aim either to understand what these are, or what design decisions are necessary to adequately respond to them. While what might be considered as “adequate” is subjective, it is, nonetheless, rational to the involved stakeholders. Consequently, techniques situated in this perspective not only discover what these risks are, but how they contribute to design in general.

5.3 Converging concepts

As Figure 5.1 illustrates, these perspectives are not mutually exclusive. Our review of the literature, particularly Section 2.2.3, concluded that certain concepts converge as design communities associated with each of these perspectives begin to understand the values of other design communities.

The concepts of Environment and Asset are shared by all perspectives, but for different reasons. In the Usability and Security perspective, these concepts are exploratory, while these reference or constraint concepts in the Requirements perspective.

Goals and tasks are conceptually shared between the Usability and Requirements perspectives; this is because user needs embodied by task descriptions in specific contexts eventually need to be reified by objective goals and requirements holding for all contexts of use. Similarly, roles, obstacles, responses, and countermeasures are conceptual interfaces between Security and Requirements because they are used by techniques from both perspectives. Only the Scenario concept is situated between the Security and Usability perspectives because, in both perspectives, these are used for exploring contexts of use rather than specifying elements within them.

5.4 Framework techniques

We now consider techniques which elicit the concepts from within these three perspectives. We have selected several candidate techniques, as summarised in Section 5.3. A *technique* can be defined as a particular way of carrying out a procedure or task, or a skilful or efficient means of achieving a purpose [4]. As such, techniques can be construed as processes in their own right, albeit ones which require knowledge and experience in their use.

The framework is instantiated by devising an IRIS process: an ordered collection of techniques, informed by the context within which it is applied. This developmental context may be shaped by similar factors to those reported in Section 2.4.2 for SQUARE. Collectively, the techniques in a process should elicit all the concepts stipulated by the IRIS meta-model. There are no specific rules about what class of constraints should follow others, nor what techniques can run concurrently with others. This is because the application of a technique from one perspective may lead to insights informing the analysis carried out using a technique in another. In general, however, the following principles apply:

- At the start of a process, one or more Requirements perspective techniques should be used to establish the scope of the system. Subsequent techniques assume that a context of design has been both agreed and specified.
- As the end product of a process is a specification, then a technique incorporating the validation of requirements is singularly or concurrently the last technique applied.

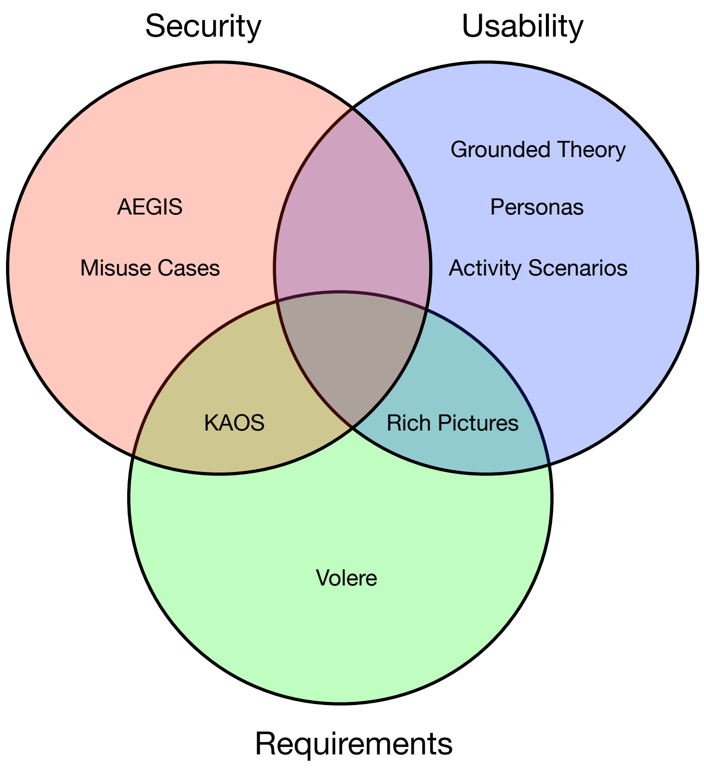


Figure 5.2: Techniques associated with IRIS perspectives

Technique	Perspective	Input	Settings	Elicited Concepts	Output
Personas	• Usability	• Rich Picture	• Fieldwork • Analyst • Workshop	• Personas	• Persona specifications • empirical data
Activity Scenarios	• Usability	• Empirical data • Goals	• Workshop • Analyst	• Tasks • Scenarios • Usability Attributes	• Tasks
Grounded Theory	• Usability	• Empirical data	• Workshop • Analyst		• Qualitative models
Rich Pictures	• Requirements • Usability	• Empirical data	• Workshop • Analyst	• Roles • Environments	• Rich picture diagrams • Goals
KAOS	• Requirements • Security	• Empirical data • Goals	• Workshop • Analyst	• Goals • Obstacles • Domain Properties • Dependencies	• Goal Model
Volere	• Requirements	• Empirical data • Requirements	• Workshop • Analyst	• Requirements	• Requirements specification
AEGIS	• Security	• Empirical data	• Workshop • Analyst	• Assets • Security Attributes • Vulnerabilities • Attackers • Threats • Risks • Responses • Countermeasures	• Risk Analysis Model
Misuse Cases	• Security	• Risks	• Workshop	• Misuse Cases • Scenarios	• Risk Analysis Model • Task Model

Figure 5.3: Techniques overview

We describe these techniques in more detail in the following sections. For each technique, we present a brief overview of its key features, the rationale for using it, followed by interpretations and variations in its application as part of IRIS.

5.4.1 Grounded theory

Grounded Theory is a qualitative data analysis technique for generating theory from observed real-world phenomena [63]. These theories, and the sense-making activities associated with carrying them out, form the raw material that User-Centered Design artifacts can build upon. Although not traditionally construed as a design technique per se, it has been used for theory-building in security and privacy research. Adams [6] used Grounded Theory as a governing method for her thesis on user perceptions of privacy in multimedia communications. The method was used for inducing a model of user perceptions of privacy from empirical data. It was also used to structure the dissertation by highlighting how each chapter contributed to this model. Fléchais [97] used Grounded Theory to induce and refine a model of the factors affecting the design of secure systems, based on empirical data gathered from several different case studies.

Strauss and Corbin's flavour of Grounded Theory [63] consists of three stages of analysis: Open Coding, Axial Coding, and Selective Coding.

To help make initial sense of the data, the analytic tool of *sensitising questions* is used during open coding. Strauss and Corbin [63] define sensitising questions as questions that help the researcher to see process, variation, and so on, and to make connections between concepts. Axial coding involves crosscutting, or relating, concepts to each other; this activity goes hand-in-hand with, rather than separately following, open coding [63]. During axial coding, the levels of abstraction are frequently interleaved between conceptual and data levels. Causes, conditions, contexts, and consequences are identified, mapped using semantic relationships, and structured into categories. This activity leads to new insights into the source material, and guides subsequent coding activities. During selective coding, emergent categories are organised around a central, core category according to specific criteria, such as ensuring the central category is abstract, it appears frequently in the data, it is logical and consistent with the data, and the concept grows with depth and explanatory power as other categories are related to it.

Although we propose Grounded Theory be used for theory development in IRIS, induced theories are not developed for dissemination beyond the design team and system stakeholders. Instead, as indicated in this section's opening, the sense-making associated with applying Grounded Theory is primarily used to support the elicitation and specification of other IRIS concepts. Because of the usefulness of tracing such artifacts back to their data sources, coupled with the quantity of data that may be elicited during a design intervention, Grounded Theory applications in IRIS should be tool-supported. Fortunately, Grounded

Theory is well-supported by Computer Aided Qualitative Data Analysis (CAQDAS) packages. In the interventions where Grounded Theory is applied in this dissertation, the ATLAS.ti CAQDAS tool [174] is used.

5.4.2 Personas

Personas were introduced in Chapter 2 as an archetypical specification of indicative user behaviour. We chose personas as a user proxy for IRIS because of their grounded representations of users. Although personas are viewed as a narrative structured by different behavioural variable types, their authorship is the least time-consuming element of their construction. In IRIS, data is explicitly collected and analysed to identify clusters of behaviour. This data is collected from representative users or related stakeholders, ideally within contexts the system needs to be situated in.

In IRIS, a photo or a picture is associated with each persona; this helps stakeholders understand that personas describe a single person [196]. Environment specific attributes about the persona are also specified; these include the roles fulfilled by the persona, and whether the persona is a direct or indirect user. Direct users are the users of the system being specified; indirect users are indirectly impacted by the system. The motivation for considering direct and indirect users comes from Friedman et al.'s consideration of direct and indirect stakeholders in Value-Sensitive Design [104]; this distinction allows the values of stakeholders who unknowingly participate in system supported activities to inform its design.

A further environmental attribute is a short narrative describing aspects of a persona relevant to a specific environment.

5.4.3 Activity scenarios

Activity scenarios centre around activities performed by users, rather than around the users themselves. In Scenario-Based Usability Engineering (SBUE) [203], these are preceded by Problem Scenarios, which describe how hypothetical stakeholders tackle current practice; these scenarios may be based on empirical data or assumptions. Notable positive or negative consequences to stakeholders in problem scenarios are marked as *Claims*, which suggest possible design criteria rather than specific requirements. Activity scenarios are written to explore claims arising from problem scenarios. Claims may also be identified from activity scenarios, although these are used to describe the pros and cons of different approaches [204].

In IRIS, activity scenarios are realised as tasks. Both can be applied in individual and participatory settings, and both are used to explore the impact of possible design decisions. However, while the scenario component of activity scenarios, and its contribution to the larger design process remains unchanged, the IRIS implementation of these differ in a number of respects.

First, claims analysis is not explicitly supported because the meta-model associations make these redundant. Different specification possibilities can be modelled using KAOS OR-goals; these can be operationalised as different tasks. A further reason for not supporting claims analysis is problems arising from their subjective nature. In participatory settings, a *pro* to one participant may be a *con* to another. The value of claims analysis in SBUE is leading debate about the claim, but in IRIS it is in presenting the claim and letting stakeholders make their own interpretations about what this means. With this in mind, Section 8.3.2 will present an example of how scenarios can be used to explore the impact of a claim.

Second, rather than treating users as hypothetical stakeholders, IRIS makes explicit assumptions about the use of personas, and the usability attributes of a persona (or personas) involvement with tasks. Despite the implicit rationale links between activity scenario claims and requirements, personas and usability attributes act only in an exploratory role.

5.4.4 Rich pictures

Rich pictures are a diagrammatic way of representing the main concepts associated with a problem, and the relationships between them. Rich pictures are not based on any particular syntax, although the meaning of the imagery drawn, be they boxes or sketches, are meaningful to both the stakeholders and the situation being described. Although the idea of representing problems using pictures is timeless, the rich pictures concept was introduced by Checkland and Scholes [47] as part of their Soft Systems Methodology. Checkland and Scholes propose the use of rich pictures to help obtain a *Root Definition*: a succinct definition of a system under investigation or being built. As such, rich pictures augment what is known as a CATWOE analysis. This analysis considers designing as a transformation process (T) with a world-view (W) that makes this meaningful. Actors (A) are responsible for the transformation itself, which may be constrained by the environment (E) in some way. This transformation may or may not benefit customers (C), some of whom may be members of a group of owners (O) with the power to stop the transformation process.

Rich pictures have been described by some Requirements Engineering proponents [13] as a *soft-system* approach due to its view of the world as a complex, human activity system. This differs from the *hard* view of a system context, which models the system as a central icon, and deals only with the events shared by interfacing entities, be these human or machine; this approach to context modelling is adopted by [143] and [200].

In IRIS, the Rich Pictures technique is shared between the Usability and Requirements perspective. From a Usability perspective, the technique is one which explores the contextual factors impacting a system. From a Requirements perspective, the resulting model is an artifact which, to stakeholders, encapsulates a system's scope, identifies key system roles, and alludes to environments within which the system will be situated in.

Despite the close relationship between a rich picture and a root definition, this latter concept has not been incorporated into IRIS for two reasons. First, contributing artifacts from a rich picture used in IRIS are prescriptive high-level system goals, subject to refinement, rather than a joint descriptive and prescriptive statement about the system. Second, when a system is being scoped we are interested in *all* roles at play in a system, rather than differentiating between different classes of stakeholder. Understanding the politics associated with social classes may be useful from a Usability perspective, but not from a Requirements perspective unless these admit of requirements which need to be explicitly represented in a system specification. In such cases, the rationale behind these requirements also needs to be explicit, which, again, motivates the use of goals as an output from this stage of analysis rather than a root definition.

5.4.5 KAOS

KAOS, like personas, was introduced in Chapter 2. Of the different Goal-Oriented Requirements Engineering approaches available, we chose KAOS for several reasons. First, KAOS defines goals as prescriptive statements of system intent. As such, KAOS goals appeal to our idea of a goal model as synonymous with an objective model of system requirements. Second, because KAOS goals deal with system, rather than user, intent, when determining how a requirement is refined from a goal, we only need to reason about goal attributes for different environments, rather than goal attributes for different environments specific to different users. Third, the goal modelling notation for KAOS is less semantically rich than i*-derived approaches, and, as such, less likely to be misinterpreted during design sessions. Finally, recent work on KAOS [255] illustrates how ideas from other goal-oriented approaches can be applied in KAOS, and how KAOS elements inter-relate with UML.

While KAOS does not distinguish between a goal and a requirement, IRIS does. The environmental variations between goal properties make this necessary; a requirement realised as part of a design is expected to hold in all contexts of use. Like KAOS, goals can be assigned to roles, but we use the recent extensions proposed by [255], and described in more detail in Section 4.7; these allow roles to also depend on other roles for goals, assets, and tasks as well.

KAOS bridges between Usability and Security techniques in three different ways. First, rather than operationalising goals by operations, we instead choose to operationalise goals using tasks. Although less precise, this modification affords the complementary use of goals and tasks for imparting the impact of goals on tasks, and vice versa in design sessions where non-technical stakeholders are present. Second, goals may be obstructed by obstacles, but we interpret these as both the accidental or intentional obstruction of goals. As Section 4.5 indicates, these can be associated with threats or vulnerabilities. Third, we apply recent work [255] which describes how *concern links* can be associated between goals and classes; these can be used to model traceability links between goals and any assets they reference or constrain.

5.4.6 Volere requirements specification

Volere is a framework for Requirements Engineering. The framework is described in detail by [201], and encapsulates industry best practice for scoping, eliciting, specifying, and validating requirements. A characteristic element of Volere is its Requirements Specification template. This describes the format a specification document should take, and the elements of a single requirement. These elements are described as a *Requirements Shell* and include the following attributes:

- Requirement Number: a unique identifier for a requirement.
- Requirement Type: the type of requirement; requirement types are also part of the Volere template.
- Event / Use Case Number: a reference to the events or use cases needing this requirement.
- Description: a single sentence describing the requirement's intent.
- Rationale: a justification for the requirement.
- Originator: the person who raised the requirement.
- Fit Criterion: a description of how the requirement can be tested to indicate that a solution satisfies it.
- Customer Satisfaction and Dissatisfaction: scores, from 1 to 5, indicating how satisfied a customer would be if the requirement was part or not part of the final product respectively.
- Priority: a rating of customer value.
- Supporting Material: references to supporting material documentation.
- History: information about changes to a requirement.

Although one of several commonly used requirements templates, e.g. [93, 132], the Volere template is distinctive in that it has evolved in line with industry best practice. It also contains several unique elements, which encourage the authorship of high-quality requirements; the most cited of these is the Fit Criterion, the lack of which indicates that a requirement is non-testable.

While the IRIS requirements specification template is largely based on that prescribed by Volere, there are a differences in the attributes of requirements, and the format of the template itself.

As indicated in Section 4.5, each requirement in IRIS either references or constrains an asset or environment. At a cosmetic level, this is reflected by the requirement identifier, which is prefixed by a short

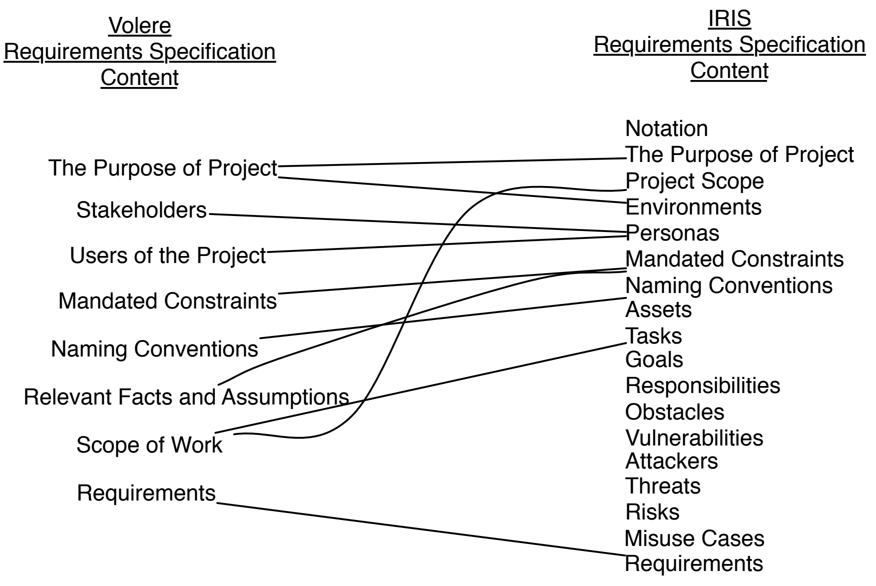


Figure 5.4: Mapping between Volere and IRIS requirements specification templates

code referencing an asset or environment, e.g. FW-1, where FW is a short code reference to a *Firewall* asset. Attributes of an IRIS requirement also do not include specific fields for history and customer satisfaction / dissatisfaction. In the case of the former, this is an attribute for tool-support, rather than something an analyst should have to manually maintain. In the case of satisfaction and dissatisfaction, it is arguably difficult to obtain reliable values for these attributes. The source of satisfaction or dissatisfaction may originate in early, contributory analysis, or may arise from a participant's subjective stake in the project. Although understanding the underpinnings of such value is useful, these are attributes best dealt with by Usability perspective techniques.

As Figure 5.4 illustrates, several new sections have been inserted into the requirements specification template. A notation section has been added to explain the modelling notations used in the requirements document; these notations are described in more detail in Chapter 6. Several sections have also been added to record the usability and security analysis contributing to the specified requirements. The ordering and naming of some sections have also been modified in IRIS; information about the scope is described prior to information about system environments and users (or rather personas) associated with the system. Finally, use cases are used by Volere as a bounding activity. In IRIS, tasks form part of this bounded analysis.

5.4.7 AEGIS

The AEGIS approach, which was briefly introduced in Section 2.1.2.2, is a participative risk analysis process. Under the guidance of a facilitator, and with the aid of one or more security experts, participants carry out an asset modelling exercise, identify risks from threats and vulnerabilities, select responses to these risks, and propose high-level security controls which counter risks requiring mitigation.

While there are several approaches to dealing with the analysis and management of risks, e.g. [70, 165], AEGIS is attractive for a number of reasons. First, AEGIS is built upon a standardised meta-model; this facilitates the development of software tools to support it. Second, AEGIS recognises that a single asset may have different values depending on the contexts it may be situated in; this facilitates approaches where the impact of a risk can be explored in different contexts of use. Third, AEGIS is a lightweight technique and not overly prescriptive on the techniques used to carry out the analysis it requires. For example, AEGIS prescribes the identification of vulnerabilities and threats, and the elicitation of requirements, but provides no explicit guidance on how these activities should be achieved. While this might be construed as a limitation, this is an advantage for IRIS as these activities can be carried out by complementary techniques.

The major difference between standard AEGIS, and AEGIS used as part of IRIS, is that applying the technique in focus groups is no longer mandatory. Although useful, participatory design activities can be time-consuming to participants, and the overall process of design may drag on if key participants are unavailable, or become compromised if participants are non-representative of users ultimately using the system. In lieu of multiple participants commenting on an asset model, insights are gleaned by interfacing AEGIS asset models with multiple techniques. For example, the asset modelling phase in AEGIS may precede obstacle modelling in KAOS to identify vulnerabilities; these can be examined in more detail with other vulnerabilities and threats by AEGIS. Similarly, goal modelling to identify the requirements a countermeasure needs to satisfy may be injected as a stage between AEGIS risk response and countermeasure selection.

Another assumption made by IRIS when using AEGIS is the specification of risk analysis elements; these are mandated by the IRIS meta-model. In lieu of any specific profiling technique, assumptions about the nature of attackers, including their motives and capabilities, need to be stipulated as part of threat identification. Risks also need to be explicitly defined as a single threat exploiting a single vulnerability. AEGIS does not distinguish between vulnerabilities, threats, and risks, nor are these concepts present within the AEGIS meta-model. By standardising on the elements of risk analysis, opportunities are now available for creating models of on-going risk analysis for different environments, rather than simply relying on UML based asset models.

5.4.8 Misuse cases

Misuse cases, which were introduced in Section 2.3.3.1, describe how hostile stakeholders carry out intentional threats to precipitate undesired system behaviour [12]. Like use cases, and scenarios in general, misuse cases add context to risk analysis by modelling the attacker, and describing how an attacker can exploit or misuse the system.

Although IRIS obtains the same value from misuse cases as the literature suggests, it reaches the final end using a different means. Rather than using them to elicit and explore threats, IRIS uses misuse cases to validate risks. For a risk to be considered valid, a misuse case needs to be authored commensurate to the risk's impact which, in turn, is commensurate to the attributes of the exploited vulnerability, and the threat taking advantage of this. If a valid scenario cannot be written based on this analysis, then the underlying analysis needs to be re-visited. The IRIS meta-model is such that, with suitable tool-support, it should only be necessary for analysts to write a misuse case narrative because contributing information, such as the misuse case objective and attacker can be determined from existing risk analysis data; an example of such tool-support is presented in Chapter 6.

5.5 Chapter summary

In this chapter, we presented the process framework used by IRIS. The framework builds upon the meta-model described in Chapter 4 by grouping IRIS concepts by Usability, Security, and Requirements perspectives. Based on these perspectives, individual IRIS processes for eliciting and specifying secure system requirements can be constructed using a number of exemplar techniques described in Section 5.4. In Chapters 7, 8, and 9, we present real-world case studies that apply processes instantiated by this framework.

Chapter 6

Tool-Support for Usable Security

This chapter presents CAIRIS (Computer Aided Integration of Requirements and Information Security): a software tool designed to embody the characteristics needed to support the IRIS framework.

We introduce the design principles that guided the development of CAIRIS in Section 6.1, before briefly describing how the tool was developed in Section 6.2. We then review the design of CAIRIS in terms of its high level architecture, physical deployment, and visual layout in Section 6.3, before describing how the tool's characteristics satisfy the design principles we wish to foster in Section 6.4.

6.1 CAIRIS design principles

So far, this dissertation has alluded to four design principles that tool-support for usable secure Requirements Engineering needs to support: Familiarity, Extensibility, Process Centricity, and Security and Usability Centricity.

6.1.1 Familiarity

Given the difficulty associated with grasping new concepts and learning new notations, the tool and its artifacts need to be familiar. Adopting a tool should require no more cognitive overhead than learning how to use the techniques associated with IRIS; as Section 5.4.5 discussed, minimising this overhead was one of the motivations for choosing KAOS as a goal-modelling technique over other approaches.

When new notations are introduced, these need to be parsimonious in terms of visual complexity. Rather than introduce complexity to a modelling notation, as indicated in Section 2.5.3, Schneiderman's Visual Information Seeking Mantra should be employed to ensure model complexity can be reduced where the context allows.

6.1.2 Extensibility

Given the nature of the Action Research interventions planned in Section 3.2.2.3, a practical tool also needs to be designed for extensibility in mind. When applying IRIS, new insights may arise; these may suggest functionality which is superfluous, or additional functionality that may be needed. Moreover, because these insights may become apparent only when they suggest needed features, it should be possible to quickly modify the tool to take advantage of these during, or shortly after, an intervention.

6.1.3 Process centricity

Section 2.5 suggests that tool-support needs to reflect how Security Requirements Engineering and HCI techniques might complement each other. The need to demonstrate this was reinforced by Chapter 5 where specific changes to techniques were proposed. Consequently, tool-support needs to be situated with an IRIS process being adopted. In particular, it needs to support opportunities for applying different techniques, and techniques which are used at the beginning and end of an IRIS process.

Tool-support is invaluable for not only ensuring that the data collected is appropriately structured, but the contribution that one technique makes to another is clear. For example, if using a particular object is critical for the successful completion of a task, then, by stating this object is a task concern, it should also be considered as an asset for risk analysis purposes.

Dealing with the initial stages of an IRIS process requires the management of project meta-data, such as project background, scoping information, and curatorship of project or domain specific definitions used during the process. Towards the end of an IRIS process, thought needs to be given to how data within the tool will be used by other stakeholders. For example, requirements may need to be reviewed by project sponsors, or models may need to be available to designers responsible for implementing an architecture the requirements need to satisfy.

6.1.4 Security and usability centricity

As Section 2.5 suggested, relying on *bolt-on* heuristics and verification techniques are not sufficient for analysing security concerns. The tool needs to support analysts who apply security and usability design techniques on an ad-hoc basis, at any stage in an IRIS process.

Tools that build upon the IRIS meta-model can rely on data being structured and categorised in a certain way. To take advantage of this, however, two considerations need to be factored into the design of tool-support. First, modelling support is needed for eliciting IRIS concepts; this includes, where possible, automatically inferring traceability between concepts. Second, given both the volume and disparity of the model data, the tool needs to automate some of the analysis necessary to explore the security and usability impact of certain design decisions.

6.2 Tool development process

CAIRIS was written primarily in Python [18], and used the open-source wxPython [19] and pyGTK [195] frameworks for windowing and visualisation support, and NumPy [16] for matrix manipulation. MySQL [15] was used for management and access to model data.

CAIRIS was developed using an iterative prototyping approach in five iterations. In the first phase, CAIRIS evolved inline with the meta-model. The tool was used to elicit data using contemporary examples where multiple contexts of use were evident. One of these examples involved analysing contemporary news reports and documentation about the Vélib bicycle sharing system, e.g. [191, 28] to elicit requirements for security controls which would not compromise the usability of Vélib. The objective of this phase was to determine whether the concepts necessary to model the different problems were reflected in the IRIS meta-model.

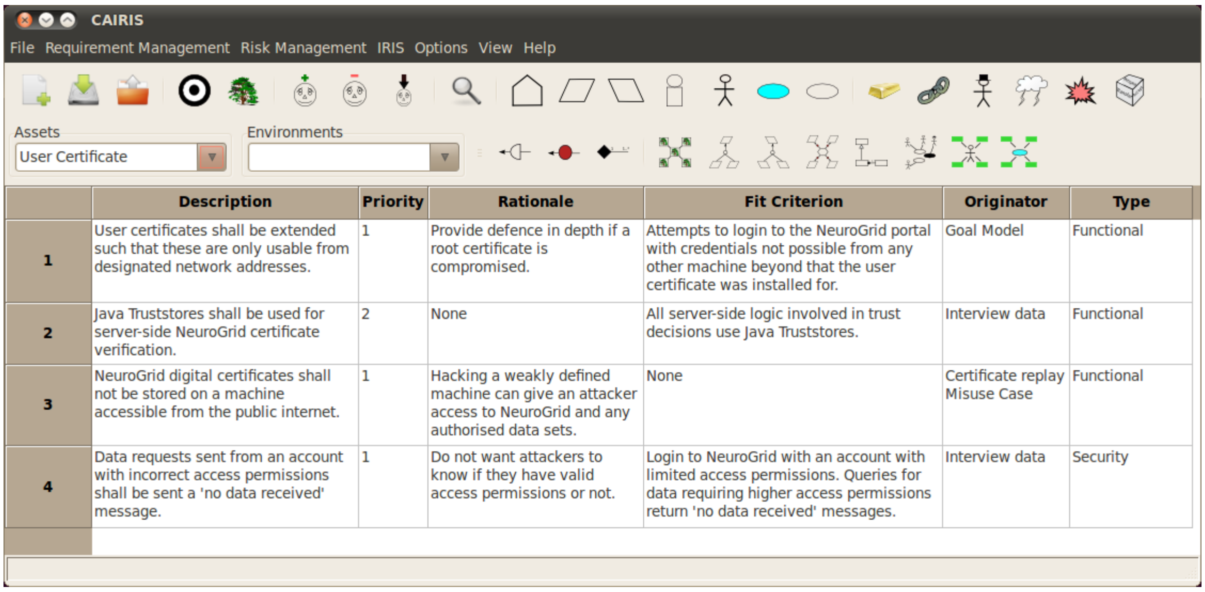
Based on early feedback from CAIRIS at academic fora [84, 83], additional concepts were added to the IRIS meta-model and the tool was evolved to support these. As the meta-model became more elaborate, additional model views were incorporated into the tool, and the architecture was re-factored to allow scaleability should further model elements and associations need to be added. At this stage, the NeuroGrid exemplar was developed to validate whether the tool was capable of modelling a complete, non-trivial problem. Using CAIRIS, the resulting NeuroGrid model was validated with one of the previous project stakeholders. Further feedback about both the model and tool was obtained during a poster presentation and tool-demonstration at the 17th IEEE International Requirements Engineering conference [87].

The tool was progressively refined for additional model concepts and bug-fixes during and following the three reported Action Research interventions (Section 3.2.2.3) where the IRIS framework was applied. Further details about the CAIRIS modifications made are described within the case study chapters.

Modifications to the CAIRIS source code were managed using a Subversion revision control system [103]. The CAIRIS Subversion repository was hosted and maintained by the Oxford University Computing Laboratory support team. A CAIRIS web portal was also set up to host stable releases of CAIRIS and its source code for interested developers and researchers [86].

6.3 Architectural design

We now describe the architectural design of CAIRIS. We begin by introducing the interface elements of the CAIRIS GUI, before describing how the high level architecture of CAIRIS maps to the elements of the Model-View-Controller architectural framework [198]. We conclude this brief introduction to the CAIRIS architecture with an overview of how the physical artifacts of CAIRIS are deployed.



Assets: User Certificate

Environments:

	Description	Priority	Rationale	Fit Criterion	Originator	Type
1	User certificates shall be extended such that these are only usable from designated network addresses.	1	Provide defence in depth if a root certificate is compromised.	Attempts to login to the NeuroGrid portal with credentials not possible from any other machine beyond that the user certificate was installed for.	Goal Model	Functional
2	Java Truststores shall be used for server-side NeuroGrid certificate verification.	2	None	All server-side logic involved in trust decisions use Java Truststores.	Interview data	Functional
3	NeuroGrid digital certificates shall not be stored on a machine accessible from the public internet.	1	Hacking a weakly defined machine can give an attacker access to NeuroGrid and any authorised data sets.	None	Certificate replay Misuse Case	Functional
4	Data requests sent from an account with incorrect access permissions shall be sent a 'no data received' message.	1	Do not want attackers to know if they have valid access permissions or not.	Login to NeuroGrid with an account with limited access permissions. Queries for data requiring higher access permissions return 'no data received' messages.	Interview data	Security

Figure 6.1: CAIRIS screenshot

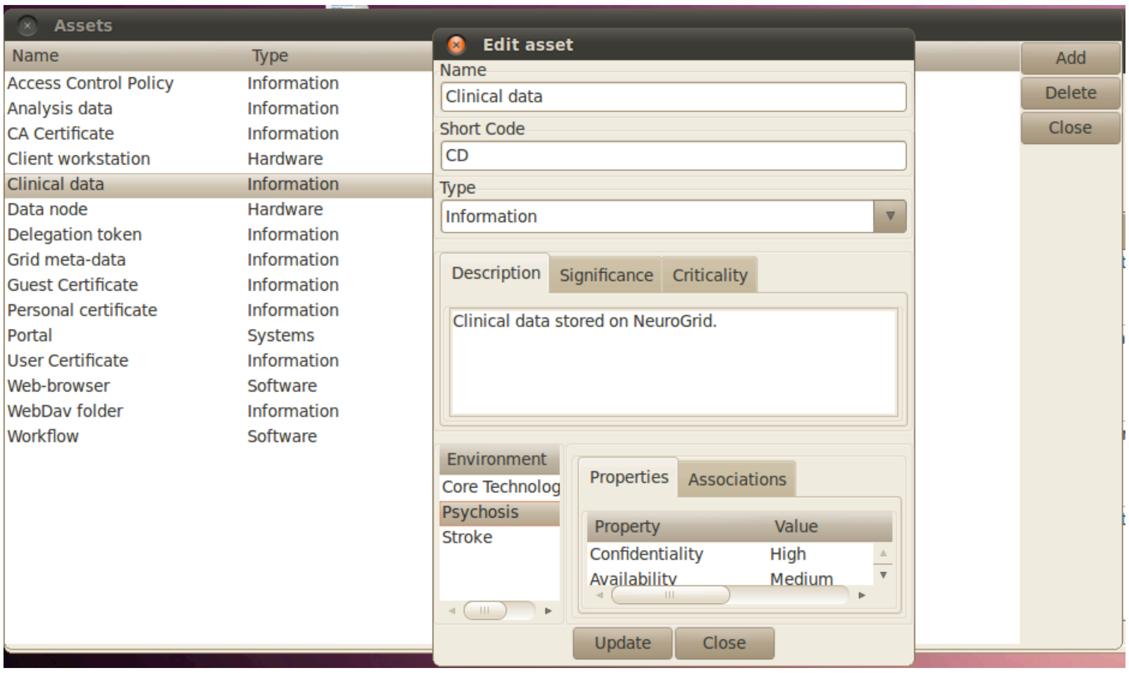
6.3.1 Visual interface design

We assume that an analyst understands both the principles of IRIS, and the techniques selected in a design process. For this reason, although a manual has been written for the tool [85], incorporating in-built help into CAIRIS was not a priority. However, tool-tip text boxes have been incorporated; these are displayed when the mouse pointer hovers over a toolbar button.

The main GUI is divided into two main segments. The top half of the screen contains menu options and toolbar buttons. The toolbar button layout is based on the principle that elements should be grouped and a clear hierarchy should be provided [61].

These toolbar buttons are laid out in two rows. The top-most rows contain buttons for adding, editing, or deleting model objects. These objects correspond to concepts in the IRIS meta-model. The toolbar buttons themselves are laid out according to when these concepts might be elicited in a typical IRIS process. For example, following the buttons for creating new models, saving, and opening existing models, the left-most button is the project options button; this opens a dialog box where project meta-data might be added. The buttons to its immediate right concern environment and role objects. The right-most buttons concern risks, responses, and countermeasures.

The second row is concerned with associations between model objects. The left-most buttons allow associations to be added between certain concepts, specifically asset associations, goal and obstacle refinement associations, and responsibility and dependency relationships. The buttons to the right of these allow the model to be viewed from the different model perspectives: Risk Analysis, Asset, Goal,



Name	Type
Access Control Policy	Information
Analysis data	Information
CA Certificate	Information
Client workstation	Hardware
Clinical data	Information
Data node	Hardware
Delegation token	Information
Grid meta-data	Information
Guest Certificate	Information
Personal certificate	Information
Portal	Systems
User Certificate	Information
Web-browser	Software
WebDav folder	Information
Workflow	Software

Name: Clinical data
Short Code: CD
Type: Information

Clinical data stored on NeuroGrid.

Environment
Core Technolog
Psychosis
Stroke

Property	Value
Confidentiality	High
Availability	Medium

Figure 6.2: Textual view of CAIRIS model objects

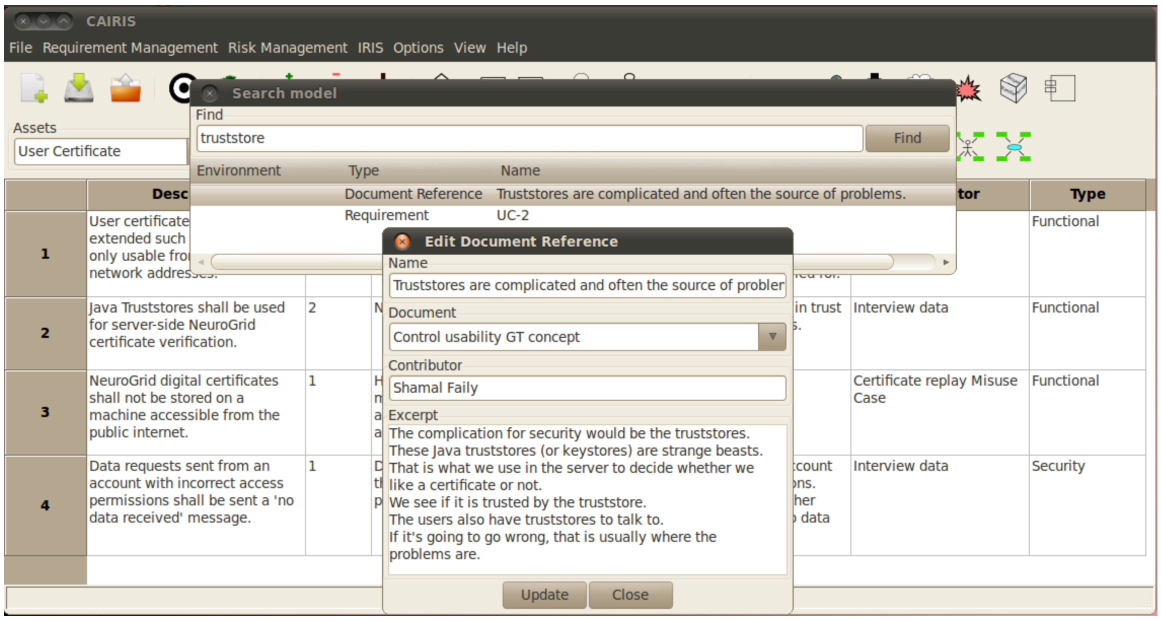
Obstacle, Responsibility, Task, Persona, and Task Argumentation.

These buttons are only a sub-set of the options available from the menu bar, however these buttons tend to be the most used.

The bottom half of the screen incorporates the requirements editor. This editor is a table with combo boxes, allowing requirements to be grouped by asset or environment. Although the concept of requirement is one of many in IRIS, its prominence in the GUI is motivated by the centrality of this concept, and previous work reporting on the effectiveness of DOORS' tabular interface for collaborative requirements editing [82].

In general, model object interaction takes place via the dialog boxes accessible from the toolbar or menu. The objects associated with a particular concept are viewed by selecting the appropriate model menu or toolbar button. From these dialog boxes, objects can be added, deleted, or modified. Objects are modified by activating a highlighted object name from this list to open a dialog box specific to that object. This *Textual View* of a CAIRIS model is illustrated in Figure 6.2.

The above hierarchical approach to model object management does not take into account the ad-hoc nature of design. A new insight may make it necessary to quickly discover or modify objects according to some arbitrary category. With this in mind, *Find* functionality is included to find and modify model objects related to a specific term. Objects related to this term are listed by environment and object type. When viewing large models, the environment that an object is situated in provides useful information


CAIRIS
File Requirement Management Risk Management IRIS Options View Help
Assets
User Certificate
Search model
Find
Update
Close

Find: truststore

Environment	Type	Name
	Document Reference	Truststores are complicated and often the source of problems.
	Requirement	UC-2

Edit Document Reference

Name: Truststores are complicated and often the source of probler

Document: Control usability GT concept

Contributor: Shamal Faily

Excerpt: The complication for security would be the truststores. These Java truststores (or keystores) are strange beasts. That is what we use in the server to decide whether we like a certificate or not. We see if it is trusted by the truststore. The users also have truststores to talk to. If it's going to go wrong, that is usually where the problems are.

	Desc			tor	Type
1	User certificate extended such only usable fro network addres				Functional
2	Java Truststores shall be used for server-side NeuroGrid certificate verification.	2		Interview data	Functional
3	NeuroGrid digital certificates shall not be stored on a machine accessible from the public internet.	1		Certificate replay Misuse Case	Functional
4	Data requests sent from an account with incorrect access permissions shall be sent a 'no data received' message.	1		Interview data	Security

Figure 6.3: Finding and updating for model elements

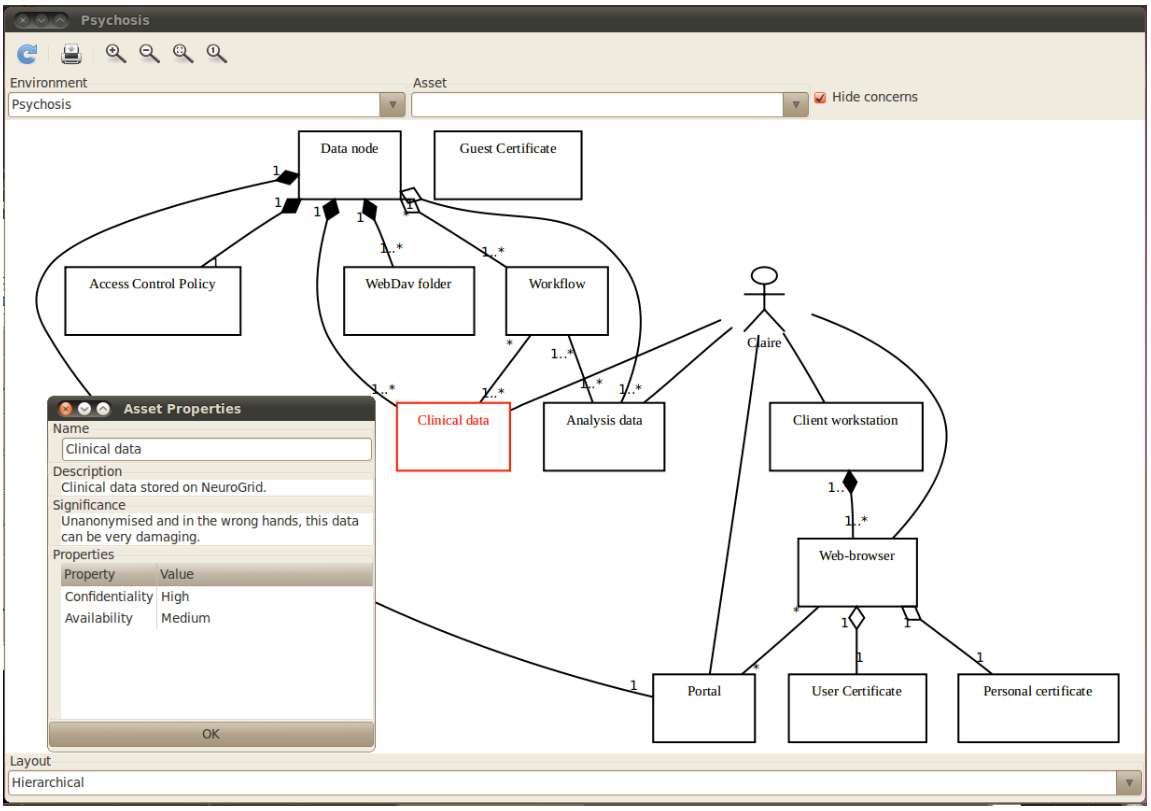
about the object's associated contexts. If a non-requirement object is activated in the search dialog, then the dialog box associated with the object is opened. This allows an analyst to browse or modify the object, as illustrated in Figure 6.3. In the case of requirements, the requirements editor is updated and the selected requirement is highlighted.

Although CAIRIS models are largely built up using textual view interfaces, it is also possible to interact with graphical views of these models. The CAIRIS Model Viewer component supports the visualisation of one of several *Graphical Views*. As Figure 6.4 shows, hit-testing is supported by this model viewer component, allowing further information about an object to be obtained.

6.3.2 High level architecture

At an abstract level, CAIRIS instantiates a Model-View-Controller architectural framework [198], as illustrated in Figure 6.5. The central component in the CAIRIS architecture is the CAIRIS GUI; it is through this that an analyst interacts with the objects stored in CAIRIS. At any given time, CAIRIS accesses a requirements model. This model represents a collection of requirements objects associated with a selected asset or environment. Analysts interact with this model using the requirements editor in the CAIRIS GUI; this editor is described in more detail in Section 6.3.1.

CAIRIS can also manage one of several other models in addition to the requirements model. Textual models are instantiated if an analyst wishes to add, modify, or remove a CAIRIS object. In any given model, an analyst may modify only those objects related to the model, and other objects visible by virtue



Asset Properties

Name: Clinical data

Description: Clinical data stored on NeuroGrid.

Significance: Unanonymised and in the wrong hands, this data can be very damaging.

Properties

Property	Value
Confidentiality	High
Availability	Medium

Figure 6.4: Graphical view of CAIRIS model objects

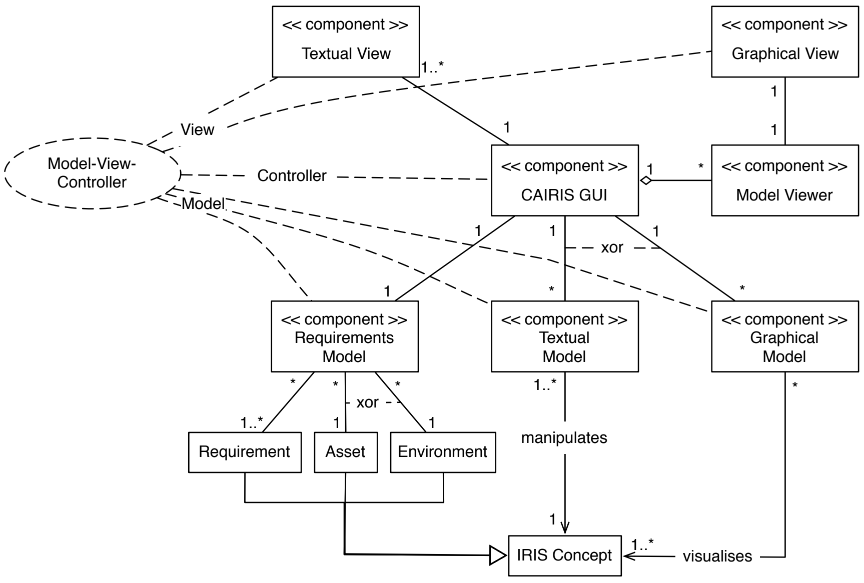
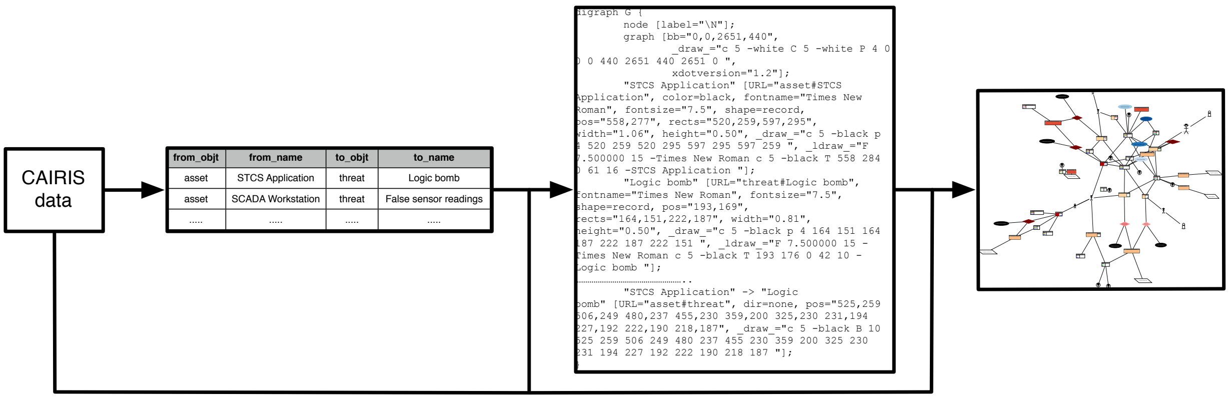

<< component >> Textual View
<< component >> Graphical View
Model-View-Controller
View
Controller
Model
<< component >> CAIRIS GUI
<< component >> Model Viewer
xor
<< component >> Requirements Model
<< component >> Textual Model
<< component >> Graphical Model
Requirement
Asset
Environment
manipulates
IRIS Concept
visualises
1..*
1
*

Figure 6.5: Component diagram of key CAIRIS components



from_objt	from_name	to_objt	to_name
asset	STCS Application	threat	Logic bomb
asset	SCADA Workstation	threat	False sensor readings
.....	.....	.....	.....

```
igraph G {
        node [label="\N"];
        graph [bb="0,0,2651,440",
                _draw_="c 5 -white C 5 -white P 4 0
 0 440 2651 440 2651 0 ",
                xdotversion="1.2"];
        "STCS Application" [URL="asset#STCS
pplication", color=black, fontname="Times New
oman", fontsize="7.5", shape=record,
os="558,277", rects="520,259,597,295",
idth="1.06", height="0.50", _draw_="c 5 -black p
 520 259 520 295 597 295 597 259 ", _ldraw_="F
.500000 15 -Times New Roman c 5 -black T 558 284
 61 16 -STCS Application "];
        "Logic bomb" [URL="threat#Logic bomb",
ontname="Times New Roman", fontsize="7.5",
hape=record, pos="193,169",
ects="164,151,222,187", width="0.81",
eight="0.50", _draw_="c 5 -black p 4 164 151 164
87 222 187 222 151 ", _ldraw_="F 7.500000 15 -
imes New Roman c 5 -black T 193 176 0 42 10 -
ogic bomb "];
...........................................
        "STCS Application" -> "Logic
omb" [URL="asset#threat", dir=none, pos="525,259
06,249 480,237 455,230 359,200 325,230 231,194
27,192 222,190 218,187", _draw_="c 5 -black B 10
25 259 506 249 480 237 455 230 359 200 325 230
31 194 227 192 222 190 218 187 "];
```

Figure 6.6: Pipeline for visual model generation

of their association with the object. For example, if a textual model is instantiated for personas, then only the intrinsic attributes of that concept may be modified, e.g. its name and behavioural characteristic. The model may be modified such that a persona is associated to one or more roles, but the roles themselves cannot be removed via this model, only a persona's association with them.

Both the requirements and textual model are manipulated via a textual view. Using database stored procedures, a tabular representation of this data is built-up. CAIRIS creates a collection of model objects based on this representation, and displays the objects using tabular and list controls.

Graphical models, as the name suggest, are manipulated via a Graphical View. Both models and views are abstractions for the corresponding IRIS models. The process of building graphical models is an extension of that used for textual views, however rather than projecting the model objects directly to GUI lists and tables, further processing is carried out, both within and outside CAIRIS; this is illustrated in Figure 6.6.

A Graphviz [21] model of nodes and edges is created from the model data and, based on this, a representation of this model is built up with information about how the model should be laid out on screen. Using components from the Pycairo and XDot open source packages [101], this XDot information is rendered in SVG and displayed using the model viewer component described in Section 6.3.1.

Although the pipeline follows a sequential process, at certain times the CAIRIS database is consulted for supplemental information. For example, the colours for risk nodes are derived from a risk score which is calculated while building the XDot representation. Similarly, when displaying risk analysis views, information about asset and threat security properties are collected when rendering the graphical models during the final stage in the process.

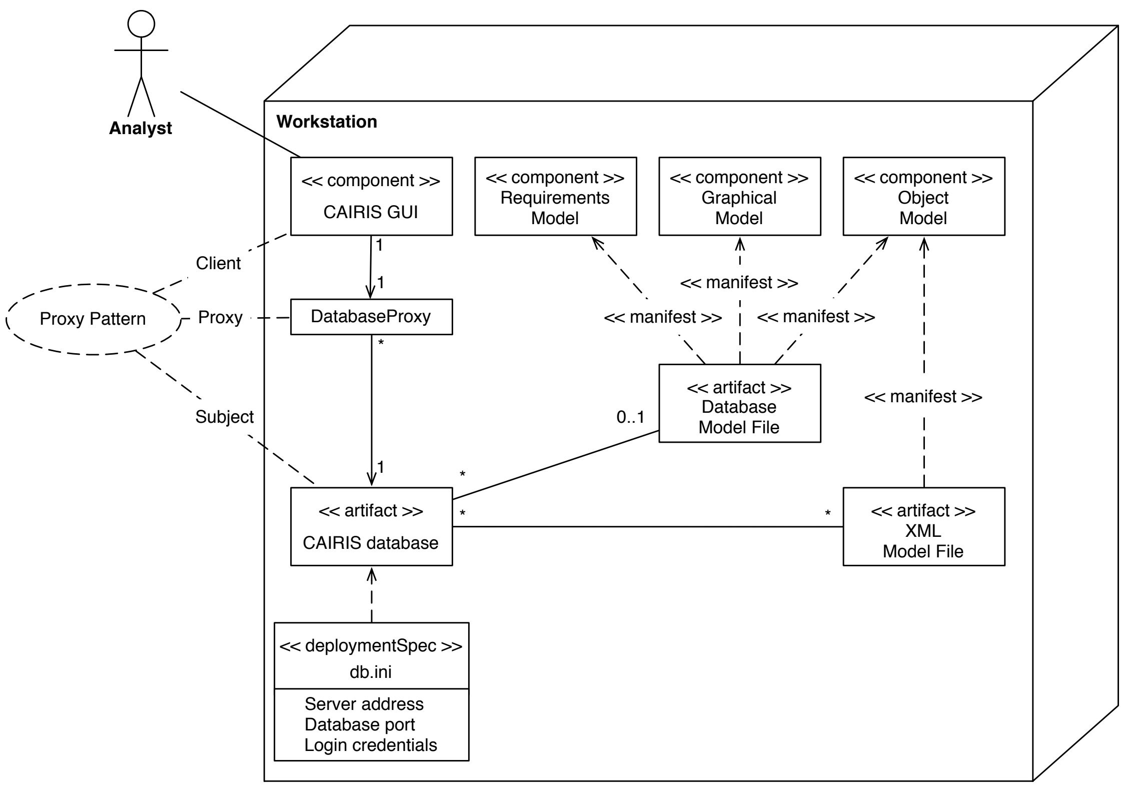

Analyst
Workstation
<< component >> CAIRIS GUI
<< component >> Requirements Model
<< component >> Graphical Model
<< component >> Object Model
Client
Proxy Pattern
Proxy
Subject
DatabaseProxy
<< manifest >>
<< manifest >>
<< manifest >>
<< manifest >>
<< artifact >> Database Model File
0..1
<< artifact >> CAIRIS database
<< artifact >> XML Model File
<< deploymentSpec >> db.ini
Server address
Database port
Login credentials
1
1
*
1
*
*
*

Figure 6.7: Physical deployment diagram of CAIRIS

6.3.3 Physical deployment

We assume that an analyst will use CAIRIS on a single workstation or laptop. The sensitivity of the data stored by CAIRIS demands that this be stored in a secure manner. However, from the perspective of this dissertation, we assume that the hardware node that hosts CAIRIS is secure.

At startup, CAIRIS connects to a single CAIRIS backend database. The connection settings to this database are defined in an ASCII text configuration file stored locally to the CAIRIS binary. The CAIRIS database constitutes both the CAIRIS database schema, and the stored procedures and functions used to manipulate the database. With the exception of opening and saving model files, the Proxy pattern [105] is used to manage access to the database, as illustrated in Figure6.7.

Two additional types of artifact are associated with CAIRIS. The first of these are CAIRIS *Database Model Files*, which store serialised CAIRIS models. For simplicity, these are implemented as SQL database dumps. These are created when a model file is saved within CAIRIS. When a model file is opened, the existing database structure is dropped and the structure and data within the file is re-created.

The second type of artifact are *XML Model Files*. These are XML files containing partial model data

Principle	Characteristic
Familiarity	Model Visualisation
Extensibility	Tool Development Process
Process centricity	Model Externalisation
Security and Usability centricity	Automated Analysis, Model Reusability

Figure 6.8: Principle to characteristic mapping

which can be imported into a current CAIRIS database instance. These are described in more detail in Section 6.4.3.

6.4 Tool-support characteristics

In the following sections, we describe characteristics that appeal to the design principles identified in Section 6.1. Figure 6.8 summarises the mapping between the presented design principles and tool-support characteristics.

6.4.1 Automated analysis

The techniques used by the IRIS framework are useful for discussing the impact that security and usability can have on a system design. Unfortunately, risk and usability ratings are sufficiently coloured by analyst perceptions that human error can creep into valuations. Without a means to rapidly score and visualise the current state of analysis, the security–usability balance can become uneven as risk analysis becomes more advanced.

In this section, we describe how simple quantitative data analysis can complement task and risk analysis activities, forming the basis of automated analysis for supporting security and usability design decisions.

6.4.1.1 Task analysis

Data about how target users plan to use the system-to-be is modelled in CAIRIS using personas and tasks. Although there is no agreed way of measuring the usability of a task with respect to its participating personas, persona and task usability attributes match attributes found in the ISO 9241-11 framework [136]. ISO 92411-11 describes how usability goals can be evaluated using the goals of effectiveness, efficiency, and satisfaction. Based on this framework, a set of task usability properties were devised (Figure 6.9); these can be used to evaluate how usable a task is to a persona. When defining tasks, these

Property	ISO 9241-11 Usability Component	Description	Values
Duration	Efficiency	The time taken by a persona to complete the task.	• None • Seconds • Minutes • Hourly or longer
Frequency	Efficiency	The frequency a persona carried out the task.	• None • Hourly or more • Daily - Weekly • Monthly or less
Demands	Satisfaction	The mental or physical demands on a persona.	• None • Low • Medium • High
Goal Conflict	Effectiveness	The degree to which the task interferes with the persona's work or personal goals.	• None • Low • Medium • High

Property	ISO 9241-11 Usability Component	Description	Values
Duration	Efficiency	The degree to which the countermeasure helps or hinders the time taken by a persona to complete the task.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance
Frequency	Efficiency	The degree to which the countermeasure increases or decreases the frequency a persona needs to carry out the task.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance
Demands	Satisfaction	The degree to which the countermeasure increases or decreases the mental or physical demands on a persona while carrying out the task.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance
Goal Conflict	Effectiveness	The degree to which the task helps or hinders the persona's work or personal goals.	• High Help • Medium Help • Low Help • None • Low Hindrance • Medium Hindrance • High Hindrance

Figure 6.9: Task (left) and countermeasure task (right) usability properties

four properties are set for each persona participating in a scenario. Each of these properties map to one of the usability components of ISO 9241-11.

$$U_T = \overline{\frac{t+f}{2}} + \overline{m} + \overline{c} \tag{6.1}$$

Each property has an associated value x which maps to a natural number in the range $0 \leq x \leq 3$; this corresponds to the qualitative values of None, Low, Medium, and High respectively. To ensure equal weighting for all three usability components, the usability of a task T, U_T, is computed using Formula 6.1, where $\overline{\frac{t+f}{2}}$ is the mean task efficiency, $\overline{m}$ is the mean task satisfaction, and $\overline{c}$ is the mean task effectiveness. Variables t, f, m, and c refer to the task duration, frequency, mental demands, and goal conflict respectively. The mean value is taken across all personas carrying out the task in question. The higher the value of U_T the less usable a task is for the personas associated with it. More meaningful values must be used for duration and frequency because values like Low, Medium, and High are ambiguous. For the duration of time, the qualitative ratings used are *Seconds*, *Minutes*, and *Hours or Longer* are associated with the values 1,2, and 3 respectively. For frequency, the ratings used are *Monthly or less*, *Daily - Weekly*, and *Hourly or more.*

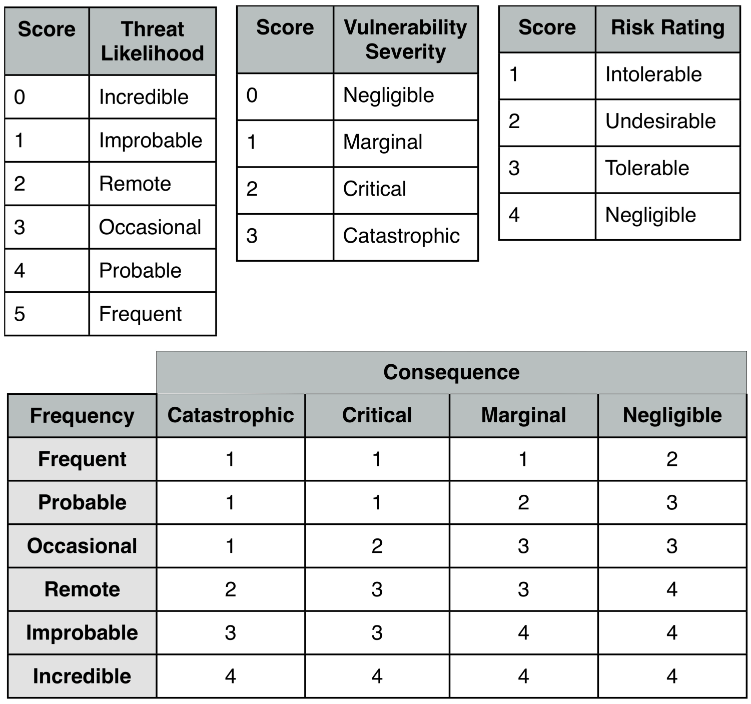
As Section 4.7 reports, when mitigating risks, one or more roles are associated with each mitigating countermeasure; these roles will, in some way, be directly affected by the countermeasure being designed. Consequently, for each task–persona pairing, countermeasure usability properties can be specified.

$$CU_T = \overline{\frac{t+f}{2}} + \overline{m} + \overline{c} \tag{6.2}$$

Based on this countermeasure usability data, it is possible to calculate the countermeasure usability factor for task T - CU_T. The right-hand side of the equation computing CU_T is identical to U_T as Formula 6.2 demonstrates. The values are, however, different. $\overline{\frac{t+f}{2}}$ is the mean contribution to task efficiency, $\overline{m}$ is the mean contribution to task satisfaction, and $\overline{c}$ is the mean contribution to task effectiveness. Like U_T, the variables t, f, t, and c refer to the task duration, frequency, mental demands, and goal conflict respectively. The mean contributing value is taken across all countermeasures affecting the task in question. However, unlike U_T each qualitative value x associated with a property maps to an integer in the range $-3 \leq x \leq 3$. Based on these equations, we compute the task summative usability for task T, SU_T, to be

$$SU_T = U_T + CU_T \tag{6.3}$$

Like U_T, the higher the score, the less usable the task is for the associated personas. After calculating U_T and SU_T the score is normalised to a natural number in the range $0 \leq n \leq 9$. Given the potential of a task to increase or decrease usability, CU_T remains unchanged irrespective of it being a high positive or negative number.



Score	Threat Likelihood
0	Incredible
1	Improbable
2	Remote
3	Occasional
4	Probable
5	Frequent

Score	Vulnerability Severity
0	Negligible
1	Marginal
2	Critical
3	Catastrophic

Score	Risk Rating
1	Intolerable
2	Undesirable
3	Tolerable
4	Negligible

	Consequence			
Frequency	**Catastrophic**	**Critical**	**Marginal**	**Negligible**
Frequent	1	1	1	2
Probable	1	1	2	3
Occasional	1	2	3	3
Remote	2	3	3	4
Improbable	3	3	4	4
Incredible	4	4	4	4

Figure 6.10: IEC 61508 tables for threat likelihood, vulnerability severity, and risk categorisation

6.4.1.2 Risk analysis

As indicated in Section 4.6, an IRIS risk is based on the likelihood of a threat exploiting a vulnerability to cause an impact. Threats are synonymous to attacks, and vulnerabilities are properties of a system making it liable to exploitation.

$$L_R = L_{T_R} - \overline{m_{T_R}} \tag{6.4}$$

A risk rating can be assigned based on the likelihood and severity tables in IEC 61508 [130] shown in Figure 6.10. Although these tables were designed to support hazard and risk analysis for safety-critical system design, they are also equally applicable for categorising threats, vulnerabilities, and risks from a security perspective. However, this risk rating does not reflect values held about individual assets or threats. To score risks with respect to the perceived value of the assets threatened, we model a security property using a row vector $[ciao]$, where c, i, a, and o represent the values held for confidentiality, integrity, availability and accountability respectively. Each element n is scored $0 \leq n \leq 3$ based on whether the value held for that element is None, Low, Medium or High. The likelihood of the threat being realised, L_R , is computed using Formula 6.4, where L_{T_R} is the likelihood of the threat T associated with risk R, and $\overline{m_{T_R}}$ is the mean likelihood value for the set of countermeasures mitigating the likelihood of threat T_R. The values of L_{T_R} and $\overline{m_{T_R}}$ exist within the range $0 \leq n \leq 5$, and map to the likelihood categories in Figure 6.10.

$$S_R = S_{V_R} - \overline{m_{V_R}} \tag{6.5}$$

The severity of the vulnerability exposed by risk R is computed using Formula 6.5, where S_{V_R} is the severity of the vulnerability V associated with risk R , and $\overline{m_{V_R}}$ is the mean severity for the set of countermeasures mitigating the severity of vulnerability V_R. Like threat likelihood, vulnerability severity values exist within the range $0 \leq n \leq 3$ and map to the vulnerability categories in Figure 6.10.

$$I_R = (I_{T_R} \times I_{A_R}) - \overline{m_{I_R}} \tag{6.6}$$

Risk impact I is described by a security property, representing the values held in the assets at risk from threat T. Risk impact I_R is computed using Formula 6.6, where I_{T_R} is the security property of the threat associated with risk R, I_{A_R} is the security property of the vulnerable or threatened assets at risk, and $\overline{m_{I_R}}$ is the mean security property for the countermeasures targeting the risk's threat or vulnerability.

$$RS_R = L_R \times S_R \times I_R \tag{6.7}$$

Finally, the calculation for the risk score of risk R, RS_R, is computed as the product of the threat likelihood, the severity of the vulnerability, and the risk impact to the threatened asset, as shown in Formula 6.7.

Each element of the row vector is added together, and the sum is normalised to an integer between 1 and 9. If, during the above computations, negative numbers are calculated, these values are resolved to 0.

6.4.1.3 Example: scoring and visualising the upload of clinical data

We now present an example illustrating how task usability and risk can be scored based on the formulae in the previous sections. The example, which was also briefly introduced in Section 4.6, looks at the usability and security associated with uploading sensitive data to NeuroGrid. We have previously defined a task called *upload-data*, which is carried out by a previously introduced persona, Claire. This task begins with Claire anonymising a clinical data set; this involves removing as much personalised data as possible while still enabling her analysis software to work. Claire then uploads this data, tagging this as available only to members of her research group. In the example, Claire spends several minutes on this task, which she usually carries out each working day. Given the persona profile, this is a low demand task which generally supports her goals:

$$\begin{aligned} U_{upload-data} &= \overline{\frac{t+f}{2}} + \overline{m} + \overline{c} \\ &= \frac{2+2}{2} + 1 + 2 \\ &= 5 \end{aligned}$$

We now consider the risk of *unauthorised certificate access* (*uca*). In this example, the risk is realised by an attacker using *social engineering* (*soc-eng*) to obtain access to a client workstation and installing a user digital certificate. Such an attacker could be a journalist more interested in the newsworthiness of the attack than the value of the compromised data.

The attacker exploits a vulnerability arising from the difficulty of obtaining digital certificates. Rather than undertaking the seemingly onerous task of legitimately obtaining a digital certificate, which involves obtaining permission from line management, filling out forms, and sending several confirmatory emails to a certificate authority, many users instead choose to share digital certificates among themselves. Such insecure behaviour arises not from malevolence, but from a desire to carry out their work without undue hindrance. The attacker exploits this *certificate ubiquity* vulnerability (*c-ubiq*).

Realising the risk might involve a journalist obtaining site access, including access to a workstation, and masquerading as a new member of staff who, frustratingly, has not been given access to all the tools she needs to carry out her work. This attack can be elaborated using a misuse case to describe how an attacker pretends to be a new post-doctoral researcher whose new supervisor happens to be away from the office when she arrives for her first day at work.

This risk can be assessed both qualitatively and quantitatively. We can evaluate the risk qualitatively based on threat likelihood and vulnerability severity. These values were Occasional and Critical respectively, giving rise to a rating of Undesirable.

We assess this risk quantitatively by calculating its risk score, with the likelihood and severity scores mapped to 2 and 3 respectively. The threat targeted only the confidentiality of these assets, but two assets were targeted by this threat. These two assets were the client workstation ([1030]) and a user certificate ([2031]); for brevity, we denotes the sum of these properties using the variable *soc-eng-target*. The asset exposed by the vulnerability is the user certificate. Using this information, we calculated the risk score RS_{ucs}:

$$
\begin{aligned}
L_{uca} = L_{soc-eng_{uca}} - \overline{m_{soc-eng_{uca}}} &= 3 - 0 \\
&= 3 \\
S_{uca} = S_{c-ubiq_{uca}} - \overline{m_{c-ubiq_{uca}}} &= 2 - 0 \\
&= 2 \\
I_{uca} &= (I_{soc-eng_{uca}} \times I_{soc-eng-target_{uca}}) - \overline{m_{I_{uca}}} \\
&= (\begin{bmatrix}3 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}3 & 0 & 6 & 1\end{bmatrix}) \\
&= \begin{bmatrix}9 & 0 & 0 & 0\end{bmatrix} \\
RS_{uca} &= L_{uca} \times S_{uca} \times I_{uca} \\
&= 3 \times 2 \times \begin{bmatrix}9 & 0 & 0 & 0\end{bmatrix} \\
&= \begin{bmatrix}54 & 0 & 0 & 0\end{bmatrix}
\end{aligned}
$$

After rounding RS_{uca} down, the normalised score resolved to 9.

In this example, we mitigate this risk by making user certificates host-based, such that an issued certificate can only be used for a given client workstation. This countermeasure is considered highly effective at targeting the certificate ubiquity vulnerability, motivating the value of 3 (High) for $\overline{m_{c-ubiq_{uca}}}$. This countermeasure also fosters, albeit only to a minor extent, values of confidentiality and accountability, giving rise to a score of [1001] for $\overline{m_{I_{uca}}}$. Based on this information, the risk score can be re-evaluated:

$$
\begin{aligned}
L_{uca} = L_{soc-eng_{uca}} - \overline{m_{soc-eng_{uca}}} &= 3 - 0 \\
&= 3 \\
S_{uca} = S_{c-ubiq_{uca}} - \overline{m_{c-ubiq_{uca}}} &= 2 - 3 \\
&= -1 \\
I_{uca} &= (I_{soc-eng_{uca}} \times I_{soc-eng-target_{uca}}) - \overline{m_{I_{uca}}} \\
&= (\begin{bmatrix}3 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}3 & 0 & 6 & 1\end{bmatrix}) - \\
&\quad \begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix} \\
&= \begin{bmatrix}8 & 0 & 0 & -1\end{bmatrix} \\
RS_{uca} &= L_{uca} \times S_{uca} \times I_{uca} \\
&= 3 \times 0 \times \begin{bmatrix}8 & 0 & 0 & -1\end{bmatrix} \\
&= \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}
\end{aligned}
$$

These results show that while certain security properties remain threatened, the severity of the vulnerability was rendered inert, thereby reducing the risk score to the lowest possible value. As this countermeasure appeared to be effective, we chose to generate a new asset for this policy. The security properties of this new asset were based on the values placed on the countermeasure, which varied based on the contexts the new asset was situated in.

This countermeasure also impacts the usability of uploading data. Because Claire now needs to make sure she uploads data only from a particular machine, introducing this countermeasure is a slight hindrance to her other goals. As such, the summative task usability can now be evaluated.

$$\begin{aligned} SU_{upload-data} &= U_{upload-data} + CU_{upload-data} \\ &= 5 + \overline{\frac{0+0}{2}} + 0 + 1 \\ &= 6 \end{aligned}$$

As a consequence, the risk is mitigated, but the task usability is reduced due to the extra effort involved in complying with the new policy.

In Section 6.4.2.4, we illustrate how this risk mitigation exercise can be visualised.

6.4.2 Model visualisation

Given the coverage of the IRIS meta-model, relying only on a single graphical view leads to large and unwieldy models. It is, therefore, necessary to support multiple graphical views where each can be used for the analysis related to the concepts in each part of the IRIS meta-model. Each view also needs to be parsimonious in terms of visualised concepts, and facilitate the concept traceability between models.

In this section, we describe how the different IRIS models are visualised in their respective graphical views. In particular, we show how the results in Section 6.4.1 can be visualised in IRIS risk analysis models. Finally, we describe some of the techniques used by CAIRIS for managing clutter as models grow.

The notation for all models described in this, and other parts of the dissertation, is described in Section A.6.

6.4.2.1 Asset model

The CAIRIS Asset Model is represented as a UML class model, where assets are represented as classes. If an asset is used within a task then the persona associated with the task is also displayed in the asset model as an actor node.

Horizontal traceability from other models to the asset model is implemented using concern links. If assets or asset relationships of concern are identified in goals or tasks, these are automatically visualised in the asset model, depending on the level of model zoom. These concerns are displayed as comment nodes indicating the traceability origin of the asset.

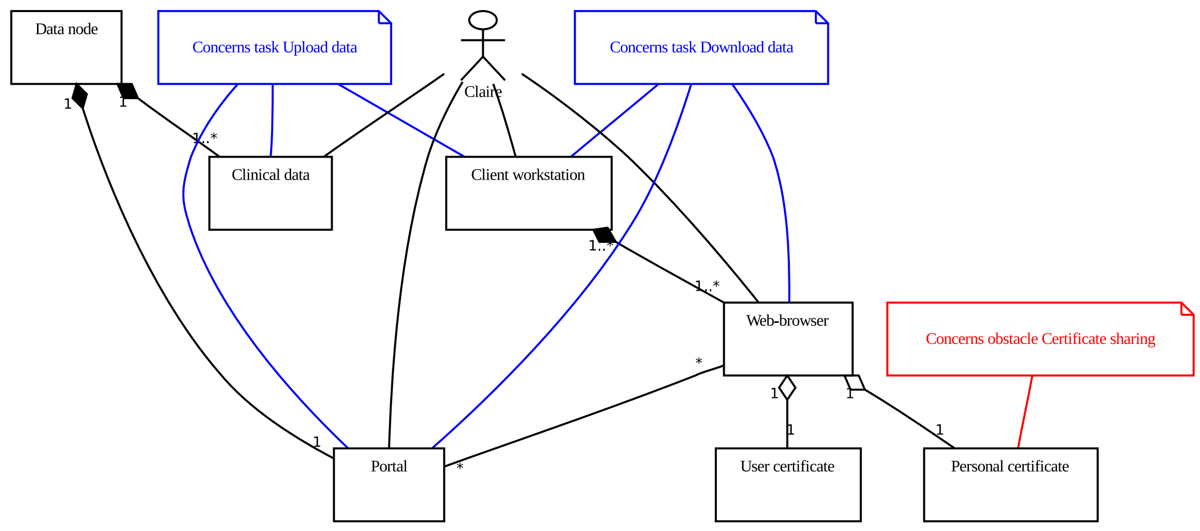


Figure 6.11: Asset model example

Assets may be generated from countermeasures as part of risk analysis. If this occurs, then an association between the asset at risk and the countermeasure protecting it is generated; this association is labelled with a *safeguards* stereotype.

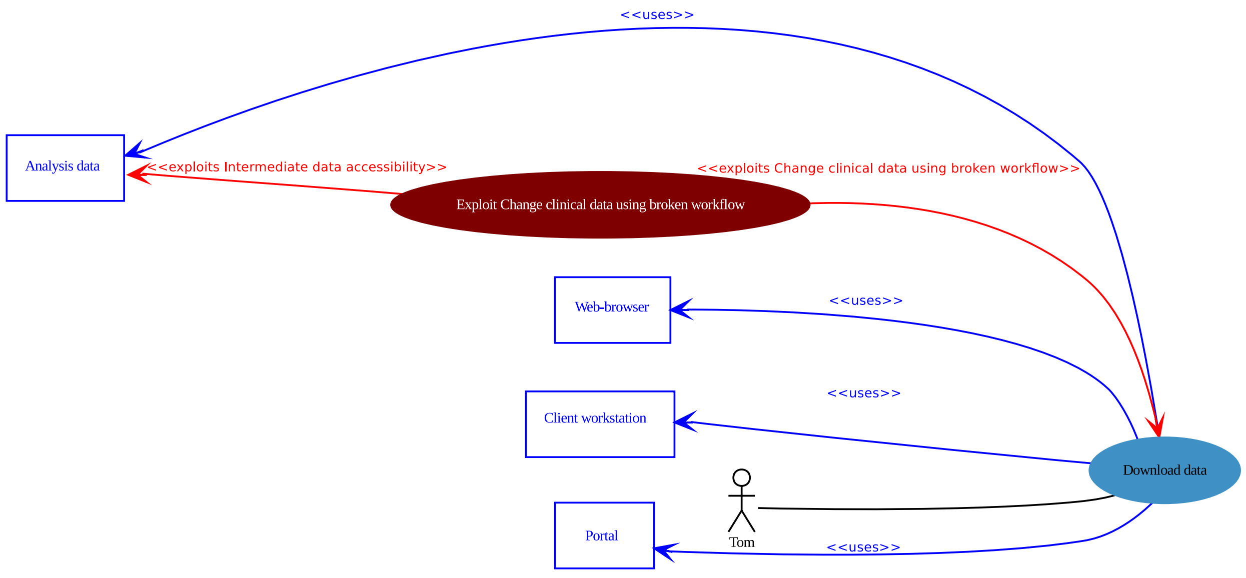
Figure 6.11 is an example of the zoomed-in asset model for the NeuroGrid Psychosis exemplar.

6.4.2.2 Task model

Personas and their task associations are represented using a modified form of UML use case diagram, where tasks are modelled as use cases, and personas are modelled as actors. This model also displays misuse cases and the attackers who realise them; attackers are displayed as actors wearing a black hat and misuse cases are represented as red ellipses with white text. The shades of red and blue used in the misuse case and task ellipses are based on the risk and task usability scores respectively; these colour shades are described in more detail in Section 6.4.2.4.

In the same manner that assets are associated with tasks, misuse cases are also, albeit indirectly, associated with assets. Each misuse case in CAIRIS is associated with a risk and, by extension, with a single threat and vulnerability. Consequently, if an asset is used by a task and also exposed by a vulnerability or endangered by a threat, then we can model this asset-misuse case relationship in CAIRIS. Because these associations potentially add to model clutter in a large task model, these associations are displayed based on the model's current zoom factor.

Figure 6.12 illustrates how these concepts come together in a task model using an example centred around the *download data* task in the Stroke exemplar. This zoomed-in model illustrates how one of the assets associated with this task is threatened by the *intermediate data accessibility* threat, which is



<<uses>>
Analysis data
<<exploits Intermediate data accessibility>>
<<exploits Change clinical data using broken workflow>>
Exploit Change clinical data using broken workflow
Web-browser
<<uses>>
Client workstation
<<uses>>
Download data
Portal
Tom
<<uses>>

Figure 6.12: Task model example

a component part of the *Change clinical data using broken workflow* risk; the risk is represented in this model via the related *Exploit Change clinical data using broken workflow* misuse case.

6.4.2.3 Goal and obstacle models

The goal-modelling notation used in IRIS is based on the KAOS modelling notation described in Section 5.4.5. The polygon KAOS model elements are comparatively trivial to render visually; this is an important property for tool-support, which needs to rapidly compute and visualise the products of analysis without hindering potentially participative design activities.

Figure 6.13 shows how the *Download analysis data* goal needs to be satisfied. Most of the parallelograms in this model are goals, but a small number of these are requirements. As indicated in Section 5.4.6, these are denoted by an asset or environment short-code, followed by a dash and a number, e.g. AD-1. Tasks operationalised by goals are also described in this model, as are obstacles which obstruct certain goals. In the diagram, the rhomboid labelled *Unauthorised portal access* indicates that unauthorised access to the NeuroGrid portal obstructs the goal stating that a user certificate is issued only if the holder has been authorised by a certificate authority.

IRIS also supports visualisation of obstacle models, allowing the refinement of an obstacle to a candidate threat or vulnerability to be explored. Figure 6.14 shows the refined obstacles causing the unauthorised portal access obstacle to be satisfied. It indicates that unauthorised access can be obtained either by controlling a web-browser (presumably a browser where the certificate has been installed), or by obtaining a user certificate which provides the necessary authorisation. The leaf obstacles in these

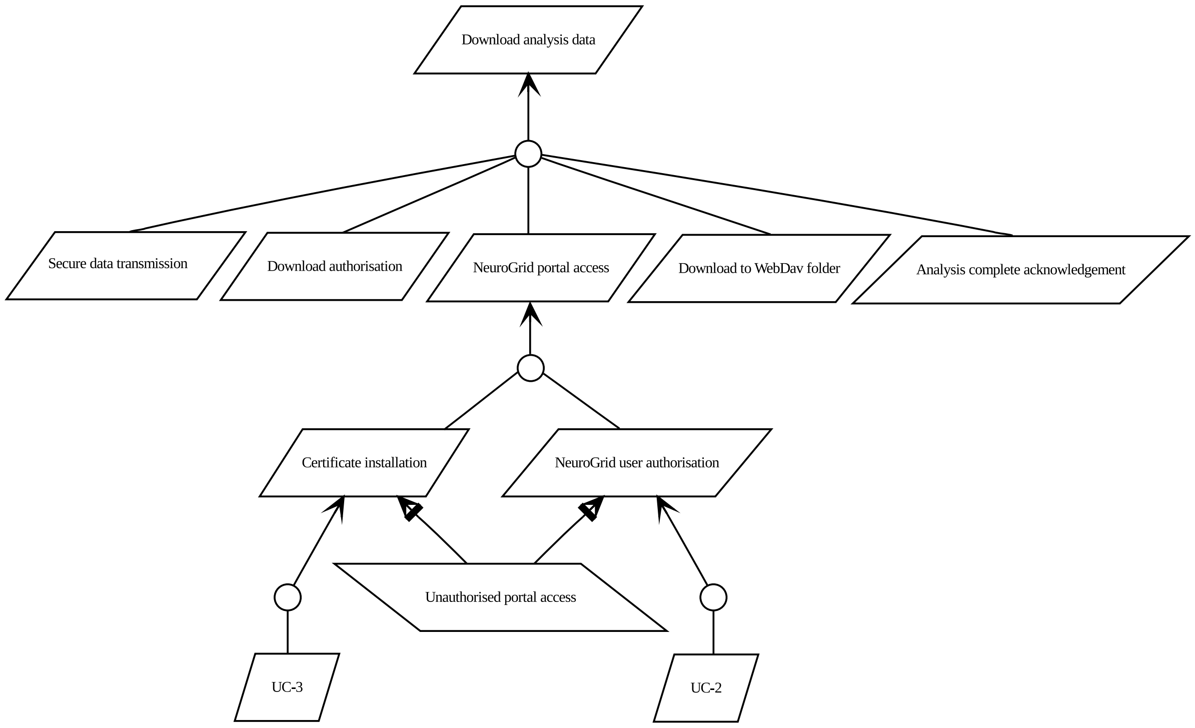


Figure 6.13: Goal model example

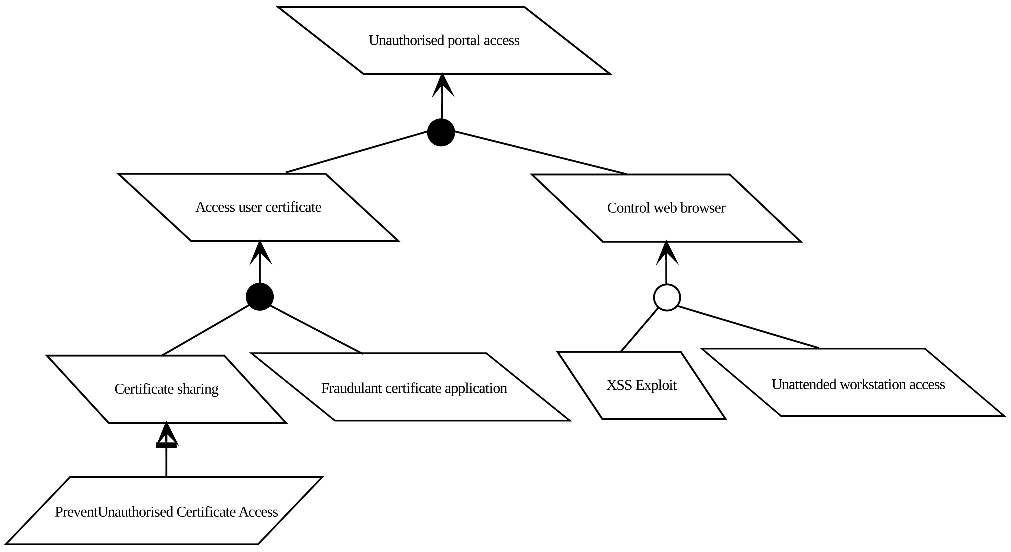

Unauthorised portal access
Access user certificate
Control web browser
Certificate sharing
Fraudulant certificate application
XSS Exploit
Unattended workstation access
PreventUnauthorised Certificate Access

Figure 6.14: Obstacle model example

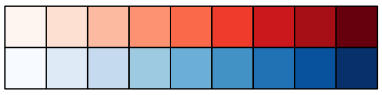


Figure 6.15: Risk (top, red) and task usability (bottom, blue) colour charts

cases are either candidate threats, e.g. XSS exploit, or vulnerabilities, e.g. certificate sharing. This model also indicates that a mitigating goal can resolve this latter obstacle.

6.4.2.4 Risk analysis model

The Risk Analysis Model provides a quick-look view of the current risk analysis. This model only displays the elements of risk analysis, and other non-risk analysis elements associated with them. This view also compresses goal trees arising from risk responses, thereby making it easier to trace risks to mitigating responses, the requirements they treat, and the countermeasures they refine.

Risk analysis model nodes are both colour coded and encoded with multidimensional data. As Figure 6.16 illustrates, information about security properties is coded within asset and threat elements. Histograms indicate whether or not values are held for each property and, if so, whether that property is low, medium, or high. The colours selected for the confidentiality, integrity, availability, and accountability histograms are the three primary colours — red, blue, and green — together with black; the use of black and primary colours provide the maximum differentiation between property types [249].

Risk analysis elements are colour coded with information, such as threat likelihood, vulnerability severity, and risk impact. Threats, vulnerabilities, and risks are also coloured based on their criticality: the more critical the element, the deeper the hue of red. The colours associated with task usability values are also based on criticality; the less usable a task is, the deeper the hue of blue. The colour charts associated with these values are shown in Figure 6.15.

Figure 6.16 displays the risk analysis model before and after the risk mitigation measures described in Section 6.4.1.3. The model illustrates the clear distinction between the shades of red in the risk node, but only a subtle distinction between the colour of the upload data ellipses. This indicates that, in this particular context, the countermeasure has a significant impact on mitigating the risk, but only a subtle impact on the related persona's activities in this task.

The risk analysis model also visualises metrics on requirements quality. These metrics are based on requirements completeness, the presence of an imperative phrase, and ambiguity. These are displayed using cartoon *Chernoff Faces* [49], and described in more detail by [263]. Eye-brow shape indicates the completeness of a given requirement. If no text is found in certain fields, or phrases like *TBC*, *none*, or *not defined* are present, the completeness score is marked down accordingly, and the eye-brows convey a

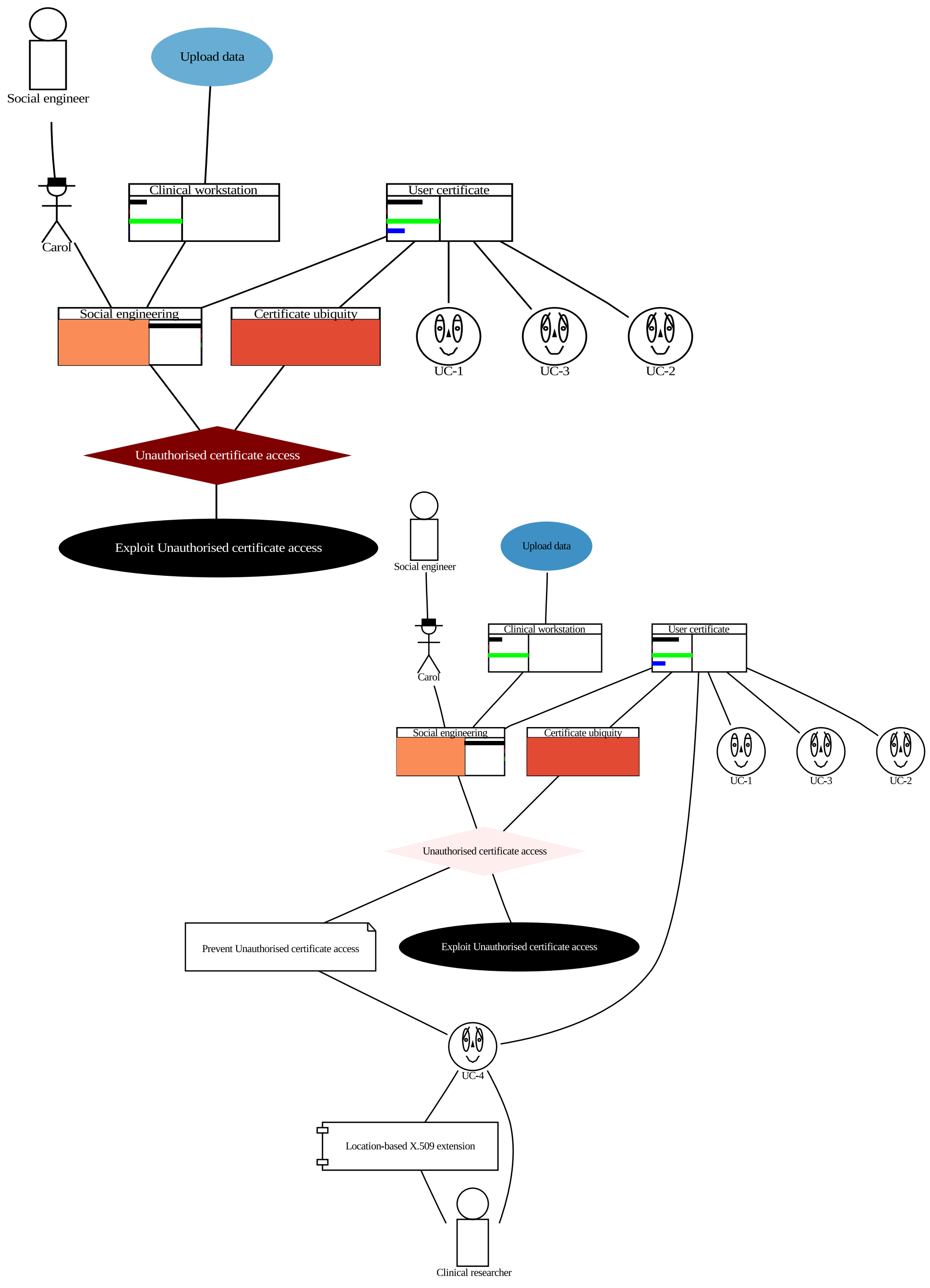


Figure 6.16: Risk analysis model before risk mitigation (top) and after (bottom)

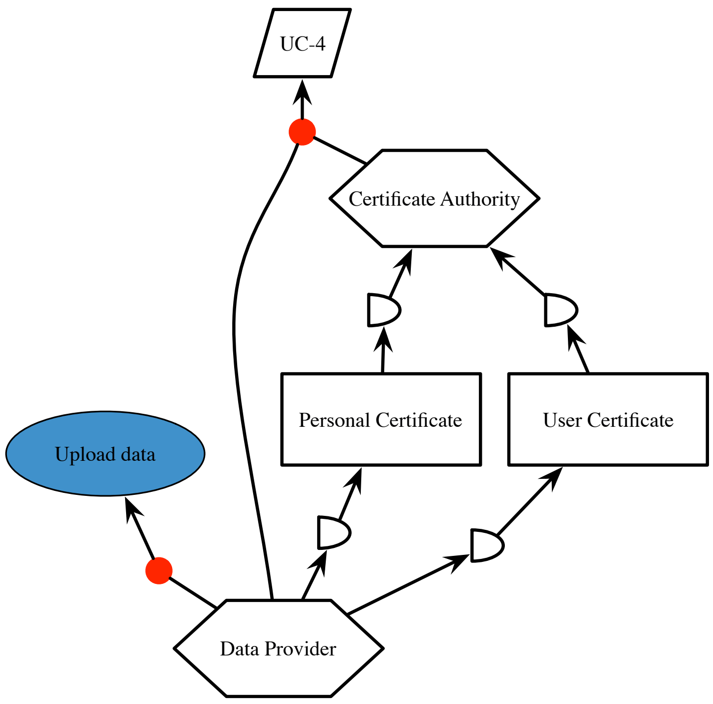


Figure 6.17: Responsibility model example

negative mood. The eye shape indicates whether or not an imperative phrase exists in the requirement description. If such a phrase exists then the eyes become vertically elongated. The mouth indicates the presence of weak or fuzzy phrases, such as *mostly*, *appropriate*, *normal*, or *adequate*; the presence of these phrases turn the smile into a frown.

In Figure 6.16, most of the requirements appear satisfactory, but attention is drawn to requirement UC-1 because of the eye-brow shape. Closer inspection of this requirement indicates that no rationale has been provided, making it only partially complete.

6.4.2.5 Responsibility model

Responsibility Models show roles that are directly or indirectly responsible for the satisfaction of goals or requirements. In Figure 6.17, we can see that users instantiating both the certificate authority and data consumer roles are responsible for the satisfaction of requirement UC-4; this requirement indicates user certificates shall be usable only from designated network addresses. This responsibility is shared because the data provider participates in a task affected by this countermeasure. Consequently, although the certificate authority is responsible for implementing this requirement, the data provider shares responsibility by conforming with the requirement. This responsibility association also indicates the roles are responsible for carrying out particular tasks. In this example, personas fulfilling the data provider

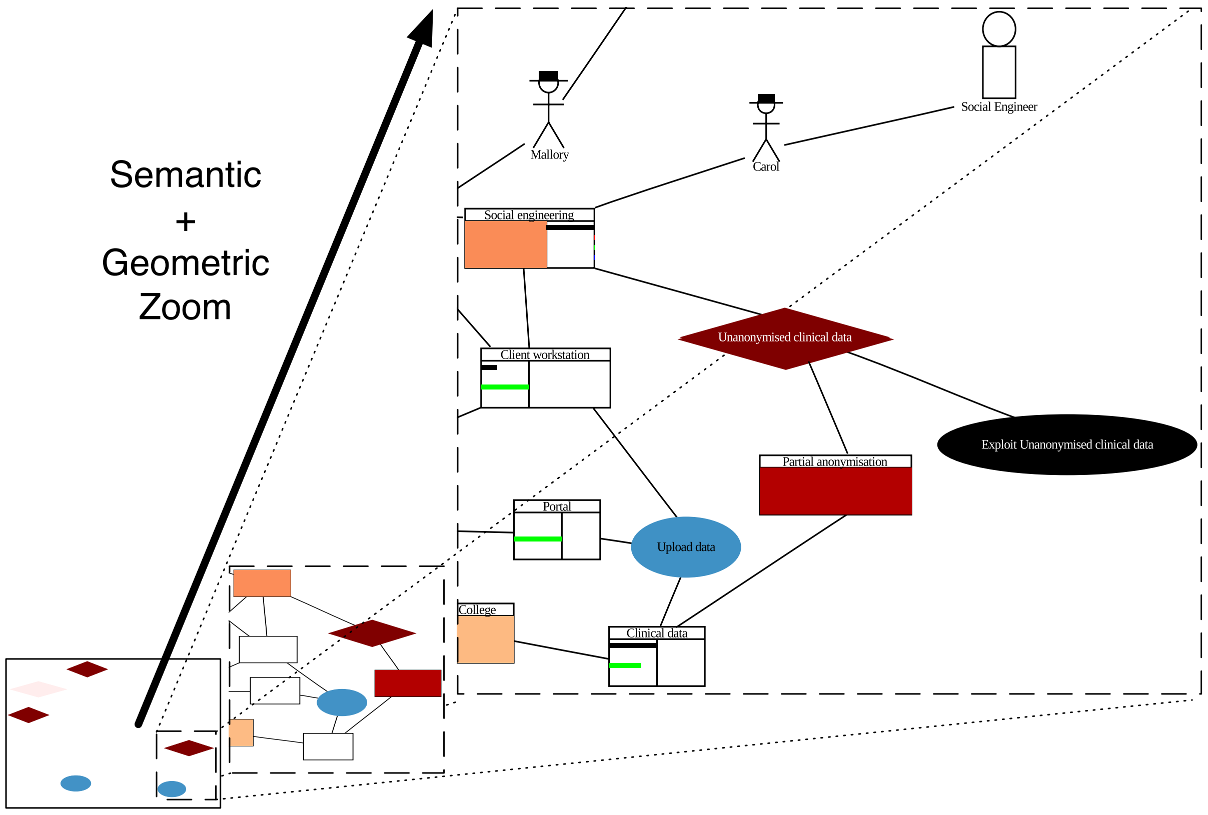


Figure 6.18: Semantic and geometric zooming in risk analysis models

role carry out the upload task.

This model also shows how roles are dependent on other roles. In this example, we can see that while data providers are responsible for access control policy decisions, they also rely on certificate authorities for their digital certificates.

6.4.2.6 Clutter management

With so much information associated with the different model views, visual clutter can become a problem as models grow.

Minimal distinctions in colour can be used to reduce visual clutter, and small contrasts enrich the visual signal increasing the number of possible distinctions [250]. To take advantage of this, we map the normalised values for R_r and SU_t to the respective risk (red) and task usability (blue) colour charts. The higher the risk or task usability score, the deeper the hue of red or blue. Threat likelihood and vulnerability severity scores map to a colour chart similar to that of risk. An example of how these colours are applied to elements on the IRIS risk analysis model is provided in Figure 6.16.

CAIRIS also uses geometric and semantic zooming to make efficient use of the available viewing area as model data increases. Geometric zooming magnifies detail at the cost of loss of context; semantic zooming conveys additional model information as the model is zoomed in [229]. As Figure 6.18 illustrates, the risk analysis model supports three levels of zooming. At the lowest zoom factor, only the most salient elements are displayed. As the zoom factor increases, further elements are displayed, together with the associations between them. At the highest zoom factor, all remaining elements are displayed, together with additional information about assets and threats. At this level of granularity, textual labels are also displayed; at lower levels, such detail would be unreadable and distracting. Zooming is also supported in the task model. Associations between personas and attackers, and tasks and misuse cases respectively are displayed at low zoom factors. Associations indicating that a misuse case threatens a task, or a task mitigates a misuse case, are displayed at medium zoom factors. Associations indicating assets used by a task, exploited or threatened by a misuse case, or assets mitigating a misuse case are displayed at high zoom factors.

Even with support for zooming, clutter remains a problem in large models when viewed at a high zoom factor. Consequently, filtering is also supported in each model. Models can be filtered by task or misuse case in the task model, and by node name and type, i.e. risk, threat, vulnerability, etc, in the risk analysis model, and by object name in the other models. Moreover, each model can also be projected according to different environments. Filtering by object name is particularly useful in large goal and obstacles models; when the filter is applied, the goal tree is re-displayed such that the filtered node becomes the root node.

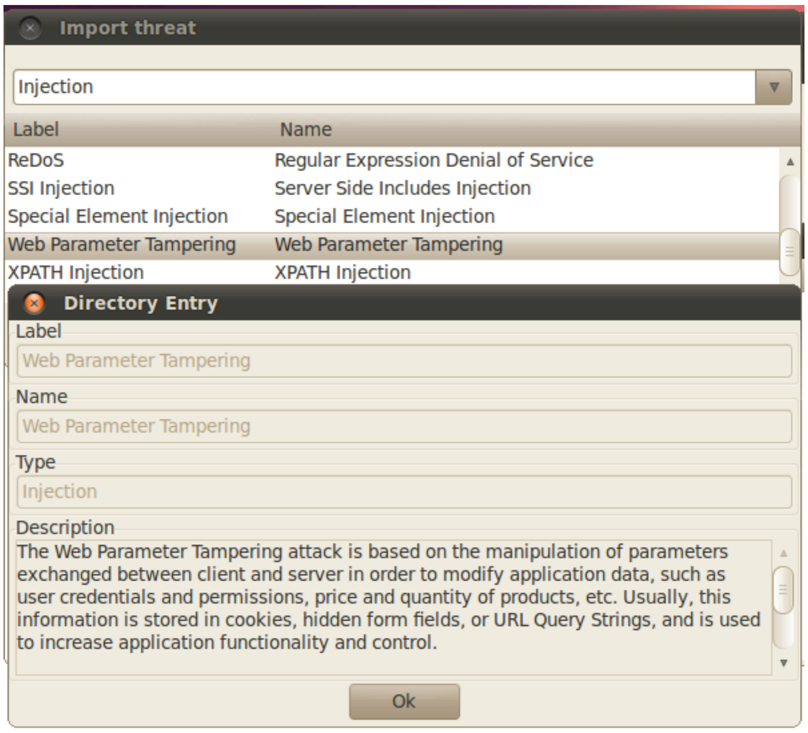
6.4.3 Model reusability

By building upon the IRIS meta-model, expectations can be made about the layout of CAIRIS models. This provides opportunities for incorporating complementary analysis into CAIRIS.

One of the explicit assumptions made by security design methods is the presence of a security expert [70, 97]. These stakeholders act as consultants to a design team and are expected to provide insight into threats, vulnerabilities, and possible mitigation measures. Unfortunately, security experts may not be available when analysis is carried out. It is, therefore, useful to encapsulate knowledge about possible problems and possibilities into the tool.

CAIRIS supports *directories* of re-usable threats and vulnerabilities. These directories are ASCII files marked up in XML according to the schema shown in Figure 6.20, and can be created from known sources of threat and vulnerability data. During this research, directories were created to encapsulate the attacks and vulnerabilities collected as part of the Open Web Application Security Project (OWASP) [17] and a Common Criteria Protection Profile for Industrial Control Systems [187].

A suitably marked up directory can be imported directly into CAIRIS from its GUI. Imported elements can be used by clicking on the Import button in the dialog boxes listing threats and vulnerabilities



Label	Name
ReDoS	Regular Expression Denial of Service
SSI Injection	Server Side Includes Injection
Special Element Injection	Special Element Injection
Web Parameter Tampering	Web Parameter Tampering
XPATH Injection	XPATH Injection

Label: Web Parameter Tampering

Name: Web Parameter Tampering

Type: Injection

Description: The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control.

Figure 6.19: Importing a template OWASP threat into the current model

respectively. As Figure 6.19 shows, these importable elements are grouped by a particular class of threat or vulnerability. When imported, a threat or vulnerability dialog box is opened, and the template is used to fill all of the non-contextual fields. It remains the responsibility of the analyst to associate these to an environment, together the threatened or vulnerable assets, and likelihood or severity of the respective threat or vulnerability.

6.4.4 Model externalisation

As Section 6.1.3 alludes to, not all stakeholders in an IRIS process are intimately associated with the analysis being carried out. Nevertheless, it is important that these are considered by tool-support in two ways. First, they need to be aware of the general context within which IRIS is being used, and the scope of analysis that delimits the activities being carried out. Second, the on-going model needs to be made available to them in a readable format.

As Section 5.4.6 discusses, the Volere Requirements Specification template forms the basis of documenting specifications resulting from IRIS. With this in mind, the project meta-data supported by CAIRIS conforms to the initial sections of this template. This meta-data can be managed using the project settings dialog shown in Figure 6.21.

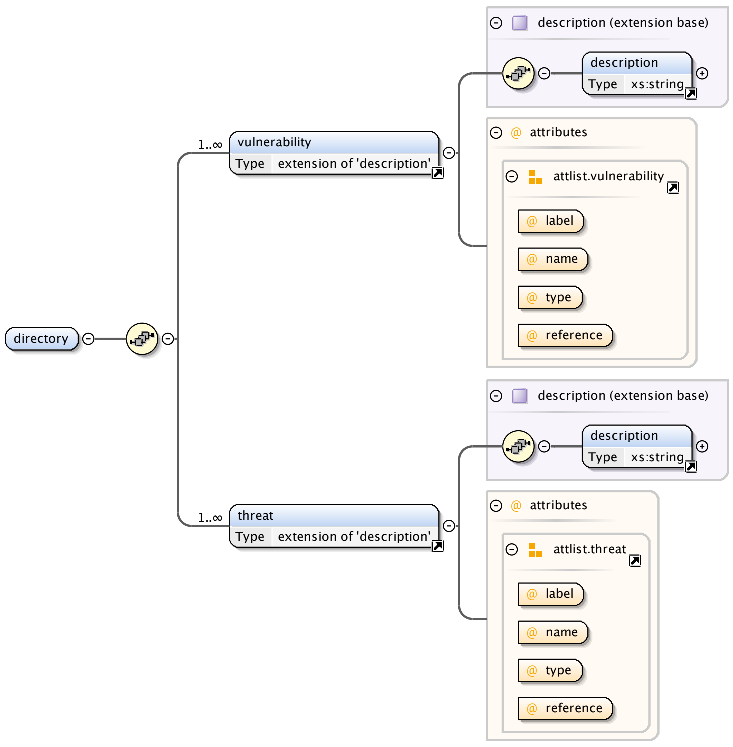
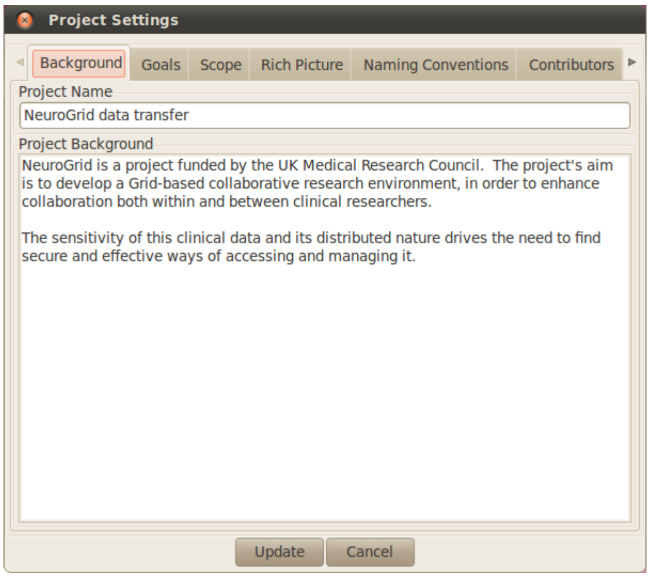


Figure 6.20: Threat and vulnerability directory XML schema



Project Name

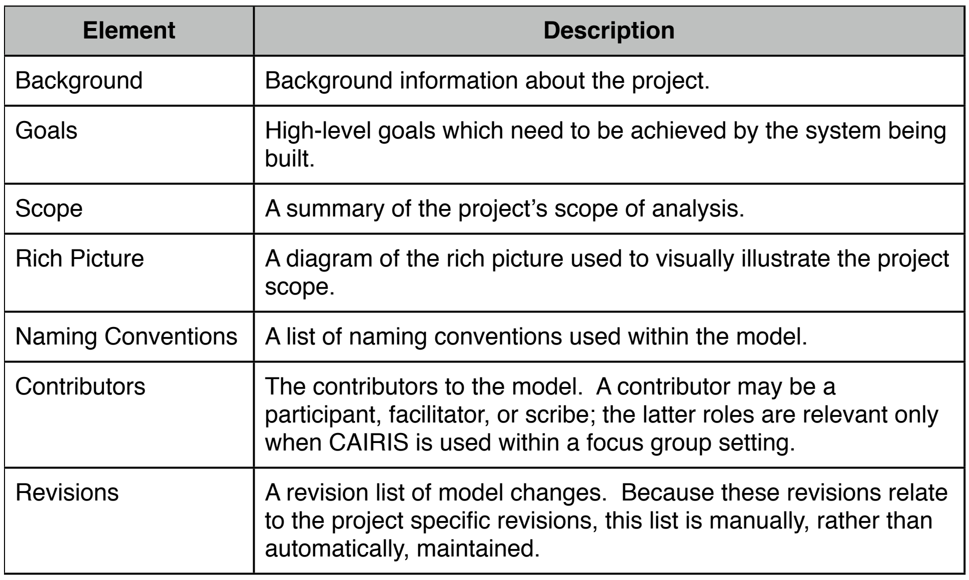
NeuroGrid data transfer

Project Background

NeuroGrid is a project funded by the UK Medical Research Council. The project's aim is to develop a Grid-based collaborative research environment, in order to enhance collaboration both within and between clinical researchers.

The sensitivity of this clinical data and its distributed nature drives the need to find secure and effective ways of accessing and managing it.

Figure 6.21: Project meta-data stored by CAIRIS



Element	Description
Background	Background information about the project.
Goals	High-level goals which need to be achieved by the system being built.
Scope	A summary of the project's scope of analysis.
Rich Picture	A diagram of the rich picture used to visually illustrate the project scope.
Naming Conventions	A list of naming conventions used within the model.
Contributors	The contributors to the model. A contributor may be a participant, facilitator, or scribe; the latter roles are relevant only when CAIRIS is used within a focus group setting.
Revisions	A revision list of model changes. Because these revisions relate to the project specific revisions, this list is manually, rather than automatically, maintained.

Figure 6.22: Project meta-data supported by CAIRIS

Generating readable documentation from CAIRIS involves a two-step process. The first step involves rendering the CAIRIS model as XML conforming to the DocBook standard [75]. The chapters of this document conform to the Volere structure described in Section 5.4.6. The PDF versions of the various graphical views within the model are also created at this stage. The second step involves taking the generated, machine-readable DocBook file, together with the various graphic files for the visual models as input, and creating a human readable output format; the formats supported by CAIRIS are HTML, RTF, and PDF.

6.5 Chapter summary

Based on work in the previous chapters, this chapter has presented four design principles for tool-support needed to support IRIS: Familiarity, Extensibility, Process Centricity, and Security and Usability Centricity.

To explore the characteristics needed to embody these principles, we presented the CAIRIS software tool. We described the tool's development process, and presented its high level architecture, before proposing four characteristics of CAIRIS embodying the previously presented design principles: Automated Analysis, Model Visualisation, Security Model Reusability, and Model Externalisation.

In the next three chapters, we will consider CAIRIS' conformance with the Extensibility principle in more detail, by considering the tool's evolution through the various Action Research interventions.

Chapter 7

Case Study: Requirements for a Control Software Repository

This chapter reports the results of a case study where IRIS was used to specify requirements for a software repository managing control software used by a UK water company.

Using the Action Research methodology described in Section 3.2.1.2, we describe the characteristics of the context of intervention. We then describe how this motivates the selection of techniques for an IRIS process for eliciting and specifying the repository requirements, before presenting an action plan for applying these techniques in three separate design phases: Scoping, Fieldwork, and Participatory Design.

Based on the action plan, we describe the results of each of the above design phases, before evaluating the intervention. We evaluate these results by discussing the efficacy of each technique within the context of the applied IRIS process.

We conclude this study by specifying lessons which re-inform IRIS and the design of the subsequent case studies.

7.1 Research environment

The author was invited by the information security manager of a large UK water company, hereafter known as ACME, to help specify requirements for a software repository; this was specified for storing and managing access to the company's instrumentation control software. ACME was responsible for providing clean and waste water services to several million people in a particular UK region. The infrastructure needed to support such a large customer base was substantial, amounting to over 60 water treatment works, 800 waste water treatment works, 550 service reservoirs, 27,000 km of water mains, 19,000 km of sewer networks, with over 1,900 pumping stations, and 3,200 combined sewer outflows.

The ACME information security manager acted as the gate-keeper for the intervention; it was through him that all human resource access requests were made. Similarly, the information security manager was responsible for inviting participants to the scoping workshop, and the design sessions described in Section 7.4.3.

At the beginning of each interview or session, permission to make audio and, occasionally, video recordings was sought. It was agreed that transcripts from these sessions would not be made available to ACME, and would be used only by the University of Oxford for research purposes.

7.2 Diagnosis

Based on the initial brief, three interviews were held with stakeholders to establish the main factors likely to guide the design of the intervention. The first of these interviews was an on-site background interview with a manager with previous experience of managing control equipment. The remaining were phone interviews with an ACME ICT support officer, and ACME's information security manager.

Based on these discussions, the following factors were identified as likely to have an impact on the planned intervention.

7.2.1 Geographic and organisational dispersal

Although ACME was accountable to government regulators for providing water services and management of their infrastructure, responsibility for different aspects of its business was outsourced to different organisations. Responsibility for water treatment services for different geographical areas and types of water treatment was outsourced to two operational partner companies. This responsibility included maintenance of the infrastructure, together with the technology and human resources necessary to support its operation. Consequently, the maintainers of this infrastructure worked for one of these operational partners, rather than ACME itself.

The ICT infrastructure for ACME's commercial operations was also outsourced to two different ICT service providers. Although the infrastructure used in ACME offices and the corporate network between different ACME locations was managed by a central ICT team and its service providers, this did not include the ICT infrastructure used in the treatment works and sites maintained by the operational partners.

7.2.2 Instrumentation and instrument technicians

Maintaining the instrumentation for the colossal amount of hardware associated with clean and waste water treatment was the responsibility of *Instrument Technicians*. Instrument technicians covered specific geographical areas, and provided working and out-of-hours cover for the equipment within their area of operations. Examples of the equipment these technicians supported included the following:

- Telemetry outstations: these were devices attached to specific equipment, such as water pumps. These devices were responsible for sending messages back to a central control station in the event of equipment anomalies.
- Programmable Logic Controllers (PLCs): these devices controlled the processes associated with a piece of equipment, or part of a plant. In many cases, these were associated with a panel that a plant operator used to control the settings of some aspect of the treatment process.
- Supervisory Control and Data Acquisition (SCADA) workstations, which monitored the operations of a particular plant.

Instrument technicians carried out their work either as part of a planned maintenance plan, or as an out-of-hours call-out. In these latter cases, instrument technicians needed to drive, at short notice, to potentially remote parts of their allotted patch in variable weather conditions. Consequently, technicians needed to carry out essential repairs to critical equipment under both physical and mental duress.

As Section 7.2.1 suggests, PCs used to support SCADA and other non-corporate ICT applications were also maintained by instrument technicians. In many cases, instrument technicians provided a form of informal IT support for these machines, despite the lack of any formal training on how such support should be provided.

7.2.3 Legacy equipment

ACME was an amalgamation of several different water companies, each of which were themselves an amalgamation of even smaller companies. Each of these separate companies adopted their own approach for SCADA and control systems and, as a result, the control systems estate was both large and eclectic. Moreover, the level of support each system enjoyed was variable. Some control systems ran on recent PCs, and were maintained under active support contracts. Unfortunately, many systems were no longer supported, relied on legacy PC hardware, and maintenance knowledge was restricted to a small number of engineers and contractors.

Because of these legacy issues, ACME instigated a long-term automation strategy; this centralised control of all assets, and collection of data about them. Implementing the strategy entailed deploying a number of centralised SCADA operations centres, together with a new software and hardware infrastructure from a single manufacturer at all treatment works to monitor and control their operation.

7.2.4 Software ubiquity

Making changes to instrumentation invariably involved making software changes. Modifications included alarm setting changes within telemetry outstations, new or altered control logic in PLCs controlling pumps or other water treatment equipment, and panel changes for SCADA workstations or panel PCs located around a large works.

Instrument technicians usually backed up the current version of working software before and after any changes, but the nature of this backup varied by location and type of software. For example, backups to telemetry outstation software were stored on floppy disks next to the outstation, but if this outstation was in the countryside then the physical media was vulnerable if the kiosk became damaged and exposed to the elements. Compromised backup disks were a frequent enough occurrence that technicians kept backups on their own laptops, but accidental or deliberate problems could occur as a result of this practice. For example, due to exhaustion, instrument technicians occasionally forgot to back-up software changes, or sometimes updated a device with an incorrect software version.

Human frailty was also a defence given by malicious inside attackers. One of the most cited examples of this was the attack on the sewage system at Maroochy Shire in Queensland, Australia. In revenge for being rejected for a job at the local council, Vitek Boden hacked into the sewage control system he helped developed and, over the course of several weeks, instigated software changes leading to the release of millions of litres of raw sewage into the local ecosystem [226]. While such an attack may sound far fetched, the migration to the new control systems infrastructure discussed in Section 7.2.3 led some technicians to question what their future role might be at ACME.

7.2.5 Information security

The Information Security team at ACME were responsible for the information security of all operations, including the ICT infrastructure. Although their responsibility did not formally extend to the security of water treatment services, the team frequently dealt with requests for allowing excessive access privileges to Instrument Technician laptops; these were used to develop and, sometimes, apply software changes to control equipment. Consequently, an information security breach may have indirectly led to the compromise of water services.

Because of the impact information security had on the security of critical national infrastructure, the UK Centre for the Protection of National Infrastructure (CPNI) became increasingly engaged with water companies like ACME, to appraise them of the risks of terrorism and cyber-warfare. These concerns were becoming heightened as control systems became increasingly distributed. Traditionally, the control system for a water treatment works was, in network terms, an island; compromising the network would only compromise a specific site. However, the move to the distributed infrastructure mentioned in Section 7.2.4 heightened concerns about possible external attacks. Such concerns also

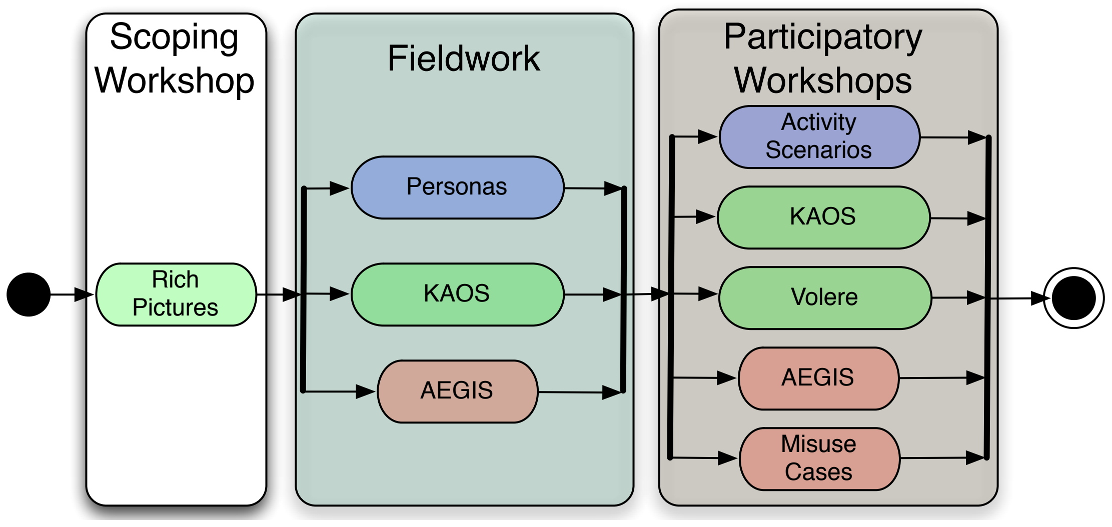


Figure 7.1: Instantiated process for the software repository specification

affected the software repository.

7.3 Action planning

Based on the factors described in the diagnosis, the process for eliciting and specifying requirements needed to incorporate the following characteristics:

- A clear scope of analysis, agreed as early as possible.
- An understanding of how the software repository could support, rather than obstruct, the work of instrument technicians.
- Ownership of the specification by the stakeholders responsible for using and supporting the software repository.

Using the IRIS framework, we devised the process illustrated by the UML activity diagram in Figure 7.1 for eliciting and specifying the software repository's requirements. We dealt with the above characteristics by applying the process in three phases, which we describe in the following sections.

7.3.1 Scoping

The scoping phase involves holding a workshop to agree the scope of the software repository. The objective of this workshop is to present the process to be adopted, develop a rich picture diagram delimiting the scope of the system, and identify the key roles and environments the system needs to be specified for.

After the workshop, a draft schedule for collecting empirical data about these key roles in their environments is agreed.

7.3.2 Fieldwork

The fieldwork phase involves carrying out three concurrent activities: persona development, initial goal modelling, and initial asset modelling.

Persona development entails carrying out in-situ interviews to collect empirical data about the key roles and the work carried out by them. Audio transcripts of these interviews is taken and analysed using Grounded Theory. The transcripts are coded to induce a model of salient grounded concepts and their relationships. After reviewing this model and discussing these salient concepts with colleagues, the model is re-evaluated, and concepts re-coded and clustered around an emerging set of themes. This process is analogous to Affinity Diagramming [32], although ATLAS.ti is used to manage concepts and help identify underlying patterns, rather than sticky notes. Based on these concepts, a persona narrative is written appealing to the concepts identified. This narrative was structured according to the behavioural variable types recommended by [61].

As empirical data is collected about the different contexts of use, candidate goals and requirements which need to be satisfied are elicited and modelled as goal trees. These are only candidate goals as they need to be validated by stakeholders during the next stage. At the same time, assets are identified from this data, together with possible vulnerabilities and threats.

7.3.3 Participatory design

Although some goals and requirements are elicited in the previous stage, the vast majority are elicited during participatory design workshops. These workshops are facilitated by the researcher, and supported by the CAIRIS tool. A laptop with CAIRIS installed is connected to a projector, so the tool and its usage is visible to all participants. These participants represent a cross-section of software repository stakeholders, including instrument technicians, software engineers, information security officers, and system administrators. At the beginning of each workshop, the developed personas are introduced in a short presentation where the salient characteristics are each are presented using a bullet point list.

Each workshop involves asset and goal modelling to determine the software repository requirements. This is supported by the development of activity scenarios to describe the impact of satisfying software repository requirements.

To explore the impact of security on the system being specified, obstacles are also elicited and progressively refined to identify the threats or vulnerabilities giving rise to these conditions. Where

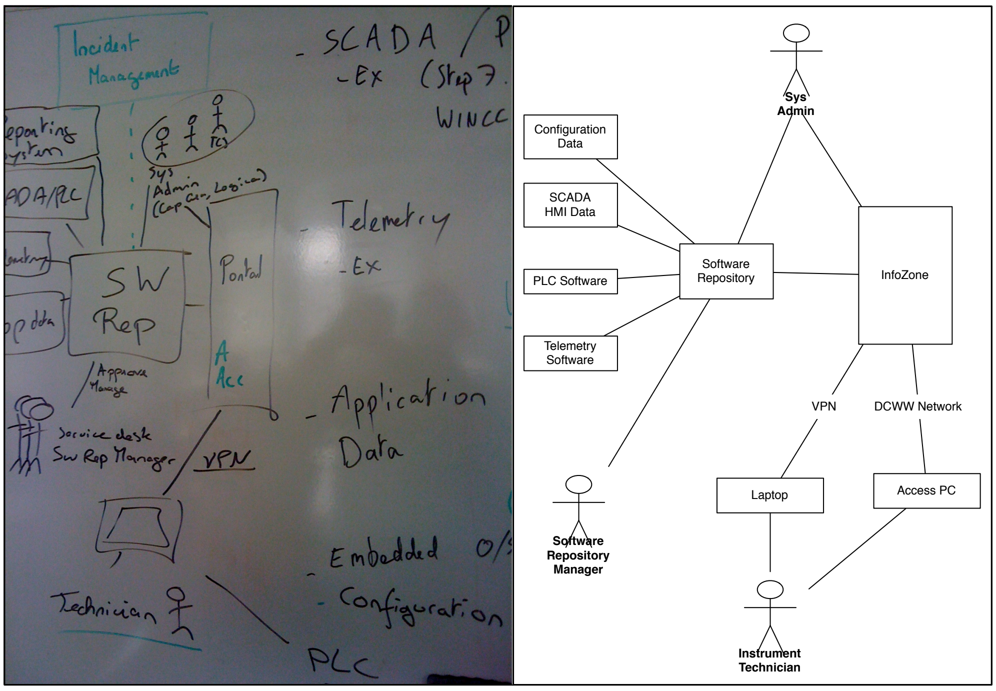


Figure 7.2: Original rich picture (left) and reproduced version (right)

appropriate, risks are defined based on these, and possible responses and countermeasures are elicited.

Following each workshop, a Volere specification is generated by CAIRIS, and emailed to all workshop participants for comment.

7.4 Action taking

We now describe how the steps in the action plan were carried out as part of the intervention.

7.4.1 Scoping

A half-day scoping workshop was held at ACME's head office on 5th November 2009, and attended by 17 different stakeholders from ACME and their operational partners. These stakeholders were members of ACME's SCADA telemetry security group, and included clean and waste water treatment plant managers, system integrators responsible for delivering control software as part of capital asset projects, telemetry technicians, ICT support staff, and the ACME information security manager.

Two researchers were present during the workshop. One researcher acted as the facilitator and, using the white board as a focal point of reference, drew rich pictures to describe the scope of the prospective repository. The other researcher (the author) observed the discussion, interrupting periodically to clarify

Date	Location	Roles observed	Observations
13/11/09	Countryside around ACME head office	Instrument Technician	Telemetry outstation modification procedures
13/11/09	ACME head office	ex-Instrument Technician	Modification of PLC and SCADA software
24/11/09	An ACME Waste Water Treatment Plant	Instrument Technician	Modification of PLC software
26/11/09	An ACME Clean Water Treatment Plant	ex-Instrument Technician	Telemetry outstation modification procedures

Figure 7.3: Fieldwork summary

the meaning of certain points of debate. An audio and video recording of this workshop was taken.

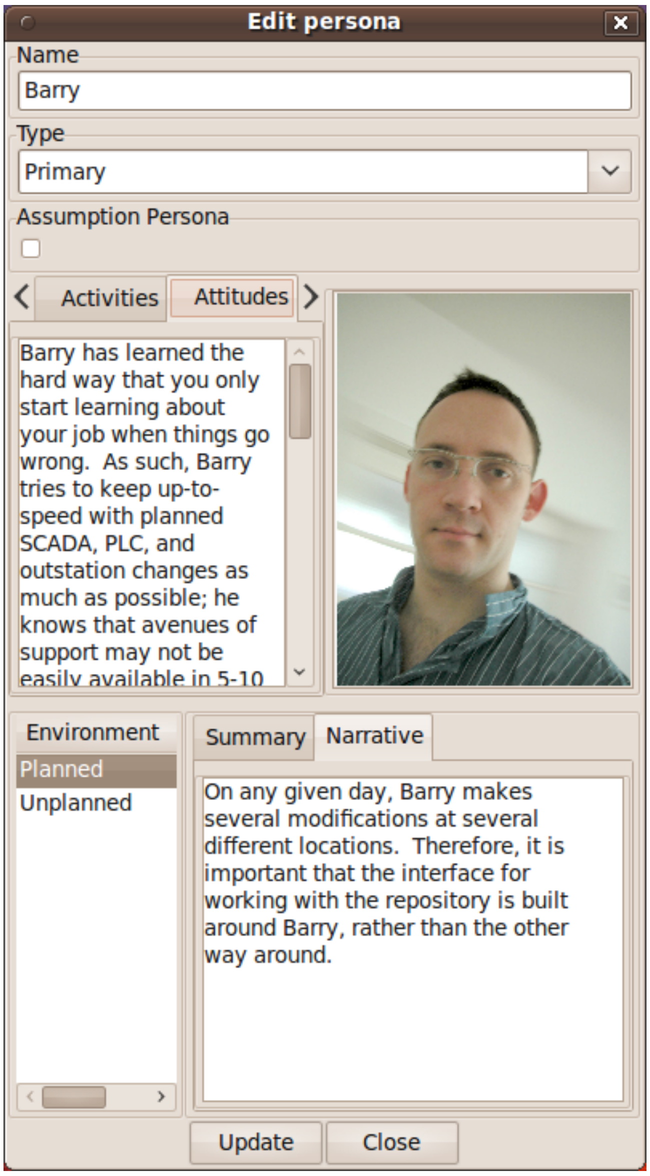
By the end of the workshop, the rich picture in Figure 7.2 (left) was agreed; this was subsequently re-produced to clarify the important concepts in Figure 7.2 (right). It was also agreed the key role for the repository would be the instrument technician. It was, however, unclear at that stage what the most relevant environments for further analysis might be. It was agreed that these would be decided and agreed upon by the end of the fieldwork phase.

Following the workshop, tentative arrangements were made by ACME's information security manager for a researcher to visit a number of sites to shadow different instrument technicians.

7.4.2 Fieldwork

As Figure 7.3 shows, several in-situ interviews were held with instrument technicians and related stakeholders during November 2009; these interviews were held at their physical work locations. On each occasion, interviewees were asked to discuss and, where possible, demonstrate, how they modified control software as part of their work, and how they currently dealt with the backup and configuration control of different types of software. During these interviews, a number of other work-related issues were identified and discussed.

Audio transcripts were taken during each interview, and subsequently analysed as described in Section 5.4.1. From the various concepts and their grounding, it became apparent that a single persona could not be derived which treated the most grounded, salient concepts. As a result, two different personas were elicited. The first persona (Barry) modelled an instrument technician who carried out software modifications as part of his day-to-day work maintaining instrumentation. The second persona (Alan) modelled a commissioning engineer who works off-site but was responsible for the initial installation of control software to plant equipment. The complete persona structure was entered into CAIRIS and, as Figure 7.4 indicates, a representative photo was associated with each persona.



Name

Barry

Type

Primary

Assumption Persona

☐

Barry has learned the hard way that you only start learning about your job when things go wrong. As such, Barry tries to keep up-to-speed with planned SCADA, PLC, and outstation changes as much as possible; he knows that avenues of support may not be easily available in 5-10

Planned

Unplanned

On any given day, Barry makes several modifications at several different locations. Therefore, it is important that the interface for working with the repository is built around Barry, rather than the other way around.

Figure 7.4: Persona details for Barry

Figure 7.5: Final focus group

During this analysis, it also became apparent that prevalent contexts of interest to the various stakeholders were not based on different types of software, but on whether instrument technicians carried out work as part of a planned or unplanned change. Consequently, details about these two different environments were entered into CAIRIS.

Once the personas were developed, a number of high-level goals and sub-goals were elicited from the empirical data; these goals related to the download and upload of telemetry software. A small number of assets were also elicited, based on artifacts observed in several of the interviews, e.g. laptops, access PCs at treatment works, and SCADA / telemetry / PLC software. This data was also entered into CAIRIS.

7.4.3 Participatory design

Three one-day focus groups were held at ACME's head-office on the 17th December 2009, 11th January 2010, and 12th January 2010 respectively. In addition to a researcher (the author), each session was attended by between 4–5 stakeholders from the SCADA telemetry security group. This group contained a sub-set of stakeholders from the initial scoping workshop. The final design session (Figure 7.5) also included two invited instrument technicians.

Prior to each session, a Volere requirements specification was generated by CAIRIS and emailed to the ACME information security manager for distribution to all attendees. Before the start of the first design session, CAIRIS was also loaded with a number of template threats and vulnerabilities from a Common Criteria protection profile for Industrial Control Systems [187], in the event that these might be needed to stimulate ideas for possible vulnerabilities and threats.

Requirements were elicited during the workshops in a number of different ways. High-level system goals were progressively refined to requirements, which were operationalised by tasks. Participants specified several tasks, which described how personas would tackle the job of maintaining different classes of control software as part of their day-to-day work. After specifying each task, participants would categorise the usability of the task by categorising the amount of time it took a persona to complete the task, how often a persona would carry out the task, how demanding a persona found the task, and how closely the task matched a persona's own work and personal goals, as described in Chapter 6.

Even with the contextual support personas provided, participants occasionally found it difficult to determine how goals could be further refined. Personas were used to kick-start analysis using two simple techniques. The first technique involved simply asking participants how the personas would approach a situation; subsequent discussion led to requirements being elicited or further knowledge being elicited to augment other analysis. The second technique involved taking a bottom-up approach to elicitation. This involved participants describing the tasks carried out by personas, after which glossed over assumptions would be explored in more detail. Based on this discussion, assumptions were explicated and modelled as goals or requirements needing to be satisfied.

Figure 7.6 illustrates a fragment of the goal model for the Planned environment. This shows the goals associated with maintaining telemetry software, as well as the goals needing to hold for the modify telemetry software task to be satisfied.

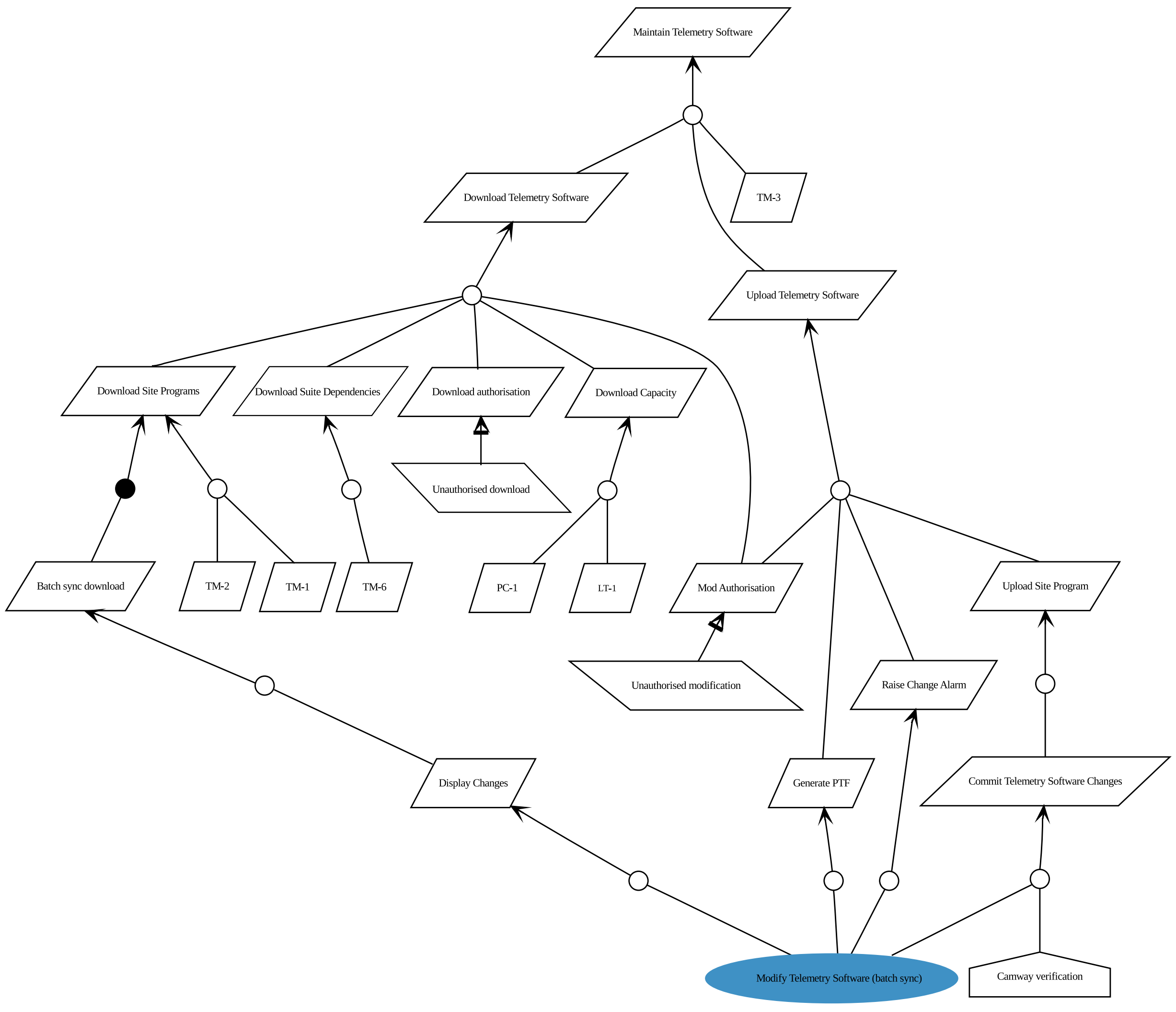


Figure 7.6: Goal model for maintaining telemetry software

7.5 Evaluating

Concept	Planned	Unplanned	ALL
Asset	9	7	9
Attacker	4	4	4
Countermeasure	0	0	0
Domain Property	N/A	N/A	8
Goal	31	31	32
Obstacle	16	16	16
Persona	3	3	3
Requirement	N/A	N/A	22
Response	0	0	0
Risk	4	0	4
Role	N/A	N/A	7
Task	4	3	4
Threat	8	0	8
Vulnerability	9	7	9
ALL	88	71	126

Figure 7.7: IRIS concepts specified on completion of the ACME software repository intervention

Following the final participatory design workshop, a Volere requirements specification was generated from CAIRIS and emailed to the ACME information security manager. A final, wrap-up meeting summarising the results and discussing the intervention was held at ACME's head-office on 23rd March 2010. This specification was used to inform the selection of a commercial software tool for the repository, a pilot of which was rolled out in September 2010.

We believe the outcome of the intervention from a research perspective successfully validated the IRIS process used for this case study. One of the testaments of this validity is, despite the constant interplay between different perspectives, the participants found the process natural. One participant, an information security officer, noted that switching between different techniques was something she would do as part of her everyday risk analysis activities. Another participant, a telemetry support technician, commented that championing the cause of his user community was something he often strived to do, so the IRIS process' position of synthesising usability, security, and requirements concepts was welcomed.

Figure 7.7 summarises the concepts elicited and specified on completion of the intervention.

As the table shows, concepts were present in either one or both of the *Planned* and *Unplanned* environments. In general, most concepts were present in both environments, although some were present only in the Planned environment; these related to auditing activities or other activities that required access to the instrument technician's depot during working hours. An example of the latter was the *written down login details* vulnerability; this stated that login details were sometimes written down and left near access PCs at depots or other office locations.

The table also indicates that while several risks were elicited, no responses or countermeasures mitigating these were identified. After discussing these risks, it appeared that mitigating strategies for resolving these were beyond the scope of the project. ACME, therefore, decided to implicitly accept these risks.

In the following sections, we evaluate the role played by each of the process techniques in more detail.

7.5.1 Rich pictures

The rich and varied nature of the problem domain described during the diagnosis suggested that rich pictures would be an ideal technique for exploring the problem domain. In hindsight, however, the act of getting people together to talk about software modification problems led to such a lively debate that the diagram became more useful as a tool for agreeing scope than it was for exploring the problem space. Nevertheless, as a scoping tool, the rich pictures technique proved to be useful. Before any pictures were drawn, discussion revolved around the challenges and difficulties associated with managing control software changes. By the end of the workshop, however, participants were left with a feeling that not only had a scope for a system to deal with these problems been agreed, but building such a system was achievable.

Some of the discussion around the rich picture also informed aspects of the Volere specification; this is described in more detail in Section 7.5.4.

7.5.2 Personas

Personas proved to be one of the defining techniques in the process for a number of reasons.

First, personas played an important role in humanising design decisions to workshop participants who, previously, had been unaware of the work carried out by users the personas represented. Although the information security officers and IT support teams had visibility of their organisation's IT infrastructure, the issues relating to systems associated with plant and water treatment instrumentation were hitherto unknown. The only contact IT staff had with instrument technicians arose from responding to requests for enhanced access privileges on laptops, which allowed the successful execution of certain software applications. Similarly, because instrument technicians were largely responsible for supporting their own hardware and software, collaboration with IT staff was rare. Maintaining the software repository would, however, become the responsibility of IT support teams, so the personas played an important part in understanding the characteristics of a role they had not previously been exposed to.

Second, personas provided insights into the sort of characteristics an insider attacker might have. One threat identified involved an inside attacker undermining Barry's work to elevate his own standing within the organisation. Initially, it was thought the attacker might have been someone based on Vitek Boden of Section 7.2.4, but, after further discussion about the threats this attacker might launch, it was felt it should represent a disgruntled insider, rather than a disgruntled contractor. This led to the specification of an attacker (Bob) which reflected the motivations and capabilities of this insider. This example reinforced the fact that attacks are rational from the perspective of an inside attacker, and, by explicating the activities, aptitudes and motivations of personas, participants could better understand the motivations and characteristics of inside attackers as well. The example also illustrates that rather than changing information about an attacker to ensure it fits the risk analysis data, we should instead introduce a new attacker and explore the impact this has on risk analysis.

Third, challenging personas can be a useful way of identifying new personas and assumptions about contexts of use. At the beginning of the third workshop, potential issues were identified with the Alan persona. Although participants believed that Alan was believable, they did not feel his activities were accurate. According to his specification, Alan not only handed over his design documentation to instrument technicians on the commissioning of new equipment, he also provided informal support in the event of any problems. However, instrument technician participants in this workshop indicated that, more often than not, although information is handed over to a company representative or a process operator at a plant, Alan might not have realised that Barry found it difficult to obtain this documentation. Moreover, although Alan might have been contracted to provide some support to Barry, this would only be for a limited time period following commission. After more discussion about who would nominally support Barry, an assumption persona was specified to capture the characteristics of somebody in this role. This persona captured the characteristics of an engineer with specific expertise on the control systems Barry maintains, but was sub-contracted to provide third line support. The persona's name — Eric — was chosen by the participants. The first letter of the name was shared with the real-life user the participants used to derive behavioural characteristics from, but the name was selected as one which would be believable. However, because of the origins of this assumption persona, participants tended to refer to the source user rather than the persona name.

7.5.3 Goal modelling and KAOS

Goal models were useful for structuring requirements, but eliciting them during participatory sessions was far from easy, even when stakeholders have different perspectives to offer on goals. As indicated in the Action Plan section, it was, therefore, necessary to occasionally interleave top-down with bottom-up analysis, and review the tasks to discover goals needing to be satisfied. By eliciting goals which operationalised to these tasks, thinking would be stimulated about other goals also needing to be satisfied.

Goal modelling was also useful for exploring goals needing to be satisfied by three proposed strategies for managing software changes: a file locking / unlocking approach, where files were locked during modification; a modify-merge approach where working copies of files were downloaded, and modifications merged into the repository; and an alarm-driven approach where central telemetry alarms, which fired in the event of a software change, were used as a trigger to the repository to automatically obtain the latest running software from a device. By using or-goal refinements and obstacle modelling, several problems were found in the alarm-driven approach; these had not been considered before the design sessions.

Although much of the goal modelling activities were carried out in these participatory sessions, it was necessary to re-word goal definitions, and prune the goal trees for superfluous or duplicate goals outside of the workshops.

7.5.4 Volere

Keeping requirements and goals separate was found to be useful. Occasionally, requirements would be elicited with no obvious linkage with any particular goal or task. These could be quickly specified in CAIRIS, and later associated with goals and tasks if necessary.

Modifications to the Volere specification, and the ability to synthesise the results of different analysis meant that rather than being a staid document, the generated specification acted as a *case* for the requirements within them because it was possible to see how the different pieces of analysis did (or did not) contribute to a requirement.

The specification was also useful for documenting insights arising from the construction of the rich picture. These insights, such as background information and high level goals, were entered as *Project Settings* in CAIRIS; these settings mapped to the project purpose and scope sections of the specification document.

7.5.5 Activity scenarios

Activity scenarios proved to be effective at surfacing assumptions participants held about the work of instrument technicians. Although the researcher typed these into CAIRIS, the text of the scenario was visible to everyone present. This enabled participants to highlight grammatical and typographical errors, or misused terms. For example, the term *download* and *upload* referred to the retrieval and transmission of data to a location or device respectively. However, in ACME the use of these terms was commonly reversed, which, although not identified during any of the interviews, was identified during the first design session when several participants expressed confusion about some of the wording for the *maintain telemetry software* task.

Contextualising activity scenarios also highlighted subtle differences with an impact on prospective repository usage. To illustrate this, the activity scenario below described how Barry would modify telemetry software as part of a planned activity:

Barry goes into the depot on a Monday morning, and batch syncs his laptop. This involves plugging his laptop into the network, looking at what files have changed, and making sure he has the latest programs for his area. He then picks up his jobs for multiple days. Barry walks 100 yards to the Motor Control Centre (MCC), locates the telemetry outstation, plugs his laptop into the outstation and loads up the program. Barry verifies his software matches up with the same software on the outstation; this is done automatically.

Barry then makes and commissions the relevant change. In this case, Barry calls up the control room to make sure an alarm has been raised based on the new element setup. Barry then saves the change to the outstation and his laptop. The software tool displays the changes and asks for verification. A software change alarm is then generated automatically and sent through to both the telemetry alarm page and the software repository. Barry will commit this change back to the repository as soon as he can. At the end of the day, Barry returns to his depot, fills in his paperwork and batch syncs to the repository.

In comparison, the Scenario in the Unplanned environment has a number of variations:

If Barry hasn't already batch sync'd his software then he VPNs in via mobile broadband to the DCWW network and batch syncs with the repository. As a result, Barry is notified of changes made and software still to be uploaded. In this case, the site he is visiting has had no changes made recently.

Barry drives to the site, locates the telemetry outstation, plugs his laptop into the outstation, and loads up the program. Barry verifies his software matches up with the same software on the outstation; this is done automatically. Barry then makes and commissions the relevant changes. This involves contacting the control room to make sure the alarm has been cleared. Barry then saves the change to the outstation and his laptop. The software tool displays the changes, and asks for verification. A software change alarm is then generated automatically and sent through to both the telemetry alarm page and the software repository. Barry will commit this change back to the repository as soon as he can.

Although the steps used to modify the software are largely similar, Barry was likely to be more tired when commencing the latter scenario than he would be in the former given the potential distance he needed to travel first. Contrasting these scenarios led to discussion about how the repository could be used to alleviate some of the administrative burden associated with making software changes. As illustrated in Figure 7.6, this led to the elicitation of a goal (Generate PTF) where information about the software change could be used to automatically generate change documentation on the technician's behalf.

Like the personas, the tasks also generated side discussions about related work. On more than one occasion, participants would stand up and walk towards the white board where CAIRIS was being

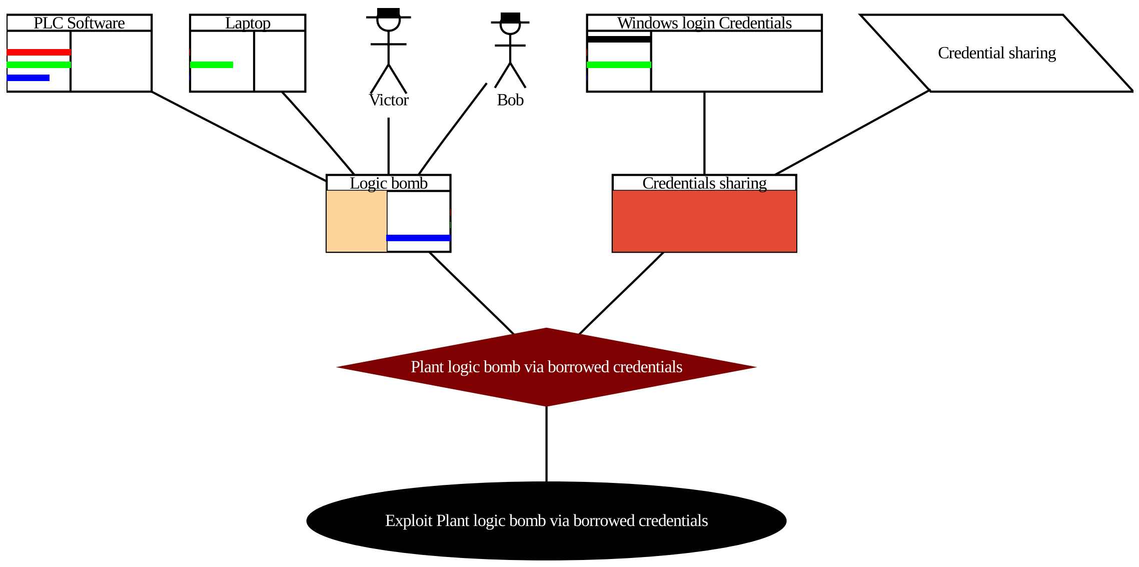

PLC Software
Laptop
Victor
Bob
Windows login Credentials
Credential sharing
Logic bomb
Credentials sharing
Plant logic bomb via borrowed credentials
Exploit Plant logic bomb via borrowed credentials

Figure 7.8: Risk analysis model for planting a logic bomb

projected to discuss issues identified by part of the task.

7.5.6 AEGIS

Even though few risks were elicited, risk analysis activities supplemented other techniques in several ways.

First, as highlighted by [97], the elicitation of assets and modelling of their relationships was a useful boundary object for getting participants to talk about security issues. However, the set-up activities immediately preceding these were just as important; these included agreeing what constituted as high, medium, and low value assets, and what the impact of a marginal, critical, and catastrophic vulnerability might be.

Second, specifying risks helped validate the analysis underpinning them. Risk scores tended to be either very low or very high; as a result, the colouring of risks in the risk analysis model was good for identifying unexpected values. For example, one risk was considered negligible because there was no accountability value associated with the PLC software asset. However, one of the attackers modelled was explicitly interested in abusing the lack of accountability associated with the asset. When the asset modelling was re-visited, it was discovered that this value was implicitly assumed, but not explicitly specified. Consequently, this value was added to the asset, leading the risk score to become maximised, rather than minimised.

Third, although environmental variations in goal models were negligible, thinking about threats and vulnerabilities within explicit environments provided interesting insights. For example, a participant noted that one particular threat involved planting a logic bomb by adding disruptive logic in PLC software, for the express purpose of precipitating a change from one environment to another. Specifically, the threat existed only in the *Planned* environment but, if it was realised, would have led to an immediate universal change to an *Unplanned* environment. Such a change benefitted an insider instrument technician attacker who gained financially by being called out to fix the problem he deliberately introduced.

As indicated at the start of this study's evaluation section, despite the usefulness of risk analysis activities for thinking about security, only a small number of risks were elicited during the intervention. We believe the main reason for this is that several potential vulnerabilities and threats identified at an early stage were factored out of the specification. For example, when exploring the goals needing to be satisfied for the software repository to automatically retrieve software changes from different devices, an obstacle was identified where a device was modified in such a way that an automatic, and unwanted, repository update would be triggered. When refining the obstacle to identify the underlying threats and vulnerabilities, the resulting issues were felt to be architecturally significant enough to render this strategy unworkable. Therefore, rather than eliciting risks from the resulting threats and vulnerabilities, the goals and requirements related to this strategy were removed from the specification.

7.5.7 Misuse cases

Because the number of risks elicited was small, there were few opportunities to fully evaluate the usefulness of misuse cases as a validating technique for risk analysis. We did, however, find evidence that describing attacks in context helped provide new insights into the analysis the misuse cases validated. For example, the following misuse case narrative described how a malicious insider attacker, Bob, planted a logic bomb (threat) by borrowing the credentials of another technician (vulnerability):

- Bob visits a depot where several instrument techs are currently working. Seeing Barry, Bob asks if he can borrow his login details because he has accidentally locked himself out and can't download some PLC changes he needs to make for a job he is about to do. After giving him his credentials, Bob downloads the software he needs for a site change, makes the required software modifications, before making a further change to turn off all the pumps connected to the PLC at a specific time.

Although the *credentials sharing* vulnerability was elicited on the basis of a previous obstacle modelling exercise, no one appreciated that this vulnerability also satisfied a social engineering obstacle until the risk was described in context. Consequently, the obstacle tree (Figure 7.9) associated with this vulnerability was revised to take this new insight into account.

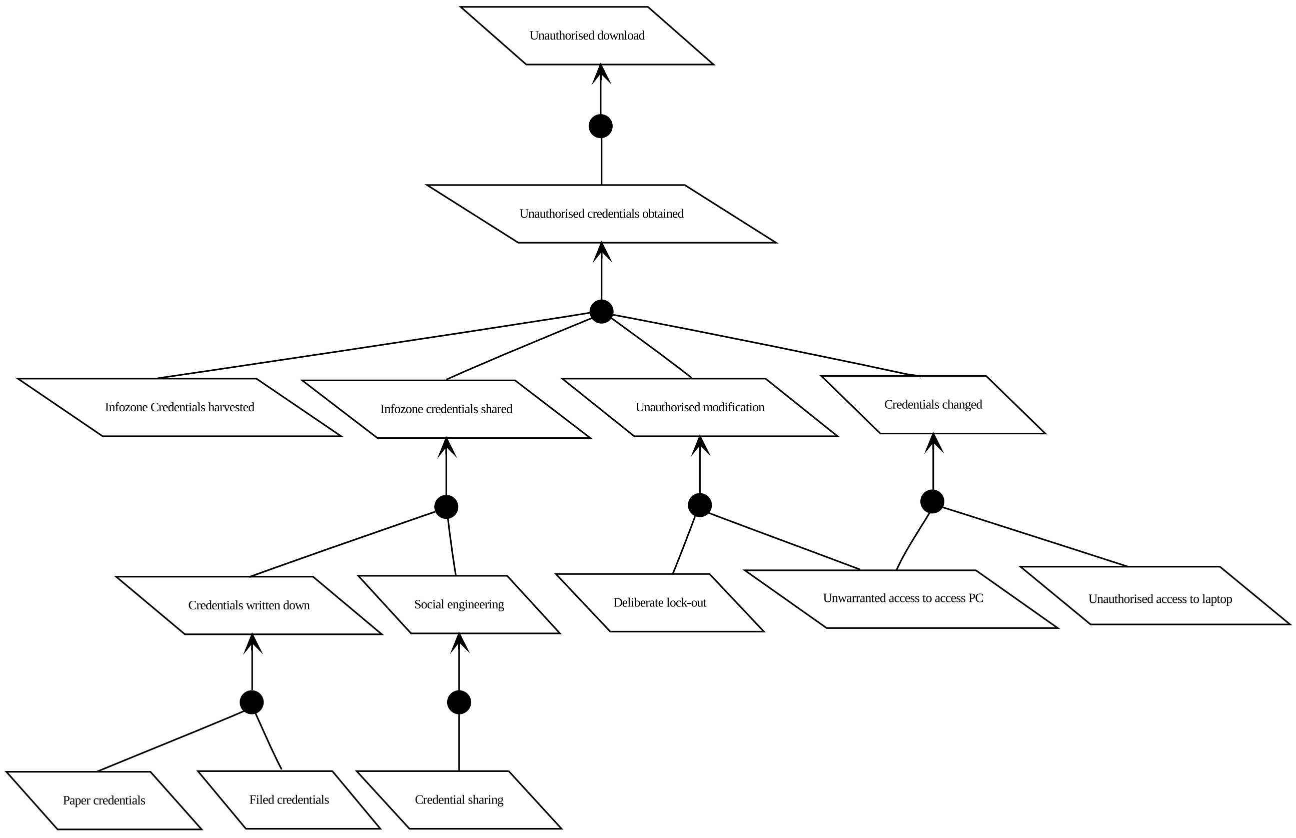

Unauthorised download
Unauthorised credentials obtained
Infozone Credentials harvested
Infozone credentials shared
Unauthorised modification
Credentials changed
Credentials written down
Social engineering
Deliberate lock-out
Unwarranted access to access PC
Unauthorised access to laptop
Paper credentials
Filed credentials
Credential sharing

Figure 7.9: Obstacle model for planned environment

7.5.8 CAIRIS modifications

Extensive use of CAIRIS during workshop settings necessitated the addition of three new features to the tool.

First, as Section 7.5.3 highlighted, several goals and requirements were elicited by developing the task narrative and determining the goals needing to be satisfied as a result. To ensure the train of participant's thoughts were not lost when eliciting tasks, it was necessary to add a short-cut allowing contributing goals or requirements to be quickly specified and associated with the task under discussion.

Second, viewing goal models and closing them down to enter goals into a dialog box proved to be a distraction. It was, therefore, necessary to add a short-cut from the goal model view allowing new goals and their refinement associations to be quickly added.

Third, much time was spent working within a single environment, but duplicating environment properties associated with an object proved to be tedious. Therefore, new functionality was added which allowed environmental properties associated with an object to be duplicated for a new environment.

7.6 Specifying learning

Based on the results of this case study, the following lessons were taken away to inform subsequent research and case studies.

7.6.1 Plan for less participation

Workshops alone appeared to be unsustainable for exclusively eliciting and analysing requirements and other concerns. Arranging workshops with knowledgeable stakeholders was found to be challenging, and some participants had more of a stake in the end product than others. As a result, it may be useful to apply techniques relying on participatory design sessions only when specific issues demand group-based discussion and negotiation.

7.6.2 Addressing assumptions

Although CAIRIS was invaluable in helping elicit and manage different types of requirements, usability, and security data, it also highlighted the importance of assumptions in User-Centered Design artifacts. There is, therefore, a need for design techniques and tools to be more sensitive to the identification and management of assumptions if these are to form the basis of a design.

7.7 Chapter summary

In this chapter, we presented a case study describing how the IRIS framework was used to support the elicitation and specification of requirements for a software repository for managing control software used by a UK water company.

Using Action Research as a methodology, we described the characteristics of the context of intervention, and how the IRIS framework was instantiated to support the goals of the intervention. We also described the steps taken during the intervention, and evaluated the role played by each of the techniques used, which were also supported by CAIRIS. Finally, we identified lessons required to inform subsequent research and IRIS case studies.

Chapter 8

Case Study: Data Support Service Data Directory

This chapter reports the results of a case study where IRIS was used to specify security requirements for a meta-data repository that facilitated the sharing of medical study data.

Using the same Action Research methodology as that described in Chapter 7, we discuss the characteristics of the context of intervention, before presenting the action plan for instantiating the IRIS framework. We also present new techniques for dealing with the lessons learned from Chapter 7: Assumption Persona Argumentation and Misusability Cases. Based on these new techniques we present how the intervention was carried out in three separate design phases: Persona Development, Design Sessions, and Misusability Case analysis; in particular, we describe the parts played by the new techniques.

Based on the action plan, we describe the results of each of the above design phases, before evaluating the intervention. We conclude this study by specifying lessons which re-inform IRIS and the design of the final case study.

8.1 Research environment

The author was invited to apply IRIS to support the UK Medical Research Council (MRC) funded Data Support Service (DSS) project by the University of Oxford's lead project investigator. The aim of DSS was to provide a software infrastructure that facilitated a national data sharing service for MRC-funded *population studies*; these were long-running, longitudinal studies of a group of people sharing some specific characteristic. By providing an accessible interface to such studies, DSS ensured that research data and meta-data could be re-used, thereby reducing the need for running unnecessary and expensive long-term studies.

The University of Oxford was one of several partners in the DSS project. The Science and Technology Facilities Council (STFC) e-Science team was responsible for building and maintaining the Data Gateway component of DSS. This was a portal providing researchers with information about referenced data sets, hosted forums, and best practice documentation on data-set curatorship. The University of Oxford was responsible for developing the Data Directory: a meta-data repository which stored the semantics for the metadata used to describe study data sets. The Data Directory was accessible via the Data Gateway and bespoke web services; these allowed researchers to find datasets of interest based on parameters such as the type of question asked in surveys.

Also partnering in DSS were individual study units. Their role in the project was to act as exemplars for study units providing sharable datasets.

The lead investigator was the gate-keeper for the intervention, and was present at all group-based focus groups. However, the author contacted individual project participants when scheduling smaller design sessions, obtaining specific information, or asking any clarifying questions.

At the beginning of each interview or session, permission to make audio recordings was sought. In one session, however, participants were uncomfortable with recordings being made, so hand-written notes were instead taken.

8.2 Diagnosis

Based on the initial brief, a kick-off meeting was held with the stakeholders to learn more about the context within which the project operated. The author also had the opportunity to observe a half-day project progress meeting, attended by Oxford and STFC team members. From both these observations and background research, the following factors were identified as likely to have an impact on the planned intervention.

8.2.1 e-Science and security

An implicit objective of DSS was to leverage technology and data towards better medical research. As such, the DSS project also furthered research in the area of *e-Science*, which is concerned with the global collaboration in key areas of science and the next generation of infrastructure that will enable it [239].

Although some of the design rationale underpinning the Data Directory was motivated by previous research [66], there was sufficient novelty in both the domain and the implementation technology to make invention, rather than quality, the primary concern of project team members. Although one aim of e-Science research is to cast light on some of these uncertainties, securing innovation tends not to be treated as a priority area. Recent work is beginning to address the challenge of securing e-Science activities [159], however, as Section 2.2.2.4 alludes to, e-Science may be one developmental climate where invention will always be prioritised over quality.

Prioritising core functionality does not mean that security is ignored in e-Science. Rather, there is, as [159] suggests, a tendency to treat risk analysis in an ad-hoc manner. Given the different perceptions stakeholders might hold about assets in an e-Science project, security design decisions might be over or under commensurate with the assets which need to be safeguarded. For example, at one level, highly aggregated data sources may not appear to be a valuable target for an attacker although, with the right search criteria, it is possible to de-anonymise data sources based on the criteria used to search data.

Without careful thought about e-Science assets and the threats and vulnerabilities associated with them, security decisions may also have an unpredictable impact on the user community served by the project. Moreover, as the NeuroGrid exemplar illustrated, the user communities associated with e-Science projects have characteristics which need to be considered as part of any intervention.

8.2.2 The DSS user community

At an early stage, it became apparent that two user roles would dominate the design of DSS. The first of these was the community of academic researchers who would use DSS to find data-sets of interest. The MRC, as project sponsors, were keen to maximise take-up by the researcher community, and initiatives which encouraged this would be looked upon kindly. The second class of users were data managers; these were responsible for curating data sets, URIs (Uniform Resource Identifiers) to which would be available via DSS. The perception held by many on the project was that data managers were key stakeholders that DSS needed to be designed for.

Although not formally documented and only informally discussed during the preliminary meetings, two other classes of stakeholder would also have an impact on the project. The first of these would be the back-end administrators responsible for maintaining the DSS components and infrastructure. The second class were the developers of DSS, specifically the Data Gateway developers at STFC and the Data Directory developers at Oxford. Given the centrality of the Data Directory and the sensitivity of meta-data stored within it, buy-in from the Oxford team was particularly important if security and usability was to be treated as anything other than an afterthought in the design of DSS.

8.2.3 Stakeholder access

Although empirical data from representative stakeholders would have made an invaluable contribution to the analysis carried out, there was no scope for collecting data from researcher end-users. Similarly, time constraints meant that data managers from the study exemplars would also not be available. The difficulty of obtaining representative users for e-Science security cases was not new. The problems experienced by Fléchais [97] in obtaining access to representative users for focus groups during an intervention for a medical e-Science project led to workshops being held with security domain experts instead. Despite the lack of participation from eventual end-users, this analysis still provoked useful discussion about asset values, yielding a number of relevant security requirements.

Fortunately, the Oxford team agreed to act as user proxies because of the time they had spent working with data managers from the exemplar studies, and their domain knowledge based on previous, related research. However, given that the Oxford team had little direct experience of the primary stakeholders of DSS, i.e. the researcher community, it was important that any assumptions that were made about both data managers and researchers needed to be as transparent as possible.

Unfortunately, limited access to stakeholders also extended to the Oxford team as well. Because the project team faced several tight deadlines, the opportunity to work collectively with the project team was severely limited. Consequently, it was essential to be parsimonious with regards to team access, while at the same time ensuring that the IRIS intervention had an impact on the development of DSS.

8.3 Action Planning

Based on the factors described in the diagnosis, the process for eliciting and specifying requirements for DSS needed to accomodate:

- an ability to inform system security as a design progresses, rather than up-front;
- a lack of end-user access, and limited access to project team developers; and
- a need to make assumptions made about users as transparent as possible during design.

The importance of working with assumption data reinforces findings in Chapter 7, which indicated that useful assumptions held by stakeholders can surface during design discussions. Moreover, the increased reliance on assumption-based data required additional approaches for capturing these assumptions, and associating them with different security and usability design elements. With this in mind, two new techniques were devised for making the most out of the assumptions that would inevitably need to be made for this intervention: Assumption Persona Argumentation and Misusability Cases.

In the sections which follow, we introduce these techniques before describing how they fitted into the planned intervention.

8.3.1 Assumption persona argumentation

8.3.1.1 Motivation

Many usability professionals are familiar with analysing assumption-based usage data, but this may not be the case for other professionals working on a Software Engineering project. For example, engineers are usually employed for their technical expertise and domain knowledge; we cannot reasonably expect them to have a working knowledge of usability design techniques as well. These other stakeholders do, however, have tacit knowledge of the problem domain and a sensitivity to the values at play within a system's contexts of use. The challenge is to not only trace assumptions made about personas to their source, but to explicate the claims these assumptions represent. By doing so, we also explicate tacit knowledge about users and their contexts. Like data directly elicited from real-world observations, this data also suggests hitherto unknown threats and vulnerabilities related to a system.

Accepting that data-driven personas are an ideal rather than a norm, we introduced the concept of assumption personas in Section 2.2.2.3: persona sketches created to articulate existing assumptions about an organisation's user population. There is, however, a danger that personas derived from assumptions may prove to be more stereotypical than archetypical [251]. Moreover, assumption personas are also open to other, more general, criticisms about personas. In particular, Chapman and Milham argue that, as fictional archetypes, personas are difficult to verify as there is no way to falsify them [46]. They further argue that questions remain about how personas should be reconciled with other information, understanding what data underpins their characteristics, and what happens when different interpretations are made from the same persona.

Techniques from Design Rationale research are useful for tracking the refinement of assumptions to architectural components and software. Such techniques may also be useful for tracking the same assumptions to less refined concepts used in security analysis. Security design has the same needs for discharging potential ambiguity grounded in assumptions; these may be sources of attack vectors if the vulnerabilities they expose are exploited.

8.3.1.2 Related work

Codifying the rationale underpinning assumption personas guides analysis and decision making, but the *rationale capture problem*, characterised by the reluctance of those involved in design activities to record their rationale, cannot be ignored [40]. Although the Design Rationale community has proposed several different approaches for building rationale capture into the design process, the Security and Requirements Engineering community has taken a recent interest in capturing rationale using the vehicle of informal argumentation. These approaches are founded on Toulmin's Argumentation Model: a logical structure for reasoning about the validity of arguments [248], the elements of which are defined in Figure 8.1.

Alexander and Beus-Dukic have described a number of simple rationale models for Requirements Engineering based on this structure [13]. In their security extensions to Problem Frames described in Section 2.3.1.1, Haley et al. use Toulmin's model to support informal inner arguments for security requirements. Each proposition within this argument is treated as a claim, and argued accordingly. Rebuttals represent *trust assumptions*; these can be countered as part of another security argument, or examined in subsequent threat modelling activities.

Element	Description
Claim	A proposition representing a claim being made in an argument.
Grounds	One or more propositions acting as evidence justifying the Claim.
Warrant	One or more rules of inference describing how the Grounds contribute to the Claim.
Backing	The knowledge establishing the Grounds for believing the Warrant.
Modal Qualifier	A phrase qualifying the degree of certainty in the argument for the Claim.
Rebuttal	One or more propositions challenging the validity of the Claim.

Figure 8.1: Elements of Toulmin's argumentation model

8.3.1.3 Approach

Based on the related work, it is possible to devise an approach for argumentation for assumption personas.

Personas are usually represented as a narrative describing the behaviour of an archetypical user. While authoring these narratives remains a creative exercise, we can augment these by structuring the assumption data contributing to them using Toulmin's previously described argumentation model. Adopting this approach allows us to treat assumptions directly contributing to part of the narrative as a claim. The task of justifying this claim both strengthens the foundations of the persona, and guides subsequent elicitation and analysis activities. These claims are represented conceptually using one or more *Characteristics*; these are propositions about a specific aspect of a persona's behaviour. Characteristics are categorised according to one of the behavioural variable types defined by IRIS personas; these are based on the behavioural variable types proposed by Cooper [61]: activities, attitudes, aptitudes, motivations, and skills. Also associated with a characteristic is a qualifying phrase representing the strength of belief in the characteristic; this qualifying phrase aligns with the modal qualifier in Toulmin's model.

Persona characteristics originate from one of two sources. The first source is some form of *Artifact*: a document related to the problem domain or the system being specified, such as a specification, or a transcript from an interview or design workshop. The second is a design *Concept*: an instance of an object defined within the work-in-progress IRIS analysis, such as a description for an asset, goal, requirement, or even another persona. Because an individual source may give rise to multiple characteristics of the same or different behavioural categories, a *Reference* is associated with a given source and characteristic. The contents of a reference will depend on the source type. In the case of an artifact, a reference contains information tying an attributable piece of information or comment to a source document or verbal comment, e.g. page number, document version, or person. In the case of a concept, a reference contains the name and type of the contributing concept. In both cases, the reference will contain as

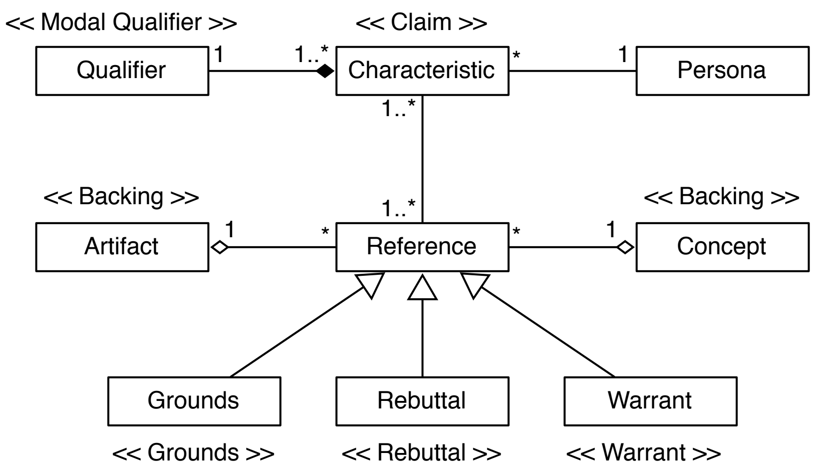
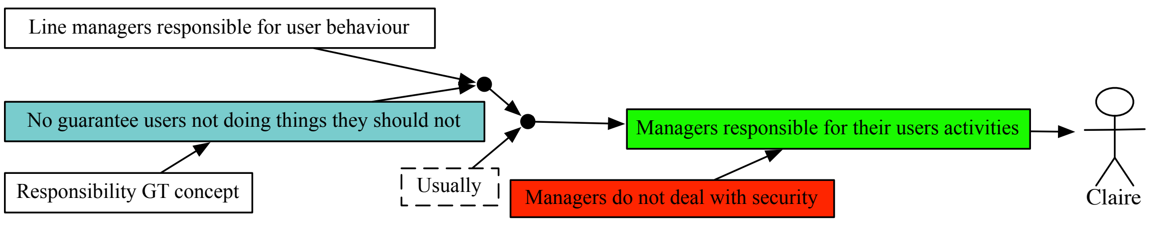


Figure 8.2: Conceptual model of assumption persona data



Line managers responsible for user behaviour
No guarantee users not doing things they should not
Responsibility GT concept
Usually
Managers responsible for their users activities
Managers do not deal with security
Claire

Figure 8.3: Assumption persona model example

much textual attribution information as necessary to justify the persona's characteristic. The name of the reference object is a synoptic proposition of this attribution information. With regards to Toulmin's model, references align to either grounds, warrants, or rebuttals. Where a reference represents a warrant, the corresponding artifact or concept acts as the warrant's backing.

The meta-model in Figure 8.2 summarises these concepts and their relationships. The stereotypes adjacent to each class represent the corresponding concept name from Toulmin's model.

8.3.1.4 CAIRIS modifications

These concepts and their associations from the previous section were added to the IRIS meta-model, and CAIRIS was modified to incorporate the elicitation and management of assumption personas. In addition to the supplemental database tables, stored procedures, and dialog boxes need to facilitate the addition, modification, and deletion of the assumption persona related concepts; the ability to generate visual *Assumption Persona Models* is also needed.

The interface for generating and interacting with assumption persona models is similar to the other models described in Chapter 6. The graphical viewer associated with the assumption persona model allows the filtering of models by persona name, behavioural variable type, and individual persona characteristic. Figure 8.3 is an example of a filtered assumption persona model for a persona characteristic associated with Claire, one of the NeuroGrid specification exemplar personas.

This example models the characteristic *Managers delegate security decisions*; this is an expectation by Claire that she takes responsibility for information security issues in her research group. The white boxes represent assumptions drawn from the data which act as grounds for this characteristic. The warrant linking these grounds to the characteristic is represented by the marine-green box. In the example, this arises from a researcher's comment indicating that, because he has no idea what authorised users do with the data they have been granted access to, he didn't want to take responsibility for their action. The grey box associated with this is the backing for this warrant, which, for this example, is observational data. Finally, while this characteristic may be reasonable, it is not indefensible; this is indicated by the red rebuttal box. This corresponds to a reference which contradicts the warrant indicating that managers *are* happy to delegate all responsibility for security to their sub-ordinates. The grounding for this assumption is not particularly strong, but is strong enough to suggest the analyst's confidence in this characteristic — indicated by the dotted modal qualifier box — is less than certain, i.e. *usually.*

8.3.2 Misusability cases

8.3.2.1 Motivation

As Figure 5.1 illustrates, scenarios are both Security and Usability perspective concepts. To usability engineers, they describe how people use a system to carry out activities achieving their personal or occupational goals. To security engineers, scenarios describe how a system might be misused towards an attacker's ends. Although flexible enough to be used in both contexts, a scenario in one context is not necessarily useful in another. A scenario describing a hacker's attack on a web-server does not necessarily provide insight into how the web-server's administrator perceives its usability.

One way of engaging developers to think about both security and usability might be to encourage them to consider the negative impact their decisions could have on the system they are building. A system could go wrong in many different ways; it could be exploited by an insider or external attacker, or it could be sufficiently unusable that key-stakeholders lose confidence in its capabilities. By focusing on the different unintended consequences of a system, developers may be encouraged to track problems back to their root causes.

In this section, we present *Misusability Cases*: scenarios which describe how design decisions may lead to usability problems subsequently leading to system misuse. We describe the related work motivating this technique, and then present its approach.

8.3.2.2 Related work

Modelling misuse with scenarios is not a new idea; for example, as Section 4.6 reported, misuse cases are already part of the IRIS meta-model. However, it remains unclear how useful misuse cases are for understanding the impact of misusability. Alexander [11] suggests that misuse cases *can* be used for examining usability issues; he presents an example where confusion about the use of an interface causes a novice user to become a negative agent. A corollary of this approach is that the user is typecast as an attacker; this, as we argued in Section 2.1.1, is analogous to treating the user as the cause of poor interface design.

Rather than treating the user as a problem, we need to discover the contributing factors to inadvertent misuse or abuse of a system. This means looking beyond the classic view of systems used precisely as their designers intended. An example of such thinking is Dunne and Raby's work on *Design Noir* [77]; this argues that our emotions and needs are played out in technology across a much broader spectrum of use originally envisaged by its designers. Other work includes value scenarios, which were briefly introduced in Section 2.2.2.2. These are vignettes which describe the systematic effects of a system to both direct and indirect users over an extended period of time [176]. Value scenarios describe both the positive and negative systematic effects of technology without considering users as malevolent; they do so by describing the negative impact of forgetting about certain values, such as prejudice and inequality. Not all software systems are as divisive, pervasive, or long-running as those typically described by value scenarios. Nevertheless, it may be possible to stimulate similar narratives with more supplemental information about the system, its users, and its contexts of use. This information may not be available during the preliminary stages of design where it is envisaged that value scenarios should be employed, but it might be available from the data collected during later stages.

The IRIS meta-model allows modelling of the impact of security decisions on the usability of tasks to personas, but the only way misusability can be explored is by treating personas as attackers, or eliciting obstacles giving rise to secure misusability. If we consider obstacles as exceptional behaviour, then use cases (Section 2.3.3) might form the basis of eliciting such obstacles. Although use cases often describe the normal course of an actor's usage of a system, exceptional behaviour can be associated with individual steps. Using this approach, accidental misuse of a system can be elicited without typecasting a persona as an attacker, but only if the requirements giving rise to the misuse are known. Yet, it may be the case that all we have are clues to what this misuse might be, and nothing that can be readily associated with a use case. Such clues might include possible ambiguity in a specification, or some assumptions about a persona's behaviour that, in some cases, might cause certain types of behaviour.

Although the IRIS meta-model supports assumptions expected to hold by the system using domain properties (Section 4.5), it does not consider assumptions that underpin the meta-model concepts themselves. Without a specific means of associating these assumptions with design, identifying the associations

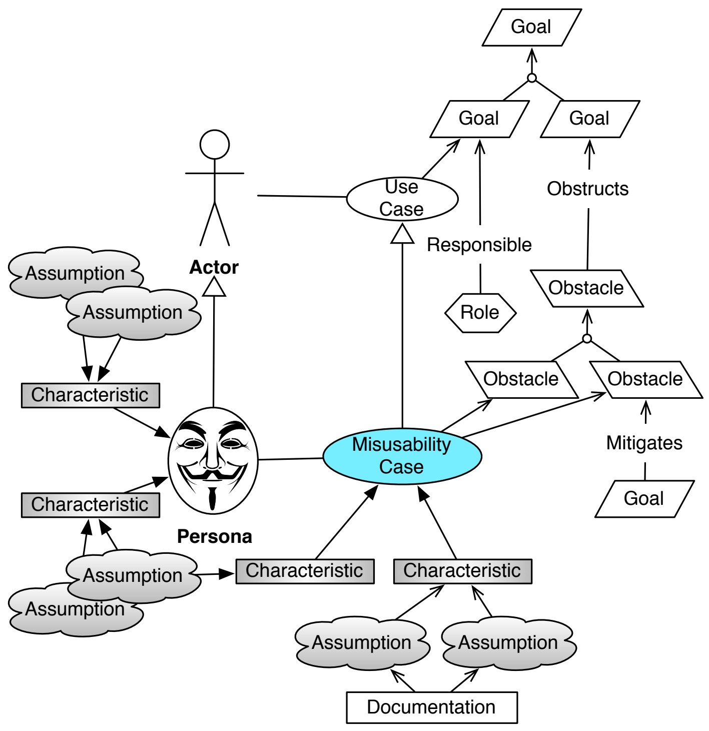


Figure 8.4: Misusability case with other design concepts

between misuse and the system design decisions that cause them remains an ad-hoc affair. The work in Section 8.3.1 explained how such assumptions can be used to help ground personas. If the Toulmin Model can be used to ground personas, it may also be possible to use it to ground assumptions about possible system or user ambiguity. Characteristics derived from these may form the basis for establishing contexts where security misusability can occur.

8.3.2.3 Approach

Misusability Cases are scenarios where a persona achieves a personal or work objective, but inadvertently exploits one or more vulnerabilities in order to do so; we encapsulate these scenarios using IRIS tasks. The factors leading to this exploitation are myriad; a user might accidentally circumvent application error checking, misread software instructions, or carry out behaviour that conflicts with organisational security policies. However, in all cases, these factors arise because of vulnerabilities in the context of use within which the task takes place.

The aims of the Misusability Case technique are twofold: first, to identify cases of insecure misuse within the environment where activities are carried out using a designed system; second, to elicit the root causes of this misuse, together with the system goals which mitigate them.

As Figure 8.4 illustrates, misusability cases do not exist in isolation, and are not elicited before other analysis is carried out first. This differs from misuse cases which have traditionally been used as a brainstorming technique for discovering possible causes of misuse. Before applying misusability cases, we assume system goals have been elicited. We also assume that one or more personas have also been elicited to reflect different roles the system needs to be designed for. As the figure also illustrates, some goals in this model are operationalised as use cases. A misusability case realises a use case describing a particular episode of system behaviour carried out by an actor. Each actor associated with the use case is realised by a persona.

Eliciting misusability cases Underpinning each misusability case are Characteristics. Figure 8.4 indicates that these characteristics are founded upon assumptions; these assumptions are structured according to the argumentation model presented in Section 8.3.1. As such, these assumptions act as grounds or warrants used to establish the claim acting as the characteristic.

The first step involves identifying implicit assumptions being made about the design related to the use case. A variety of techniques can be used to discover these assumptions. When analysing documentation, such as architectural design documentation or user manuals, techniques proposed by Dewar's Assumption-Based Planning methodology [72] are particularly useful. These techniques include using journalist questions (Who? What? When? Where? Why? and How?) about items of data, and looking for statements where the words *will* and *must* are used.

Using the conceptual model described in Section 8.3.1, references are created for each assumption. Each reference contains a statement summarising the assumption, a link to the source material, together with an excerpt from the source material justifying the assumption. References may also be elicited from design artifacts, such as personas.

Once a collection of references have been elicited, the characteristics for a convincing misusability case are developed. The process for developing misusability case characteristics are analogous to those used for developing persona characteristics described in Section 8.3.1. A claim is made about some characteristic of the system which might be liable for misuse. The references are used to act as grounds or a warrant to this claim or, if necessary, a rebuttal. Finally, a modal qualifier is associated with the characteristic based on the analyst's confidence in this claim.

The final stage involves writing a supporting task satisfying these characteristics while, simultaneously, successfully carrying out the steps within the use case. Enacting the task is the persona fulfilling the use case actor's role. The behaviour exhibited by the persona should be commensurate with the characteristics built into the task; if the characteristics of the misusability case are such that they conflict with the persona's objectives then this should be reflected in the task narrative.

Applying misusability cases The next stage involves identifying the obstacles directly contributing to the different aspects of misusability in the misusability case. Based on these obstacles, the higher-level

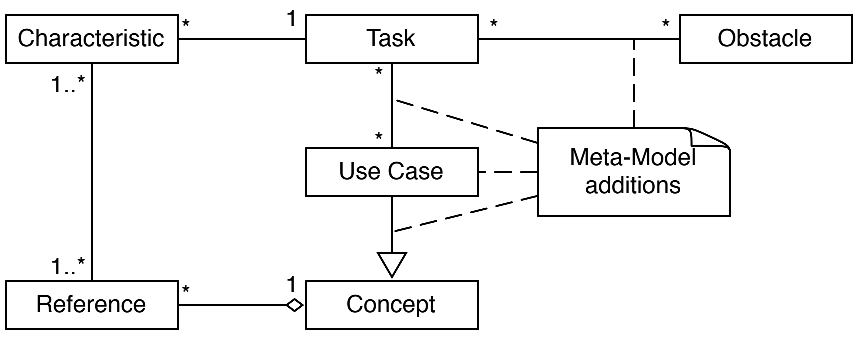


Figure 8.5: IRIS meta-model revisions to incorporate misusability cases

obstacles these lower-level obstacles help satisfy are elicited. This step continues until system goals are identified, or new goals are elicited, which are obstructed by these obstacles. Although this step could be construed as an exercise in bottom-up analysis, fitting the misusability case and its contributing obstacles into the larger goal model necessitates both top-down and bottom-up analysis.

Once the misusability case has been reconciled with the goal model then one of two actions may be possible. Eliciting both the misusability case and contributing obstacles may have provided insights suggesting new goals to resolve the obstacles identified. If this is the case then these are added to the goal model. Alternatively, it may not be possible to mitigate the elicited obstacles because further investigation into the problem domain is needed, or the controls needed to mitigate the obstacles are out of scope. In such cases, the obstacles are assigned to a particular role. This role is responsible for further analysis leading to eventual mitigation of the obstacle. Explicitly assigning the obstacle to a role mitigates the possibility of diffusion of responsibility, where unresolved problems are ignored because no single agent is responsibility for them [69].

CAIRIS modifications Both the need to elicit use cases and the need to model how obstacles contribute to tasks motivated the need for a revision to the IRIS meta-model. Fortunately, as Figure 8.5 indicates, these changes have only a modest impact on both the IRIS meta-model and CAIRIS. Although a new IRIS concept, Use Case, is added, the remaining changes involve adding associations to existing models. Moreover, because the concept of characteristic is used, the augmentations to CAIRIS for supporting assumption personas are also re-usable.

An additional graphical view, Task Characteristic Model, was added to CAIRIS; this was identical to assumption persona models, with the exception that characteristics are associated to task ellipses, rather than persona stick figures.

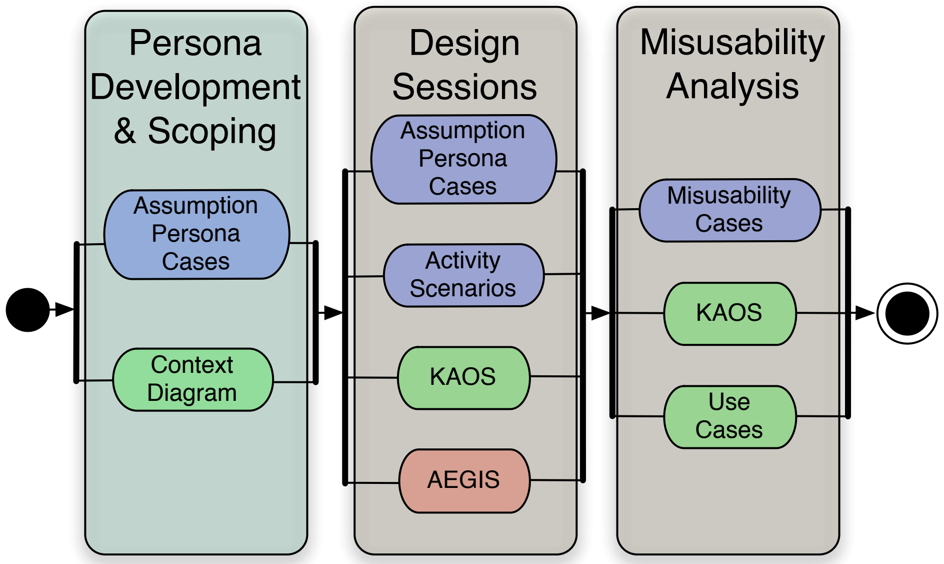


Figure 8.6: Instantiated process for the Data Directory specification

8.3.3 Action plan

The process illustrated by the UML activity diagram in Figure 8.6 was orchestrated. In the following sections, we describe an action plan for eliciting and specifying security requirements for the DSS Data Directory; this includes information on how the techniques in Sections 8.3.1 and 8.3.2 would be used.

8.3.3.1 Persona development and scoping

While the project team's responsibility formed a natural scope of analysis around the DSS Data Directory, there was concern that security issues might cross-cut organisational boundaries; such issues may be the responsibility of one team, but the adverse impact could affect others. It was also necessary to understand the different expectations held about the prospective DSS user-community. To deal with both of these issues, it was decided to identify implicit assumptions in the available documentation, and use these to form the basis of assumption personas. In doing so, the expectations about end-users are made explicit, and subsequent discussion around these confirm a useful boundary for the analysis to be carried out in later stages.

For each role relevant to the scope of analysis, the available documentation is reviewed to elicit document references for each role. These are used to establish persona characteristics and, based on these, assumption personas. Once the assumption personas are developed, these are presented to the project team for review. Any issues raised by the team are used to revise the assumption personas or correct any misinterpretations held about DSS.

8.3.3.2 Design sessions

By carrying out this intervention in parallel with on-going project activity, using participative design sessions alone was infeasible. Conversely, however, limited documentation artifacts meant that group-based design sessions would be required to elicit the data contributing to the requisite IRIS concepts.

The Design Sessions stage entails holding small focus groups with project team members. Each session focuses on the use of activity scenarios, KAOS, or AEGIS. An activity scenario session involves modelling scenarios carried out by the elicited assumption personas in their respective contexts. A KAOS session involves eliciting goals and requirements needing to be satisfied in order that elicited activity scenarios development can be carried out. In both session types, assumption personas are used as an authority for user expectations; these are modified if aspects of the analysis challenge their characteristics. AEGIS sessions involve carrying out asset modelling for the different environments, and discussing possible attackers, threats and vulnerabilities that might arise due to environmental factors; based on these, risks are identified, together with countermeasures mitigating them.

After the final session, each goal is examined and assigned a responsible role. Following this, a specification document is generated and sent to the project team members for their review.

8.3.3.3 Misusability case analysis

In order to help make assumptions more transparent and, simultaneously, help inform system usability and security using the work of the project team, a misusability case analysis phase was appended to the IRIS process.

Using both previously analysed and new documentation produced by the project team, assumptions are identified leading to the elicitation of misusability cases. These artifacts are used to elicit contributing obstacles, together with the goals these obstacles obstruct. This activity stimulates innovative thinking about new goals for resolving these obstacles.

Elicited goals are refined to requirements and assigned a responsible role. Following this, a revised specification document is generated and sent to the project team members for review. Once the team is given sufficient time to review the analysis, a final wrap-up session is held where the final results are presented.

8.4 Action taking

In this section, we describe how the steps in the action plan were used to guide the actions taken during this intervention.

8.4.1 Persona development

By the time this stage began, only two documents were available for eliciting assumptions: a requirements specification for the DSS Data Gateway, and a technical annexe to the DSS contract signed by all project partners.

After a review of the documentation, three roles were evident: researchers, data managers, and gateway administrators. Based on these roles, the documentation was analysed three times to elicit assumptions. On each occasion, assumptions were elicited about behaviour which could be reasonably assumed if the documentation accurately represented the concerns of the particular stakeholder role. For example, a requirement indicating that first-line support to the Data Gateway would be provided between the hours of 9 am to 5.30pm might reasonably suggest that the authors believe researchers work only during commercial office hours.

By the end of this stage, 46 document references were elicited from the two documents; 18 of these were for researchers, 14 for data managers, and 14 for gateway administrators. Three skeleton assumption personas were created for each of these roles: Alex (an academic researcher), Brian (a data manager), and Colin (an administrator for the Data Gateway). The document references for each persona were further structured by behavioural variable type, and persona characteristics were directly derived from each document reference. Finally, for each persona's behavioural variable type, a narrative was written commensurate with these persona characteristics. For example, based on the persona characteristics *In no hurry* and *Looking to apply data-set once discovered*, the following Motivations narrative was written for Alex:

Alex is looking to use a data-set as soon as he discovers one which is suitable. That said, he isn't in a particular hurry, so is prepared to wait for his registration to the Data Gateway and the respective data set to be approved.

On the 26th May 2010, the results of the analysis to date were presented to the Data Directory project team. This presentation described the scope assumed for the analysis, provided an overview of the analysis carried out, and Alex, Brian, and Colin were presented. Each persona was presented in a single slide, where each slide described three particular persona characteristics. The third characteristic was chosen as the most divisive, in order to stimulate lively discussion about the persona. At the end of this session, it was agreed that Colin's activities were not relevant to the scope of analysis, and this persona was dropped from the remainder of the intervention.

8.4.2 Design sessions

Figure 8.7 summarises the design sessions held. Due to project deadlines, rather than having access to multiple developers per session, only a single developer was available. The same developer was consistently used for each session, and was available for email clarification when queries arose outside

Date	Session Type	Participants
08/06/10	Activity Scenarios	Data Directory developer
09/06/10	KAOS	Data Directory developer, domain expert
18/06/10	AEGIS	Data Directory developer
30/06/10	AEGIS	Data Directory developer

Figure 8.7: Design session summary

of these sessions. In addition to the allocated data directory developer, a non-project domain expert participated in one of the design sessions. Although this participant was only partially knowledgeable in the on-going DSS project, she was aware of the problem domain in general.

In addition to eliciting the concepts associated with the techniques used in each session, the assumption personas were progressively refined and embellished with further references from both documentation and other design concepts.

8.4.3 Misusability case analysis

Due to project commitments, a five month hiatus separated the final design session and the beginning of the misusability case analysis phase. During this time, the architecture and core components of the Data Directory had been designed and developed, and were under-going user testing. In the intervening period, two additional documents were available for review: a user-guide for the Data Gateway, and an implementation guide for the Data Directory. This implementation guide acted as an overview of the interfaces to the Data Directory, and discussed the system's approach to qualities such as security, performance, and testing.

After a review of both documents, it was noted that the import of meta-data into the Data Directory was within the project's scope of analysis, but was discussed in few places. Consequently, a use case describing how a data manager would import meta-data into the Data Directory was prepared using the information available in the implementation guide. These documents were then reviewed in more detail to identify implicit assumptions being made about the design related to the use case. These assumptions were modelled in CAIRIS using document references.

Once a collection of references had been elicited, three candidate misusability cases were developed. Each misusability case was subject to the further analysis described in the action plan section.

In the following sections, we demonstrate how misusability cases were developed and applied.

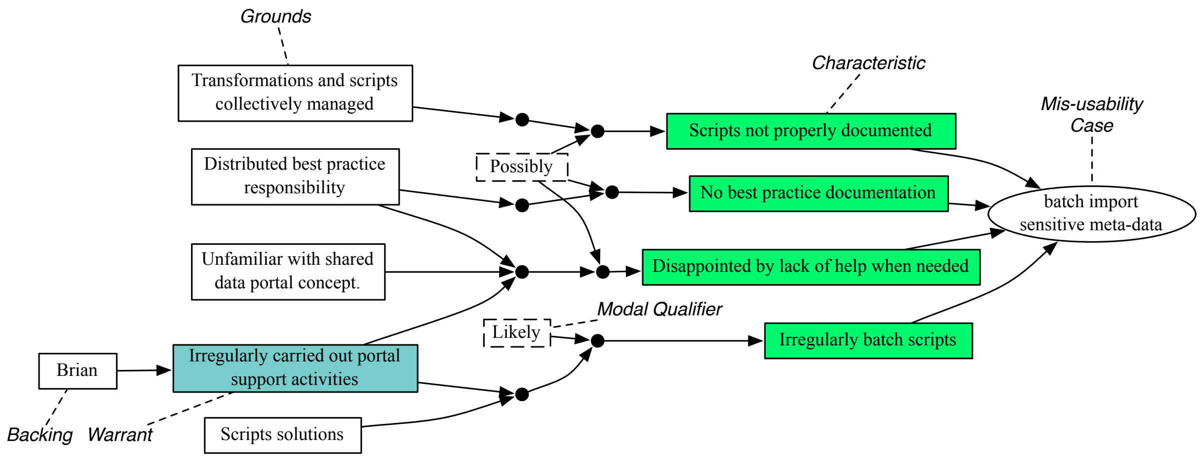

Grounds
Transformations and scripts collectively managed
Distributed best practice responsibility
Unfamiliar with shared data portal concept.
Brian
Irregularly carried out portal support activities
Scripts solutions
Backing
Warrant
Possibly
Likely
Modal Qualifier
Characteristic
Scripts not properly documented
No best practice documentation
Disappointed by lack of help when needed
Irregularly batch scripts
Mis-usability Case
batch import sensitive meta-data

Figure 8.8: Characteristics and argumentation structure underpinning a misusability case

8.4.3.1 Developing the misusability case

We will consider the functionality associated with importing study meta-data into the Data Directory, via the following *Batch import meta-data* use case:

The data manager completes the fields of a mapping file and, from his web-browser, enters a URI into the meta-data upload page. When the upload page is displayed, the data manager enters the location of the mapping file and clicks on the Upload button. The system uploads the meta-data to the Data Directory based on the data in the mapping file and, after several minutes, acknowledges successful import of the meta-data to the data manager.

The mapping file describes meta-data about the meta-data, such as the file names, data locations, and security policies associated with the meta-data. Also associated with this use case was a pre-condition that the meta-data itself had been prepared and ready for import.

While the use case suggested little to form the basis of security misuse, a number of clues were found in related artifacts. The implementation guide suggested that utilities to support data managers in preparing their meta-data *may* be made available via a central repository on the portal. This central repository also hosted forums and best practice documentation about how to make best use of the portal. Other clues were found in the related persona description. Brian was found to irregularly use the portal and, because of his unfamiliarity with the mechanics of sharing data outside of his study unit, was unfamiliar with the process of importing data to the Data Directory. Consequently, a lack of documentation and best practice was likely to be a cause of frustration. We also know that Brian was adept at scripting, and thereby likely to script the process of satisfying the use case preconditions. Assumptions were recorded as references, following which characteristics were elicited that underpin a misusability case undermining the use case; this argumentation structure is illustrated in Figure 8.8.

Using these characteristics as guidance, the below *Batch import sensitive meta-data* misusability case was written:

Brian had spent most of the morning preparing meta-data for export. Deep (sensitive) meta-data would be imported into local, study databases, while shallow (summarised) meta-data was targeted for the Data Directory. He had hoped to use standards and guidelines on the gateway, but was disappointed by the lack of anything useful that would help him. Nevertheless, Brian managed to organise his meta-data into the layout he managed to induce from some of the XSLT scripts he was able download. After finally finishing the preparation of his data-sets and meta-data, Brian created the mapping files needed for the data import process. Fortunately, most of them were very similar so most of the files he used were based on an initial template he created for one of his data-sets. Brian entered a URI he had been provided for uploading meta-data to the Data Directory, and logged in using his data manager credentials. Brian then specified the mapping file corresponding to the meta-data he wanted to upload and hit the Upload button. Several minutes after clicking the Upload button, Brian received a message from the portal indicating that the meta-data had been uploaded.

Although not written in the misusability case, Brian had accidentally uploaded a mapping file for public meta-data pointing to sensitive meta-data. As a result, sensitive meta-data has been made publicly available on the portal.

8.4.3.2 Mitigating the misusability case

Figure 8.9 presents an excerpt from the goal model associated with the aforementioned misusability case. As the figure illustrates, we identified three contributing obstacles causing the misusability case to be realised.

The first of these obstacles was a lack of documentation about the import layout. On considering possible root obstacles, we discovered that satisfying this obstacle contributed to the obstacle of Data Directory documentation being unavailable. One reason that this documentation was not available is because no one was explicitly responsible for publishing anything. We, therefore, mitigated this first obstacle by specifying a goal (Layout documentation) stating that the expected data layout shall be published when an import tool is made available to data managers.

The second obstacle pointed to a lack of contributed best practice documentation. The frustration caused by this, like the first obstacle, lengthened the time taken to complete the activity; this possibly increased the likelihood of the slip occurring during the misusability case; this slip was reflected by the inadvertent specification of the mapping file data policy as public. Although the management of portal documentation was de-scoped from this analysis, the impact of the obstacle affected the Data Directory. For this reason, the obstacle was assigned to the portal administrators to ensure it was addressed by this team.

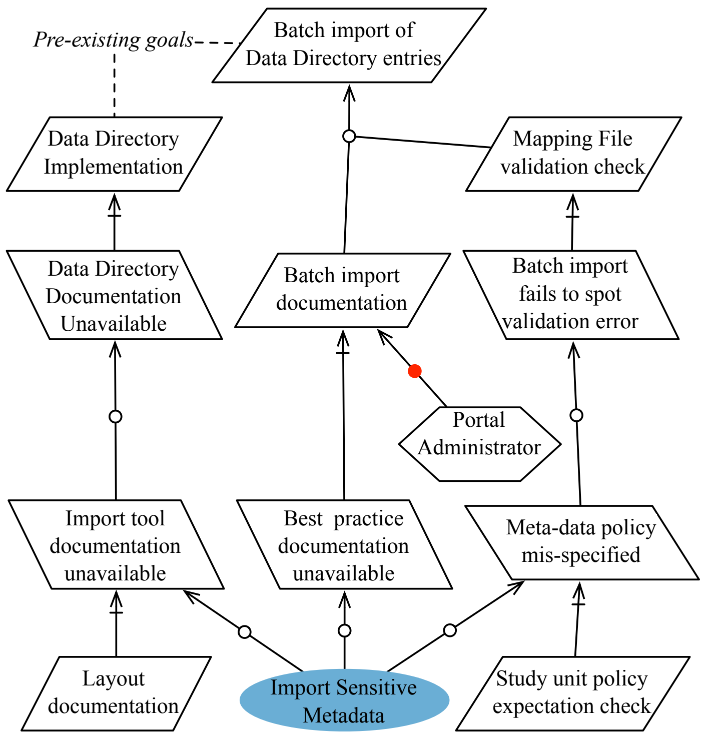

Pre-existing goals
Batch import of Data Directory entries
Data Directory Implementation
Mapping File validation check
Data Directory Documentation Unavailable
Batch import documentation
Batch import fails to spot validation error
Portal Administrator
Import tool documentation unavailable
Best practice documentation unavailable
Meta-data policy mis-specified
Layout documentation
Import Sensitive Metadata
Study unit policy expectation check

Figure 8.9: Misusability case contribution to goal model

The third obstacle related to the upload of the inappropriate meta-data to the Data Directory due to mis-specification of the mapping file template. The obstacle stated the sensitive meta-data was specified as publicly accessible. This obstacle was too granular to immediately suggest a mitigating goal, thereby requiring further thought about root obstacles this was satisfying. Because the obstacles were based on system rather than user errors, it was inappropriate to define the immediate parent obstacle as a slip on the part of Brian. However, the consequences of the slip suggested that some form of validation safeguarding against such slips failed. A root obstacle that might have been satisfied by the contributing obstacles stated that the batch import process failed to spot the validation error. Yet, this obstacle begged the question of what such a validation error might entail? One means of providing this validation involved stating a priori expectations about the data manager's unit's policy for exporting different classes of data and meta-data. By consulting these expectations, discrepancies between unit data and the unit data policies could be highlighted during a pre-import validation check. To realise these requirements, two goals were added to the goal model. The first of these stipulated that a mapping file validation check shall be carried out in order to satisfy the goal of batch importing meta-data into the Data Directory. The second goal resolved the meta-data policy mis-specified obstacle; this involved stating that a study policy expectation check shall be carried out as part of the validation process.

Behavioural Variable	Persona	
	Alex	Brian
Activities	6	3
Attitudes	5	3
Aptitudes	2	4
Motivations	1	0
Skills	1	1
Environmental Narrative	1	0
ALL	16	11

Persona	Reference	
	Document	Concept
Alex	30	3
Brian	14	2

Figure 8.10: Persona characteristics summary by behavioural variable type (left) and document and concept references associated with personas (right)

8.5 Evaluating

A final, wrap-up meeting to summarise the analysis and walk through the misusability cases was held with members of the project team on 25th November 2010. Based on this meeting, a number of minor revisions were made to parts of the IRIS model before a final specification was mailed to the DSS team a few days later.

While the outcome of the intervention was less positive than originally hoped, we believe the outcome of the intervention from a research perspective successfully validated the IRIS process used for this case study. One of the testaments of this validity was the ability to elicit relevant security goals and requirements for the Data Directory despite a lack of access to representative users, and only irregular access to the project team. These were key lessons identified in Chapter 7.

In the following sections, we evaluate the different stages of the IRIS process applied in more detail.

8.5.1 Persona development

Figure 8.10 (left) summarises the different types of persona characteristic associated with each of assumption persona.

While it is difficult to draw clear conclusions from this data, two observations can be made. First, it appears that, despite the nature of the documentation used, it was possible to elicit a surprising amount of data about both the possible activities and attitudes of personas. Moreover, if we also compare the number of researcher and manager related references elicited at the end of the Persona Development stage (18 and 14) with those reported in Figure 8.10 (right) then we notice that a significant number of additional references were elicited during the Design Sessions and Misusability Case Analysis stages. As such, both personas evolved throughout the design sessions. This concurs with best practice in the use of personas, which suggests that personas should be fostered throughout a project lifecycle [196].

Figure 8.10 (right) also indicates that a comparatively small number of references were elicited from IRIS concepts. These concepts were elicited during the design sessions, where the focus was on eliciting other concepts rather than purposefully revising the personas. A total of 4 references were elicited from assets (1) and tasks (3) respectively. Interestingly, these references arose from insights while eliciting and modelling security and usability concepts rather than requirements concepts such as goals and, to a lesser extent, obstacles. We believe this reinforces the notion that requirements concepts serve best as a boundary object for what a system specification will look like, rather than as a means of eliciting data about the problem domain it will be situated in.

Although identifying grounds for characteristics was found to be straightforward, identifying warrants was more difficult. It was found that, prior to their initial validation, many of the characteristics were based exclusively on grounds. As such, value judgements about the source data and context were directly reflected in these characteristics. Although the initial workshop surfaced a number of these issues, it was usually not until the personas were directly used to model tasks in design sessions that certain invalid characteristics were identified. Applying the personas within a specific context did, however, help identify missing inferential data, or guide the refactoring of the argumentation structure for affected characteristics.

8.5.2 Design sessions

Figure 8.11 summarises the concepts elicited and specified on completion of the intervention.

As the table shows, concepts were present in either one or both of the *Research* and *Study* environments. The Research environment was concerned with the context of use where Alex interacts with the Data Gateway as part of his research. The Study environment was concerned with the context of use where Brian interacted with the Data Directory, either via the Data Gateway, or via other, provided, interfaces.

The table also shows that the number of concepts elicited from the risk analysis meta-model was comparatively small. This was mainly due to the resolution of many security and usability problems during the KAOS and activity scenario sessions, coupled with the misusability analysis exercise. Another reason for the small number of explicit risk analysis elements was a tendency by the security team to dismiss security issues that were deemed out of scope. As such, on more than one occasion, assets which had originally be identified as being in scope, such as gateway specific documentation about the use of the Data Directory, was de-scoped. This issue of passing responsibility for out-of-scope issues was also apparent from the threats and vulnerabilities highlighted in both areas, most of which were derived from

Concept	Research	Study	ALL
Asset	9	11	13
Attacker	3	1	3
Countermeasure	0	0	0
Domain Property	N/A	N/A	1
Goal	12	21	30
Obstacle	5	20	20
Persona	2	1	2
Requirement	N/A	N/A	4
Response	1	1	1
Risk	4	0	4
Role	N/A	N/A	6
Task	4	3	6
Use Case	2	1	3
Threat	4	0	4
Vulnerability	9	2	9
ALL	55	61	107

Figure 8.11: IRIS concepts specified on completion of the DSS intervention

the OWASP threat and vulnerability catalogue (Section 6.4.3) and appeared to target and exploit Data Gateway assets and associations.

The issue of scope deferral was also contextual, rather than specific to concepts in any particular model. Most of the risk analysis concepts were elicited from the Research environment; these concepts targeted assets which were out of the project scope. The few risk analysis concepts which did concern the Study environment were also marginalised as out-of-scope. For example, of the four risks elicited, only one (a Man-in-the-Middle attack on the Data Directory) concerned the Study environment. Upon discussing resolutions to this, it was agreed that the Data Directory relied on a secure channel between it and the Data Gateway. Therefore, responsibility for mitigating more general Man-in-the-Middle attacks was delegated to the Gateway administration team.

Although the project team were reluctant to take a *defence-in-depth* approach to tackling security problems, the IRIS process ensured that security concerns were eventually reflected by the model. This was possible by focusing attention on goal obstruction within the Study environment which, unlike the Research environment, concerned IRIS concepts which *were* within the project scope. This allowed threats and vulnerabilities to be mitigated at the design level when considered in context with other Data Directory goals. This was especially useful because, besides the generic internet-facing threats and vulnerabilities associated with OWASP, it was not entirely clear what the threat model facing DSS might be.

With regards to the design sessions themselves, the approach taken proved to be more flexible than envisaged in the action plan, which indicated that specific activities would take place in each type of session. In practice, multiple techniques were used when the situation deemed it useful. For example, both activity scenarios and KAOS were used in each of these respective sessions when it was felt most appropriate. Similarly, elements of AEGIS were also used in both sessions to elicit assets, their relationships, and concerns arising from goals and tasks. As per Chapter 7, switching from the use of one technique to another did not seem to hinder the analysis process of the participants in the sessions.

During the design sessions themselves, the presence of a domain expert in one session was also found to be useful. Haley and Nuseibeh [185] observed that experts provide essential domain knowledge about the subtleties of threats, but non-experts ask journalist questions challenging implicit assumptions assumed by the domain expert. The results of the design sessions concur with this observation. When the tasks carried out by one of the personas was modelled during one session, one non-expert participant raised pertinent points about implicit assumptions in the task description; these were not accounted for by the personas, and led to the rebuttal of one characteristic.

8.5.3 Misusability case analysis

One of the benefits of the Misusability Case technique was its ability to increase developer engagement towards understanding how poor usability can both hinder take-up of the system they were trying to build, and compromise security.

A similar misusability case to the example was developed to explore the impact of corrupt meta-data causing the import process to misinterpret the quality of the data being transferred. Consequently, although draft data was imported into the Data Directory, this was interpreted as real study data. When this misusability case was presented, together with goals which would mitigate this problem, the developers were adamant that the obstacle should not be mitigated. Doing so, they argued, might discourage data managers from importing any meta-data into the Data Directory; the developers did not want to pre-empt what data managers should or should not wish to import into the Data Directory.

Following this discussion, a second misusability case was presented to the developers. This was a corollary of the first and described the impact of the corrupt data from the perspective of Alex. In this misusability case, the invisible control characters in the imported meta-data caused the portal to leave the quality indicator field blank when information about a study is viewed by an end-user. As a result, Alex obtained the data and used it in his own research without realising it wasn't real.

After both misusability cases were presented, the developers acknowledged that data managers would only use the portal if it was seamlessly integrated into their work processes. Although the system documentation did not specifically allude to draft data being uploaded to the Data Directory, it was agreed that encouraging data managers to use the portal would require further thought about how synthetic data could be imported, and distinguished from actual study data.

This example also demonstrates another engaging feature of misusability cases: its point of application. Unlike many Security and Usability Engineering techniques which assume their application very early in the design process, misusability cases can be applied at a comparatively late stage. While deferring usability and security design techniques until late in the design process should not be condoned, many teams dedicate significant time and resources to understanding the complexity of a problem domain, leaving themselves little time for applying either Security or Usability Engineering techniques. Critics might argue that misusability cases encourage, rather than discourage, deferred thinking about security and usability, but this approach demonstrates that Security and Usability Engineering techniques can be situated with Software Engineering practice. As a corollary, it is hoped that this leads to a re-think about the utility of said techniques.

The misusability case example illustrated how security issues can lead to a scope of analysis review. For example, the documentation-related goals and assets that were deemed to be out of scope for the Data Directory proved to be relevant to this scope of analysis when put in context with the misusability case. In hindsight, therefore, it was necessary for such goals to be present; this included making explicit who was responsible for ensuring their satisfaction, and how these goals could impact goals and obstacles which were within scope.

The example also highlighted the importance of ensuring that unresolved obstacles were also assigned to responsible agents. Previous work in responsibility modelling has considered roles held by responsible agents towards securing systems, and modelling responsibility relationships between these agents, e.g. [236]. Vulnerabilities are not, however, considered something that can be brought to account. Although risk management approaches deal with the idea of *transferring* unmitigated responses to one or more agents, assigning ownership of obstacles during the early stages of design can ensure that usability-related vulnerabilities are promptly addressed. Fléchais and Sasse [99] argue that assigning liability motivates the assigned stakeholders; because failure to act responsibly damages both the project's assets and its reputation, this reputation loss may lead to loss of trust in the whole system.

8.5.4 CAIRIS modifications

In addition to the CAIRIS modifications made to support assumption personas and misusability cases, only one additional modification was deemed useful for this intervention: the addition of a short-cut menu in the persona dialog box, associated with the persona narrative text boxes. This modification allowed the persona characteristics associated with a particular persona narrative section to be quickly added, modified, or removed. This menu also allowed assumption persona models to be visualised, and filtered by both the persona name and the behavioural variable type associated with the narrative.

8.6 Specifying learning

Based on the results of this case study, the following lessons were taken away to inform subsequent research and case studies.

8.6.1 More engaging artifacts

One of the difficulties in completing this study arose from the lack of engagement with the project team. Although the project team appeared to be genuinely interested in IRIS and the analysis being carried out, their time was too limited to properly integrate this analysis into the day-to-day running of the project.

The success of the misusability case exercise suggests that focusing on the impact of non-security design decisions is a more effective technique for engaging developers in security issues rather than relying on fear of more generic threats, particular when these threats may or may not be relevant to the scope of analysis.

8.6.2 Focus on adding value

This case study reinforced the need for innovative thinking to ensure that important security issues were built into the system. This is an important finding as the Usability, Security, and Requirements perspective techniques forming part of the IRIS framework were designed on the basis that they would be used early in the design process. In contrast, not only was this IRIS process carried out comparatively late, it also ran in parallel with other design activities.

These findings lead us to conclude that when an IRIS process is devised, its techniques should scale to working with less than optimal input data. Moreover, as highlighted in the previous case study, the process should attempt to carry out as much analysis as can reasonably be carried out without disrupting other project activities.

8.7 Chapter summary

In this chapter, we presented a case study describing how the IRIS framework was used to support the elicitation and specification of security requirements for a meta-data repository supporting the sharing of longitudinal medical study data.

Using Action Research as a framework, we described the characteristics of the context of intervention and, based on this, motivated an action plan to guide the intervention. As part of this plan, we presented two new design techniques — Assumption Personas and Misusability Cases — which deal with specific aspects of this intervention. We described the steps carried out during this intervention before evaluating the effectiveness of both these new techniques, and IRIS in general, for meeting the original objectives. Finally, we identified lessons which need to inform subsequent research and validating IRIS case studies.

Chapter 9

Case Study: A Plant Operations Security Policy

This chapter reports the results of a case study where IRIS was used to analyse the security and usability impact of a security policy for plant operations at a UK water company, and identify missing security policy requirements.

Building upon the findings of Chapters 7 and 8, we also present a new technique for better aligning personas with the design of secure and usable systems. This technique, *Persona Cases*, builds upon ideas developed in Chapter 8 to facilitate development of personas which are both grounded in, and traceable to, their originating source of empirical data.

Using the same Action Research methodology as that described in previous chapters, we discuss the characteristics of the context of intervention, before presenting the action plan for instantiating the IRIS framework, and the steps taken during the intervention. We conclude by evaluating the results of the intervention, and recommending lessons for further informing IRIS.

9.1 Research environment

The author was invited by the information security manager of ACME, the water company introduced in the first case study, to help elicit missing policy requirements for a security policy covering staff at water treatment plants.

The ACME information security manager, again, acted as the gate-keeper for this intervention, and the same policy as Chapter 7 was used for obtaining audio and visual data during the intervention.

9.2 Diagnosis

Unlike the previous intervention at ACME, the scope of the intervention was agreed over email rather than in person. This was necessary as initial results would need to be available within one month. Based on these discussions, the following factors were identified as likely to have an impact on the planned intervention.

9.2.1 StuxNet

Until very recently, SCADA and control software was not considered vulnerable to information security threats. TV series like *24* suggest that terrorists have the potential to hack into terminals on an electricity smart grid, and gain the ability to cut the power for an entire nation. The reality, however, especially in the water industry, is that different sites have only limited connectivity to the outside world, and this is via industrial networking protocols rather than TCP/IP. Security experts have also argued that the biggest threat to critical infrastructure is a malicious insider with knowledge of both industrial control software, and the domain this software is situated in [257].

In July 2010, reports began to appear of a virus, the StuxNet trojan, which appeared to be affecting SCADA software in industrial plants around Europe [59]. These initial reports of StuxNet shook up senior management at ACME for several reasons. First, the claim that the obscurity of industrial control software protected it from internet-facing threats was rebutted; the virus explicitly targeted the same type of SCADA software used by ACME. Second, by combining knowledge of zero day threats with a realistic means of spreading the virus, i.e. via USB sticks, the operating software running in plants no longer seemed as isolated as it once was. Finally, although the motivation of the attacker was, at the time, unknown, the technical sophistication of the virus suggested it was professionally developed to cause harm. While ACME didn't believe they were the virus' target, they were acutely aware that the impact of being infected was unknown. They did, however, agree that an effective means of mitigating the likelihood of being threatened was to devise a specific information security policy for those working in plant operations.

9.2.2 Plant operations

Although most of ACME's clean and waste water infrastructure was unstaffed, clean water plants and sewage works serving large urban areas were not. These plants were staffed by operators working as part of a shift system. When on-shift, these plant operators were responsible for the running of the plant, and the treatment operations controlled by it. For waste water plants, this ensured that the effluent leaving the plant met fixed quality targets. For clean water plants, as well as monitoring for water quality leaving the plant, operators also monitored the rivers and unstaffed sites feeding water into the plant.

Plant operators were acutely aware of the safety implications of clean and waste water treatment. Waste water effluent which was not properly treated could have a significant impact on the ecosystem and the food chain. The clean water treatment process was critical enough that quality warnings were automatically forwarded to ACME chemists and quality assurance teams. Plant operators were also made aware of the security implications of deliberate attacks on the clean water infrastructure. Information security communications were regularly sent to all ACME staff, and police periodically visited clean water sites due to the perceived risk of possible terrorist action. There was, however, a feeling held by the information security team at ACME that these plant operators perceived many of the threats described in information security emails and notices to be irrelevant.

9.2.3 The enterprise SCADA project

In Section 7.2.3, it was reported that much of the control system infrastructure used by ACME was ageing and, as a result a new automation strategy was under development. In the months between the end of that intervention and the start of this study, a prototype Enterprise SCADA system had been developed. This prototype was being evaluated at two different waste water treatment plants; this controlled the new *Advanced Digestion* equipment at these locations. This equipment aimed to use the methane by-products associated with waste-water treatment to generate electrical power; this could then be contributed back to the UK national grid.

The new security policy needed to cover both the existing infrastructure and the new Enterprise SCADA system as it was gradually rolled out to other parts of ACME. There were, however, two issues which needed to be considered when designing policy requirements for this system. First, access to stakeholders working in this project was limited. The project relied on external contractors, several of whom were paid a substantial hourly rate for their expertise. Their insight would be required for this intervention, but their time needed to be carefully managed. Second, a number of technical requirements had been stipulated by the Enterprise SCADA system manufacturer. At the start of the intervention, it was not clear what impact these might have on the security policy and the ability of ACME to enforce it without compromising the running of the new system.

9.3 Action planning

Based on the factors described in the diagnosis, it was determined that the process for eliciting missing security policy requirements would need to incorporate the following.

- The need to complete the intervention in a timely manner, to ensure that results were made quickly available to senior managers.

- Stakeholders, from plant operators to managers assisting in the design activities, needed to be engaged in the process without underselling or overselling the importance of security and usability in policy decisions.
- Design activities would need to be informed by the development of the new Enterprise SCADA system, and access to resources working on the Enterprise SCADA project would need to be carefully managed.

Completing a balanced security and usability analysis quickly required a new approach for eliciting Usability perspective concepts. There were not sufficient documentation resources available to develop assumption personas, and the fieldwork and data analysis activities associated with data-driven persona development was time-consuming. To speed up the persona development process without compromising the integrity of the analysis, we devised the *Persona Case* technique. This approach allows the sense-making activities associated with qualitative data analysis to be re-used for generating the persona characteristics introduced in Section 8.3.1.

In the section which follows, we introduce and describe the Persona Case technique in more detail.

9.3.1 Persona cases

9.3.1.1 Motivation

In the previous chapter, the Assumption Persona Argumentation technique was introduced for structuring the assumptions underpinning assumption personas. This approach allowed lightly structured assumptions to be initially prepared, and developed inline with other activities as the design process progresses. Many personas are, however, fully developed (or at least as fully developed as possible) at an early stage using empirical data. Consequently, challenging the validity of personas can be a risk when these are finally introduced to the larger project environment. A developer may be unhappy about some system feature which directly appeals to some aspect of a persona. One way the developer can argue his case is by refuting some aspect of the persona, weakening its legitimacy, and calling into question other characteristics of the persona as well. As previously discussed, Chapman and Milham argue that, as fictional artifacts, personas cannot be falsified and, therefore, disproved [46]. They further argue that appealing to the rigorous use of interviews and ethnographic research also fails to validate personas, as specific examples of data cannot be provided to prove their accuracy.

Although Chapman and Milham argue that interviews and ethnographic approaches are not sufficient to validate data-driven personas, these are considered necessary by the two dominant approaches for building personas. In the first of these, behavioural patterns are identified based on how interviewees cluster around particular behavioural variables elicited from empirical data [61]. In the second, factoids are culled from the empirical data, before a group-based affinity diagramming exercise is carried out to induce behavioural clusters [32]. In both cases, sense-making activities are used to induce behavioural clusters upon which narrative descriptions of personas are written.

The data analysis associated with affinity diagramming is analogous to the qualitative data analysis approaches used in social-science inquiry, such as Grounded Theory. However, rather than culling factoids directly from data, Grounded Theory involves coding data transcripts to elicit concepts relating to these questions. Moreover, although both approaches induce emerging themes from data, Grounded Theory also involves drawing out relationships between thematic concepts, creating memos with insights from this analysis, and preparing papers describing the theory resulting from the analysis; Corbin et al. [63] suggest that this latter act of writing clarifies thinking and elucidates breaks in logic. Finally, Grounded Theory is also supported by Computer Aided Qualitative Data Analysis tools; these can manage large data-sets, and support theory development by managing codes, memos, and automatically visualising emerging conceptual relationships.

In Chapter 7, we demonstrated how Grounded Theory could be used to develop a theory from which personas were derived. Although the validity of the personas was obtained by integrating them into the design of secure systems, the problem of validating personas remains. Even though the Grounded Theory artifacts were available during the building of the personas, they were not once the personas were used in practice to elicit requirements during workshops. This semantic gap between Grounded Theory artifacts and persona narratives became problematic when questions were raised about characteristics of one of the personas in this study; this subsequently led to the creation of a new persona.

In Chapter 8, we examined how Toulmin's work on developing practical arguments can be aligned with the characteristics of personas. If treating a persona characteristic as a defensible argument reduces the gap between empirical data and personas, aligning these claims with the argumentative discourse associated with Grounded Theory models may close it. To this end, we propose the *Persona Case*: a persona whose characteristics are both grounded in, and traceable to their originating source of empirical data. In the next section, we describe this technique in more detail.

9.3.1.2 Approach

Our approach for developing persona cases is driven from a Grounded Theory model; this model constitutes a collection of thematic concepts, and a set of relationships between them. The model is induced using data elicited from interviews or ethnographic research. The first stage involves identifying the most salient thematic concepts, and eliciting propositions associated with them. These concepts are selected based on the most grounded and networked concepts in the Grounded Theory model of core themes; these represent the key themes that the persona cases being developed need to explore. The propositions are based on the quotations associated with each of these selected themes. These propositions are factoids because while accepted as fact by the participants, these may not always be established. For example, the quotation *The worst thing that could happen on a sewage site is that it would flood something* could be reworded as the proposition *The worse case scenario on a sewage site is a flood.*

The next stage involves enumerating and arguing each relationship in the Grounded Theory model. This involves succinctly describing the claim (or claims) justifying each relationship; this claim represents a potential characteristic of a persona. To justify this characteristic, propositions are selected as its grounds or warrant, and the modal qualifier will be associated with this based on the analyst's confidence in the relationship. Propositions which may rebut this characteristic are also recorded here. Such a rebuttal might arise if there is debate or disagreement about some aspect of the Grounded Theory model. Also associated with each characteristic is the type of behaviour the characteristic represents.

The final stage involves writing a supporting narrative for each section characterised by the behavioural variable types. This narrative should be commensurate with the elicited persona characteristics and, as such, act as supplemental validation of the personas; if a commensurate narrative cannot be written then the characteristics and propositions should be reviewed in line with the Grounded Theory model. There may be a number of reasons why a narrative cannot be written to be consistent with the elicited characteristics. One reason is that not all characteristics may be relevant because they cover poorly grounded themes, or the characteristics are not relevant to the context of analysis. In this case, it is reasonable to omit characteristics if incorporating them leads to the persona becoming more elastic. Another reason is that multiple personas may be needed to reflect the elicited persona characteristic. In this case, characteristics should be sorted into natural groupings using affinity diagrams; these characteristic groups then form the basis of separate personas.

9.3.2 Action plan

The process illustrated by the UML activity diagram in Figure 9.1 was orchestrated for eliciting additional requirements for the plant operations security policy. In the following sections, we describe an action plan for developing this policy, including details about how the Persona Case technique is used to build personas for this intervention.

9.3.2.1 Scoping

Existing documentation about ACME information security policies is provided and, on the basis of this, an initial rich picture diagram of the policy scope is defined. This diagram is discussed remotely with the gatekeeper who, in turn, solicits comments from the ACME policy stakeholders. Although preparation for the Fieldwork stage commences during this stage, the final rich picture diagram acts as the delimiter of scope during later stages.

On completion of this stage, the key roles the policy needs to be written for, together with the environments for subsequent analysis, are determined.

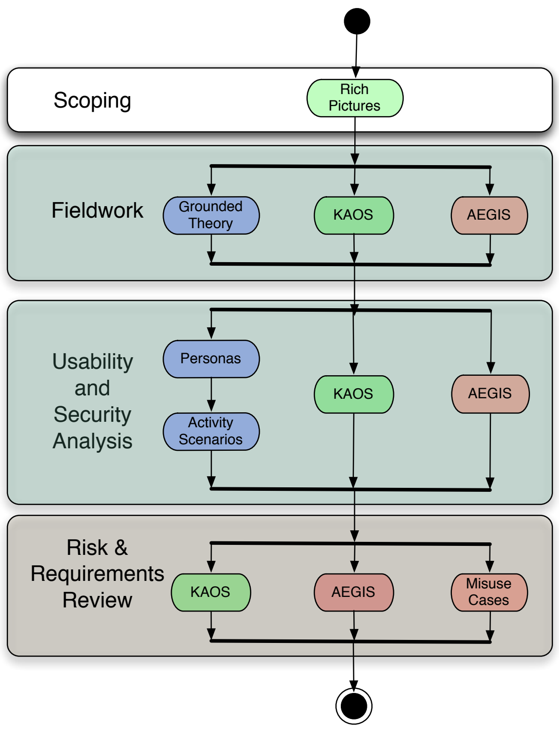


Figure 9.1: Instantiated process for the plant operations security policy

9.3.2.2 Fieldwork

The objective of the Fieldwork stage is to develop a qualitative model of plant operations security; this is used to derive one or more personas representing plant operators for later design activities. To ground this model, in-situ interviews are held with plant operators at different water treatment works. These interviews are recorded, and the transcripts analysed using Grounded Theory. The results of this analysis are a qualitative model of plant operations security perceptions. In parallel with the qualitative model development, existing policy documentation is analysed to elicit a goal tree of policy goals for different environments. Assets are also elicited from this documentation, and, where necessary, concern links are associated between goals and assets.

9.3.2.3 Usability and security analysis

Like the previous stage, a number of parallel activities take place during Usability and Security Analysis.

First, the persona case technique is applied to develop personas for the roles identified during the scoping phase. For each environment the personas are associated with, tasks are modelled to describe the typical activities these carry out.

Second, the goal model for each context continues to be developed, and new policy requirements are elicited using a three-step process. In the first step, the obstacles obstructing these goals are also added. Where possible, goals with the ability to mitigate these obstacles are elicited; these goals constitute missing policy requirements.

The second step involves reviewing the empirical data used to develop the qualitative model and personas to identify possible vulnerabilities. Any identified vulnerabilities are entered into CAIRIS and associated obstacles are added to the goal model. After identifying the goals obstructed by new obstacle, the obstacles are refined until leaf vulnerabilities are identified that can be mitigated by policy goals.

The third step involves using both the empirical data and domain-specific threat models to identify candidate attackers and threats these might carry out. Like the second stage, identified threats are associated with obstacles and obstructed goals, before refining the obstacle and, where possible, identifying mitigating policy goals.

At the end of this stage, any unresolved vulnerabilities and threats are, where appropriate, combined to elicit new risks; misuse cases are also modelled to describe these risks in their respective contexts. Finally, roles are assigned responsibility for the satisfaction of leaf policy goals.

9.3.2.4 Risk and requirements review

The final stage involves holding a focus group with the project stakeholders and presenting the misuse cases encapsulating the unmitigated risks. These misuse cases are described in context with the assets they threaten, tasks impacted by these threatened assets, and personas affected as a result. The objective of the focus group is to discuss each risk in order to agree accepted mitigating controls from which new policy goals can be elicited. Following the focus group, the IRIS model is revised to reflect the selected risk responses and new policy goals.

Once the policy goal tree is complete, this is reviewed with the information security manager to resolve any unmitigated risks, and confirm both the policy goals and the responsibility relations are acceptable.

9.4 Action taking

In this section, we describe how the steps in the action plan were used to guide the actions taken as part of this intervention.

9.4.1 Scoping

The rich picture diagram in Figure 9.2 was developed and maintained at Oxford, although it was distributed to ACME stakeholders via the information security manager. The feedback provided by stakeholders involved word changes which seemed minor, but were semantically significant to ACME. For

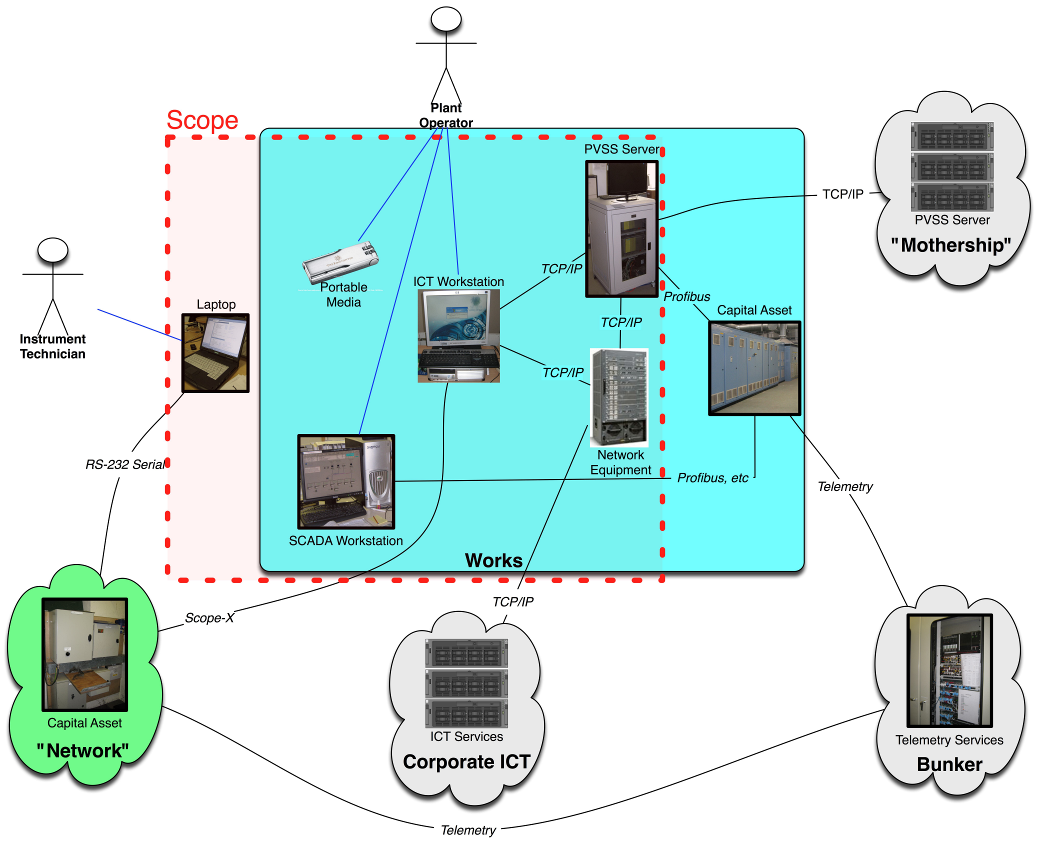


Figure 9.2: Finalised context diagram bounding the analysis for the security policy

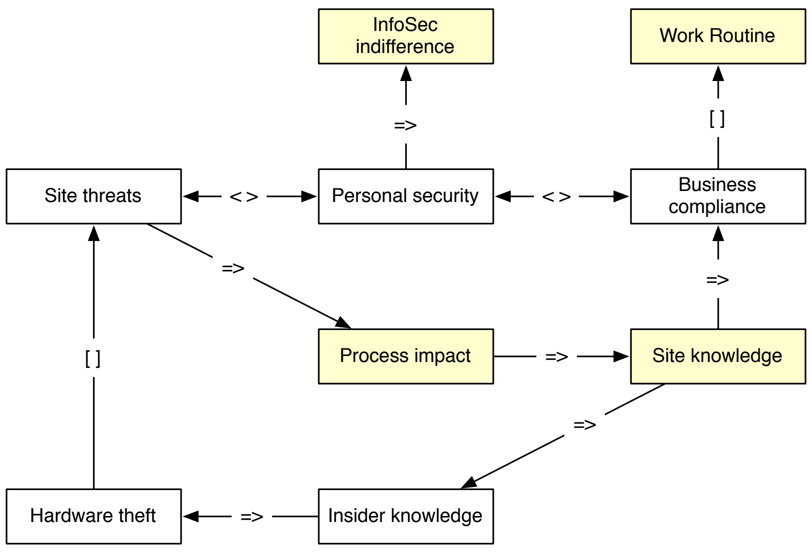


Figure 9.3: Grounded theory model of plant operations staff security perception

example, an association was drawn between one system in scope to another box named after the physical location of ACME's head office, where the telemetry group and their servers were located. Although the association was valid, the box was renamed to *Bunker* to emphasise the data flow to the telemetry group rather than other, related, groups also located at the physical location; the name was commonly used to refer to the group because they were located in a secure, bomb-proof building.

As the diagram was circulated, more objects were added. These included additional items which stakeholders wanted included in the policy, such as USB sticks, and related users and systems, such as instrument technicians and laptops. To mitigate against the risk of scope creep, a red bounding box was drawn on the diagram to indicate the agreed scope of the policy.

Although it was intended that the diagram would be agreed before commencing the Fieldwork stage, time constraints meant that this was not finalised until after the in-situ interviews had been completed. As a result, some images were based on photos taken during the site-visits.

Upon final agreement, a one sentence summary of the policy scope was agreed. This was *All ICT infrastructure in support of Enterprise SCADA, Telemetry and Control Systems indicated in the rich picture.*

9.4.2 Fieldwork

Four different water-treatment works (two clean water and two waste water) were visited to hold in-context qualitative interviews with plant operators and related stakeholders. Although these interviews were largely open-ended, high level questions dealt with the nature of work undertaken, including what

Date	Location	Roles interviewed
05/08/10	ACME Waste Water Treatment Plant 1	Plant Operations Manager, SCADA Engineer
09/08/10	ACME Waste Water Treatment Plant 2	Plant Operator, SCADA Engineer
09/08/10	ACME Clean Water Treatment Plant 1	Plant Operator
10/08/10	ACME Clean Water Treatment Plant 2	Plant Operator

Figure 9.4: In-situ interview summary

plant operators were responsible for, who they worked with, and how they obtained help if necessary. Plant operators were also asked about important artifacts and activities, and the sort of problems they often faced. A summary of the interviews held, and the roles interviewed is given in Figure 9.4.

Following the interviews, qualitative data analysis, using Grounded Theory, was carried out on the interview transcripts. Following an exercise of open and axial coding, approximately 200 initial thematic concepts were refined down to 61. Conceptual relationships were visually modelled in one central model containing the most networked and grounded concepts; this model is illustrated in Figure 9.3. The boxes in yellow represent concepts which were central nodes to smaller sub-models.

To validate this analysis, the Grounded Theory models were presented to another researcher. Following this presentation, a number of minor adjustments were made to concepts, and a slide presentation of the Grounded Theory models was sent to the ACME information security manager for his comments.

In parallel to the fieldwork activities, goal modelling commenced. The documents used to drive this activity were a draft security policy that ACME had prepared, an ACME information handling guidelines document, and ACME's organisational security policy. As the aim of the intervention was to elicit missing requirements from the first of these documents, this was the primary document used to elicit goals. For each policy recommendation in this document, a goal was defined. As goals were elicited, a goal tree was induced based on statements which relied on the satisfaction of other statements. Where supplemental documents were referenced, the referenced statements also formed the basis of goals. For example, the *Secure Roles* goal was derived from the applicability section of the draft policy document, and was defined as *The security responsibilities of ACME employees, partners, and contractors responsible for installing, configuring, implementing, and operating ICT infrastructure supporting SCADA, Telemetry, and Control Systems shall be specified and communicated to all relevant parties.* In order to satisfy this, the goals in Figure 9.5 would need to be satisfied.

Although the *Responsibility clarification* goal existed in the same section as *Secure Roles*, *Router misuse prevention* was listed in a section for routers.

Name	Definition
Responsibility clarification	When staff responsibilities and applicability is unclear, the Information Security Manager shall be consulted for advice.
Router misuse prevention	All ACME employees responsible for maintaining information security measures within their area of responsibility shall ensure routing devices are not compromised through misuse of other information processing facilities.

Figure 9.5: Policy goal examples

It was also necessary to omit 8 policy statements from the draft policy because they were either out of scope, or were liable to be refined to goals which were. An example of this was the statement *Wireless systems shall be securely deployed using best practice guidelines.* From the perspective of this policy, the deployment of generic wifi equipment was not within the scope.

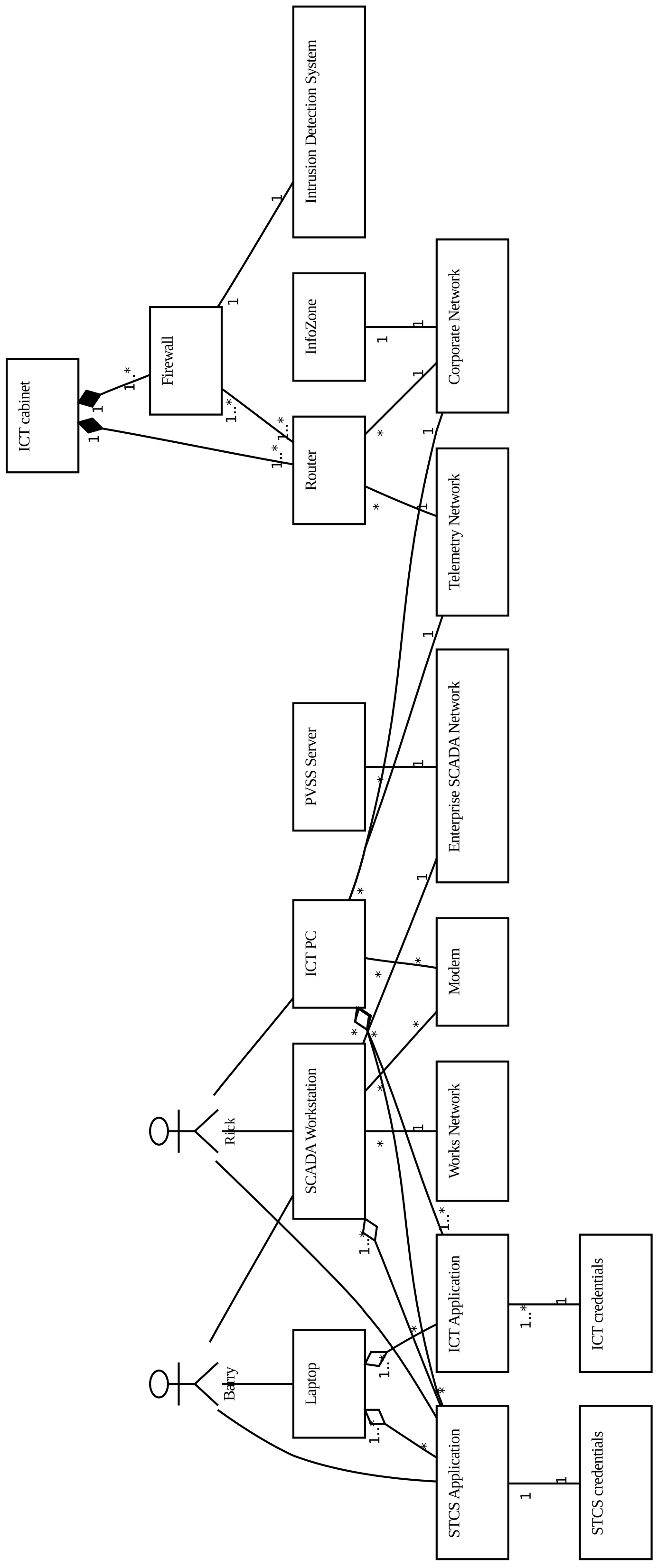


Figure 9.6: Asset model associated with the security policy

In parallel with other activities, an asset model (Figure 9.6) was progressively developed.

9.4.3 Usability and security analysis

The Usability and Security Analysis stage took place over a period of two and half weeks following the development of the Grounded Theory model. During the Fieldwork stage, the asset model (Figure 9.6) progressively developed was mature enough to form the basis of analysing possible security issues.

Two sets of activities took place within this phase, both of which were driven from the data collected in the previous stage. The first involved developing personas and activity scenarios. The second involved carrying out vulnerability and threat analysis. The actions taken in each activity set are described in more detail in the sections below.

9.4.3.1 Persona development

A plant operator persona — Rick — was developed using the Persona Case technique described earlier in this chapter. Rick was developed on the basis of 32 persona characteristics (11 activities, 7 aptitudes, 11 attitudes, 3 motivations). Each characteristic reflected a conceptual relationship, and underpinning these were 126 propositions associated with 34 concepts.

In the following sections, we illustrate the steps taken to apply the technique by describing how the follow paragraph was derived from Rick's attitude section:

Rick isn't particularly concerned about people trying to hack into the SCADA system he uses. "The only way the SCADA will get infected," Rick says, "is by an instrument tech plugging a virus infected laptop into it."

Summarise propositions A spreadsheet was used to catalogue propositions derived from quotations associated with the selected thematic concepts. Each proposition contained two components: a sentence containing the proposition itself, and a phrase acting as an abstract for this sentence. In our example, the paragraph of interest was derived from the following three quotations, Q1, Q2, and Q3; these quotations were coded with the *Information Security Indifference* (Q1), and *Island Mentality* (Q2, Q3) concepts:

- Q1: *Am I worried about people that want to hack my system? No, not really.*
- Q2: *Of course, security has always been an issue. Predominantly, what you'll find is that our systems are stand-alone, and they are not connected to the network. If they are then, in very limited cases, there could be an island network where a screen over here is looking after a site 8 miles up the road.*
- Q3: *If they work on a system and didn't have that, if they had to bring in their laptop with a cable and connect it in then, yes, if some guy infected their own laptop by doing other things, this could*

Component	Definition
Relationship	*Information Security Indifference* is a cause of *Island Mentality.*
Characteristic	SCADA isolation makes hacking unlikely.
Behavioural Variable Type	Attitude
Grounds	II-5, IM-3
Warrant	IM-4
Backing	Island Mindset concept
Qualifier	Probably
Rebuttal	None

Figure 9.7: Characteristic argument example

infect your own work. That's the only thing that could get infected because it's linked to all the other systems.

The propositions representing Q1, Q2, and Q3 are *Not worried about people trying to hack the SCADA systems*, *Systems are stand-alone because they are not connected to a network*, and *Systems are infected only if engineers connect infected laptops* respectively.

Duplicate or superfluous quotations were omitted from this step. Examples of such quotations included plant shift-specific comments, and references to clean or waste water treatment operations out of scope for this study.

Argue characteristics A sheet was added to the previously created spreadsheet to argue each association between the thematic concepts. Each row in this sheet represented a relationship, with columns for the characteristic claim being made, the behavioural variable type, grounds, warrant, backing, modal qualifier, and possible rebuttals. The modal qualifier was subjectively assigned based on the grounding of the underlying quotations; the greater the grounding, the more confident the qualifying noun was. Figure 9.7 describes the components supporting the persona characteristic *SCADA isolation make hacking unlikely.*

Like the previous step, not all relationships were included in this analysis. Reasons for omission included a lack of grounding, and relationships which were grounded in overly site-specific data. Another reason was that the analysis itself led to new insights into the Grounded Theory model which, when investigated further, rendered a relationship superfluous.

Task	Description	Environments
Take chemicals delivery	Take delivery of a scheduled order of chemicals.	Day
Resolve reservoir alarm	Resolve operational alarm due to high reservoir water level.	Day, Night
Broken instrument alarm	Fix flatlined trend.	Day, Night

Figure 9.8: Tasks performed by plant operator persona

Write persona narrative Information about personas, including their narrative and characteristics, was stored in CAIRIS. Because the CAIRIS model database supported the meta-model additions described in Section 8.3.1, the spreadsheet was exported to multiple Comma Separated Value (CSV) files. A Python script was created to map these sheets into their corresponding model elements, and automatically generate persona characteristics. Rick's validating narrative text was entered into CAIRIS, and associated with the generated persona characteristics.

9.4.3.2 Activity scenario development

Once Rick had been developed, the empirical data was used to derive three activity scenarios; these were modelled as tasks in CAIRIS as shown in Figure 9.8. Where tasks were carried out in more than one environment, the steps carried out by the persona varied. For example, the narrative associated with the *Resolve reservoir alarm* task during the day was as follows:

Rick looks at the SCADA monitor nearest to him and notices that the levels of the reservoir nearby are unusually high. When the level gets too high, the entire works need to be shutdown. In this situation, Rick knows exactly what to do. After stopping the alarm, Rick logs into the ICT PC next to the SCADA workstation, and clicks on the Xtraview icon. After logging into Xtraview, he finds the location of a pumping station 10 miles upstream on the map and connects to it. After a few moments, he masters the main pump before switching it off. Rick then returns the pump to its normal slave setting before shutting down Xtraview. The alarm periodically starts and stop again but, after about an hour, the reservoir level normalises again.

Although the above task was identical for both the Day and Night environment, there were a number of variations in the *Broken instrument alarm* task due to the necessity for on-call instrument technicians to resolve a problem that on-site fitters could have fixed during working hours.

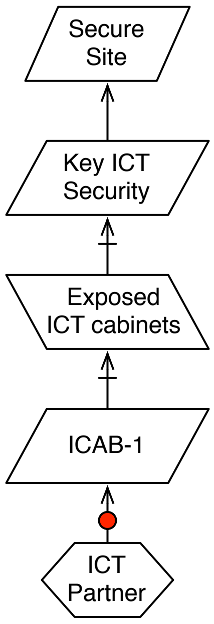


Figure 9.9: Goal tree associated with exposed cabinets vulnerability

9.4.3.3 Vulnerability and threat analysis

At this stage, the goal tree was analysed to find obvious vulnerabilities which required further analysis. Although no obstacles were forthcoming, a number of goals suggested policy requirements needing to be present in order for it to be satisfied. One such requirement was *Authorised STCS network point data shall be available to authorised Plant Operators on InfoZone.* This requirement arose from a goal stating that information about authorised network points should be available to authorised plant staff; this was necessary to allow plant staff to identify network points which might be unauthorised.

Although no obstacles were obvious from the goal tree, examining the empirical data collected during the Fieldwork stage identified the four vulnerabilities described in Figure 9.10.

An example of how these obstacles were integrated into the goal tree was provided by Figure 9.9. In this example, the newly created *Exposed ICT Cabinets* obstacle was introduced and anchored to pre-existing goals for securing the physical infrastructure. This particular obstacle was mitigated by the requirement *Key ICT equipment shall be stored in a restricted access computer room.* As the figure indicates, this requirement references the ICT cabinets asset, the short-code to which is ICAB.

The threat analysis step involved identifying candidate attackers, and possible threats these might carry out. From the empirical data, two classes of attacker were identified. The first related to thieves who would attempt to break into water treatment works to steal scrap metal or other plant equipment. Several plant operators expressed concern about these attackers because the damage to monitoring

Name	Description	Affected assets
Unknown applications	PC and PC applications with no clear provenance may be more liable to exploitation. These may be used as an attack vector for obtaining sensitive data or credentials.	ICT PC
Ubiquitous identity and knowledge	The user id component of ICT access credentials are multi-use. This knowledge is commonly known and, occasionally, shared.	ICT credentials
Redundant hardware	Legacy PC hardware may contain sensitive data or credentials.	ICT PC
Exposed ICT cabinets	Equipment cabinets are easily accessible to unauthorised personnel.	ICT cabinet

Figure 9.10: Vulnerabilities and affected assets

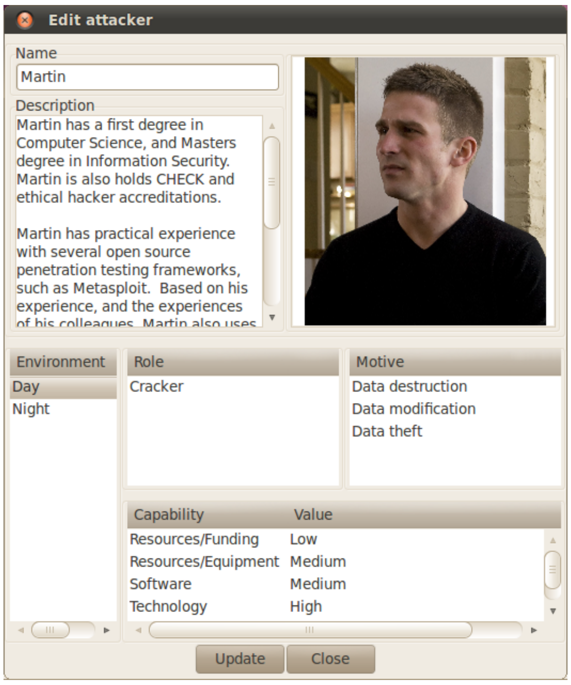
equipment they could cause is usually much greater than the value of the items stolen. Plant operators were also worried about their own personal safety should they be required to confront them out-of-hours. A *Kit theft* threat was defined to model the impact of this attack.

The second class of attacker arose from a general indifference of plant operators and engineers towards information security threats. Even after describing the recent reports of StuxNet, participants remained unconvinced that *hackers* were as potent a threat as the press and the information security communiques would have them believe. Consequently, it was decided to model an attacker based on a penetration tester commissioned by ACME. A number of open-source resources and texts on professional penetration testers were consulted and used to develop an attacker profile for a professional penetration tester, as illustrated in Figure 9.11. Based on this attacker, several threats were identified, such as war-dialling modems, *footprinting* to determine information about possible ACME network services, and the enumeration of possible passwords using known defaults for software applications.

Although several obstacles were elicited based on these threats, no mitigating requirements were identified without further discussing the threats and their consequences with ACME stakeholders.

9.4.4 Risk and requirements review

Candidate risks were identified by finding instances where threats might exploit unresolved vulnerabilities. Based on these, seven different risks were identified. At this stage, a specification document was generated using CAIRIS and sent to the ACME information security manager for distribution to other policy stakeholders. Based on the initial feedback, the instrument technician persona (Barry) from the first case study was imported into the model to examine the impact that this analysis may have on his



Name: Martin

Description:
Martin has a first degree in Computer Science, and Masters degree in Information Security. Martin is also holds CHECK and ethical hacker accreditations.

Martin has practical experience with several open source penetration testing frameworks, such as Metasploit. Based on his experience, and the experiences of his colleagues, Martin also uses

Environment	Role	Motive
Day	Cracker	Data destruction
Night		Data modification
		Data theft

Capability	Value
Resources/Funding	Low
Resources/Equipment	Medium
Software	Medium
Technology	High

Figure 9.11: Penetration tester attacker profile

activities. In addition to importing Barry, two of his tasks (Modify PLC Software and Modify SCADA HMI software) were also imported.

This analysis was presented to ACME stakeholders at their head-office on 2nd September 2010. These stakeholders were operational managers responsible for plant security and a representative ICT manager. Because only a limited amount of time was available, the presentation of the analysis was centred around a discussion of the risks of most interest to ACME: a virus-infected SCADA workstation, and a site-break in. The misuse case associated with each risk was presented, discussed and, based on the outcome, mitigating strategies were proposed. For the first risk, a policy requirement was added to remove USB access to SCADA workstations. Responsibility for the second risk was provisionally passed to the ACME facilities management department.

After updating the CAIRIS model based on these discussions, a revised specification document was re-issued to ACME. Because of the limited time available on the 2nd September, and the unavailability of the information security manager, a more detailed review of the analysis took place at one of ACME's offices on the 28th October 2010. This was a one-to-one session with the information security manager where the goal model and elicited policy requirements were validated, and the risk analysis was reviewed. The purpose of this session was to ensure that all goals and requirements were assigned a responsible role and all responses were elicited for each risk.

After this session, the CAIRIS model was again revised, and the final specification document was re-generated and sent to ACME.

9.5 Evaluating

We believe the outcome of the intervention from a research perspective successfully validated the IRIS process used for this case study. One of the testaments of this validity was the ability to complete the study comparatively quickly without compromising the quality of the artifacts created. A further testament were indications from the information security manager that both the proposed recommendations would be incorporated into ACME's operations. Following the end of the intervention, ACME requested a copy of CAIRIS and the IRIS model generated during the intervention for their own use.

Figure 9.12 summarises the concepts elicited and specified on completion of the intervention.

Concept	Day	Night	ALL
Asset	18	18	18
Attacker	4	3	4
Countermeasure	0	0	0
Domain Property	N/A	N/A	1
Goal	102	4	106
Obstacle	20	0	20
Persona	2	2	2
Requirement	N/A	N/A	8
Response	1	1	2
Risk	6	1	7
Role	N/A	N/A	10
Task	5	4	5
Threat	8	1	9
Vulnerability	8	8	8
ALL	174	42	200

Figure 9.12: IRIS concepts specified on completion of the plant operations security policy intervention

As the table shows, concepts were present in either one or both of the *Day* and *Night* environments. The vast majority of the goals elicited were based on the Day environment; this reflects many of the day-to-day concerns that participants had with regards to security policy coverage. Similarly, although the attackers, vulnerabilities, and assets tended to be invariant, most threats were resident within the

Day environment. This is because the threats most evident from the empirical data were based on attacks expected to take place during daylight hours. The sole exception was a war-dialling attack that the penetration tester attacker (Martin), deliberately performed out-of-hours.

In the following sections, we evaluate the different stages of the IRIS process in more detail.

9.5.1 Scoping

Given that an ill-defined scope of analysis was a significant risk, it was less than ideal that data collection occurred before the scope had been formally agreed. However, this risk proved to be quite low for several reasons. First, ACME's own work preparing a draft policy document meant that the scope was reasonably firm before the start of the intervention. Second, pre-existing data and domain knowledge from the first ACME case study meant that suitable imagery and terminology could be used in the rich picture diagram; where terms were incorrectly stated or mis-used, these were quickly picked up. Third, because the ACME information security manager was heavily involved in facilitating the site visits, he was also available to quickly comment on diagram revisions at the same time. Finally, because of the StuxNet factor, ACME were keen on ensuring that problems agreeing scope did not hold up subsequent activities.

9.5.2 Fieldwork

Because of the compressed time-scales, ACME were supportive in providing timely access to sites and interviewees. A classic Grounded Theory exercise was carried out where interview data was immediately transcribed (where time permitted) following each interview, coding was carried out, and the insights were used to inform possible questions for subsequent interviews. Carrying out goal modelling also assisted in this sense-making exercise because analysing the policy documentation helped put some of the data collected in context, and vice-versa.

Focusing on security design activities at the same time as fieldwork also heightened awareness of possible threats and vulnerabilities at elicitation time. For example, on one site-visit, questioning the purpose of one particular PC led to the discovery that not only was it superfluous to plant operations, but the modem device attached to it was vulnerable to war-dialling attacks. On another visit, a chance conversation about a car driving up to the gate on a CCTV camera led to the discovery that the plant had a second gate, and that the access control system for it was particularly weak. From this, we believe that fieldwork makes two important contributes to security design. First, de-familiarisation associated with in-situ interviews leads to the identification of hitherto unseen affordances. Second, opportunities for identifying and analysing vulnerabilities can happen at any time and, quite often, such insights might have otherwise remained hidden.

9.5.3 Usability and security analysis

An interesting feature of this intervention was that it was possible to elicit useful information about attackers and threats without attempting to elicit threats up front. This is because threat analysis could be informed by the sense-making activities associated with other analysis. IRIS' approach to threat identification is an improvement over security design methods relying on solely on anecdotal information from stakeholders or security experts to derive threats, e.g. [97, 70] for two reasons. First, threat elicitation is not solely contingent on participatory approaches, which rely on getting stakeholders together in a single location. Second, the task of eliciting *attackers* followed by *threats* is easier than trying to elicit attacks in their own right. While the empirical data can point to possible attackers, further research is often necessary to determine what threats these attackers can give rise to and, as a result, what assets might be threatened.

Unlike the other case studies, where requirements analysis led to insights into security analysis, this study illustrated how the reverse could also be true. This was illustrated by the lack of obstacles apparent before analysing the empirical data for vulnerabilities, even though it could be later seen how these led to the obstruction of several goals. Unlike vulnerabilities, however, many of the threats could not be goal-anchored. For example, one of the threats, *footprinting*, entailed Martin carrying out activities to determine information about the network and systems on it. However, although these activities provide information about vulnerabilities to a penetration tester, they are not easily construed as attacks; the goals related to distinguishing normal and abnormal network traffic were not within the scope of this policy analysis. Ultimately, this threat contributed to a risk where footprinting could be carried out due to the vulnerability of incomplete fire-wall rules, where assets were within scope were impacted. Like the threat, the obstacle related to this vulnerability could not be anchored in the goal model. This example illustrated that, in some cases, it is necessary for stakeholder intervention to decide an appropriate risk mitigation strategy.

9.5.4 Risk and requirements review

One of the benefits of the persona case argumentation structure was spotting cases where participants used fallacies to undermine personas as a means of rebutting possible policy requirements. When the misuse cases were presented, there was sometimes a tendency for participants to rebut one mitigation by discussing possible attacks to different assets. For example, after indicating that Rick would not typically use USB sticks when connecting to a SCADA workstation, one participant highlighted that there were still ways that attackers *could* get malware onto a system, while another participant indicated that an engineer *might* want to use USB to install a software update. However, both examples were non-specific, while the second example also glossed over the possibility that an engineer could use a CD or DVD to carry out the same update.

Misuse cases also appeared to be useful for spotting more general fallacies made when arguing against the tenability of a risk. In particular, there was also a tendency to undermine the impact of the threat or the severity of the vulnerability by focusing only on the threat's likelihood and the asset directly under threat. During discussion of the *Site break-in* misuse case, some participants highlighted the limited number of staffed sites within the ACME network coupled with the relatively high frequency of PC theft as a reason why incorporating policy requirements to mitigate this risk might be untenable. However, when it was highlighted that the PCs themselves were less important than the monitoring they facilitated, and that the quantity of staffed and unstaffed sites had no bearing on the impact of the risk, it was agreed to transfer responsibility of the risk to ACME's facilities management department rather than ignore it altogether. When discussing the risk further during the follow meeting with the information security manager, it was highlighted that transferring the risk in its entirety was inappropriate. The physical site security goals were reviewed to determine which were the responsibility of the facilities management department, and which needed to be pro-actively managed by the information security team.

9.5.5 CAIRIS modifications

Because of the number of goals elicited, it was necessary to modify the goal model view. Support was added for creating refinement links to existing goals and requirements directly from the goal model; this facilitated the sense-making activities which would inevitably occur. To ensure that responsibility relations were easily captured, changes were also made to allow these to be added on an ad-hoc basis from the goal model.

9.6 Specifying learning

Based on the results of this case study, the following lessons were taken away to inform subsequent research.

9.6.1 Qualitative model re-usability

The data used to derive Rick was sufficiently homogeneous that a single persona encapsulated most of the characteristics elicited. Although not described in this dissertation, we further validated the Persona Case technique by using it to derive Personas from the NeuroGrid Grounded Theory model. Although these were never used, discussions with stakeholders from the study in question suggest that the personas were accurate archetypes of their respective user communities.

These findings suggest the re-use potential for other Grounded Theory models towards the design of personas. For example, existing theories about privacy for different domains might be easier to digest by software developers in the form of a persona than a conceptual model and accompanying discourse.

These extensions do not, however, guarantee that a Toulmin model is the best bridge between all possible qualitative models and User-Centered Design artifacts. Further research into applicable techniques from the Design Rationale community is needed to determine their usefulness.

9.6.2 Incorporating stakeholder activities into the intervention diagnosis

Although personas, tasks, and misuse case narratives were found to be engaging during the workshop, some participants still found it difficult to let go of inappropriate generalisations and stereotypes. A likely reason for this might have been the limited contact that participants had with the documented analysis prior to the workshop. The diagnosis for the intervention highlighted the compressed time scales available, but it did not highlight that other workshop participants may have been equally pressed for time. Fortunately, focusing the discussion on the misuse case diagrams ensured that the important findings of the analysis could be imparted. Nevertheless, it is useful to consider the roles and activities of participants taking part in focus groups at the planning stage to ensure that an appropriate dissemination method for the analysis can be found for all stakeholders, rather than just the intervention's gatekeeper.

9.7 Chapter summary

In this chapter, we presented a case study describing how the IRIS framework was used to elicit security policy requirements for a plant operations information security policy at a UK water company.

Using Action Research as a framework, we described the characteristics of the context of intervention and, based on this, motivated an action plan to guide the intervention. As part of this plan, we presented a new design technique — Persona Cases — which builds upon ideas developed in Chapter 8. We described the steps carried out during this intervention before evaluating the effectiveness of both Persona Cases, and the IRIS in general, for meeting the original objectives. Finally, we identified lessons which need to inform subsequent research.

Chapter 10

Conclusion

In this chapter, we synthesise the results of this dissertation, and summarise the important findings. We then review and critically analyse the thesis to determine how successful it is at answering the research questions posed in Chapter 1, and whether the contributions arising from this dissertation answered the research questions. Finally, we briefly discuss future directions for extending the research carried out in this dissertation.

10.1 Key findings

In the following sections, we summarise the key findings this dissertation has contributed towards the design of secure and usable systems. Building on these findings and the results of the case studies, we then re-visit the original IRIS framework and propose modifications based on the results of the validating case studies.

10.1.1 Environments as a design concept

Supporting the concept of *Environment* allowed IRIS processes to specify and reason about the elements of several different contexts of use for a given system. Based on the case studies, three observations can be made about the usefulness of specifying environments during requirements elicitation and specification activities.

First, different environments should be elicited at the same time as the system is scoped. During the diagnosis stages of the first case study, it was felt that the environments of most relevance would be situated around the different types of control software the repository needed to support. It was not until after the scope was agreed that it was discovered the most relevant environments would be activity, rather than artifact, based.

Second, environments of most value tended to be social or cultural rather than physical. When considering the NeuroGrid specification exemplar in Chapter 4, we noticed that security properties associated with certain assets have markedly different security properties in different environments. In the case studies, the security properties varied less considerably, and the environments were used to compartmentalise the analysis of activities according to the context of most relevance. Occasionally, however, some discussion arose by comparing the variances between tasks carried out in different social contexts, e.g. the *Maintain Telemetry Software* task described in Chapter 7, or discussing how the tasks carried out in one environment which had an impact in others, e.g. the *Batch import sensitive meta-data* misusability case in Chapter 8.

Third, although the meta-model supported the composition of an environment from multiple environments, this did not prove to be very useful in practice. Although the environments modelled in the case study examples were non-trivial, they were also distinct enough that combining them added little value to any analysis carried out. However, as indicated in the previous point, reasoning about dependencies between two environments was occasionally useful; in the first case study, it was also highlighted that attackers might exploit knowledge about environments to cause a shift from one environment to another or, as the misusability case example in Section 8.5.3 demonstrated, how a risk in one context leads to a subsequent risk being introduced in another.

10.1.2 Scope and security

During participative sessions in each of the case studies, it was noted that several interesting threats and vulnerabilities tended to form around boundaries of scope. This was noticed during obstacle modelling in Chapter 7, investigation of the research environment in Chapter 8, and discussions around site security in Chapter 9. Some of these issues were identified by the facilitator and deemed out of scope by participants; in some cases, these issues were identified by the participants themselves. For example, during the final workshop in the first case study, obstacles were modelled to determine how an inside attacker might manipulate the internal state of a telemetry outstation. In this case, the obstacle did not lead to the elicitation of a vulnerability because it was outside the scope of analysis. Nonetheless, its capture meant that information about this general organisational vulnerability was available to ACME, despite the lack of support for vulnerabilities outside of the planned system's scope.

One way of dealing with such vulnerabilities was assigning responsible parties to them. In general, assigning responsible parties to goals, requirements, and obstacles was useful for explicating who dealt with these boundary concerns because these might have otherwise been ignored. In particular, in the final case study in Chapter 9, doing so ensured that concerns which were passed-off during focus group discussions could be addressed by the appropriate parties in the subsequent review session.

Given that attackers may be agnostic of organisational boundaries, IRIS (and CAIRIS in particular) was useful for tracking who was responsible for what, identifying what goals or obstacles were unassigned, and also what vulnerabilities might be related to assigned or unassigned goals or obstacles.

10.1.3 Security through usability

As Chapter 9 alludes to, although risk analysis has been a dominant technique in several security design methods, their applicability in IRIS tended to be for dealing with analysis where stakeholder participation was necessary. For example, the risk analysis models were most useful in Chapter 7 for highlighting risks which were scored as unexpectedly low due to oversight in the setting of security properties for exploited assets. Across all studies, few risks were elicited because many of the vulnerabilities which might have given rise to them were addressed during requirements analysis.

As such, rather than being used as an elicitation technique like their Usability counterparts, Security techniques appear to be most valuable for validating and explicating design decisions. For example, misuse cases were useful for validating the analysis underpinning associated risks, and the risk analysis model was useful for obtaining a high-level overview of how security and usability were impacting design at different points. In particular, the Misusability Case technique introduced and applied in Chapter 8 showed how innovative requirements for usable and secure controls could follow by exploring the design implications of system vulnerabilities being exploited.

Section 5.2 alludes to Security and Usability perspectives treating design as a hermeneutic circle rather than simply a linear process. Throughout the case studies, it was seen that usability insights can inform security, and vice-versa. In Chapter 7, personas were found to be useful for grounding discussion in participatory sessions, but the data used to generate them helped identify possible vulnerabilities. Chapter 9 went a step further in that the empirical data used to develop personas was more heavily used than the personas themselves.

Although security and usability continually informed each other throughout the case studies, the concepts associated with the IRIS Requirements perspective tended to be treated as objective boundary objects. When new insights arose from examining these, these were usually attributed to the Security or Usability concepts that related to them. This phenomenon reinforces the importance of Requirements Engineering concepts remaining objective, together with the techniques for specifying and refining them.

10.1.4 IRIS framework review

Although the IRIS Meta-Model and process framework in Chapters 4 and 5 were initially validated using a specification exemplar, changes were inevitable when applying these to real-world case studies. The following section consolidate the IRIS meta-model and process framework modifications arising from these studies.

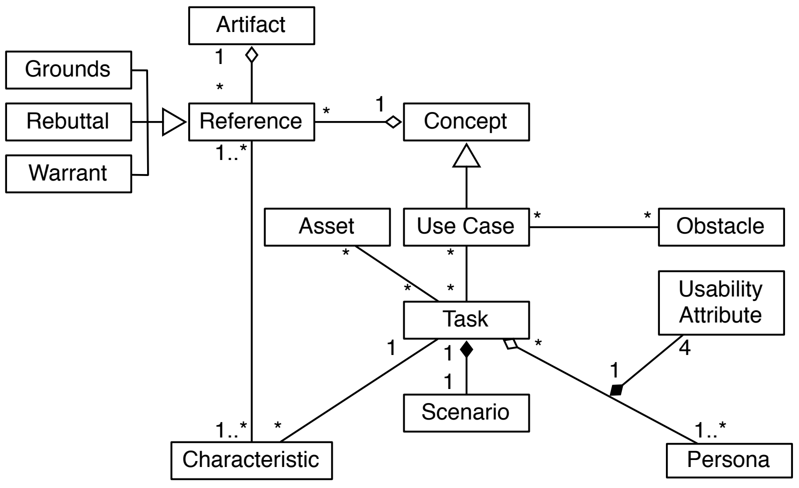


Figure 10.1: Updated Task Meta-Model for Misusability Cases

10.1.4.1 IRIS meta-models

The IRIS Meta-Model remained largely unchanged throughout the case studies. However, the following modifications were necessary to support the new design techniques and key findings:

- The concepts described in Figures 8.2 and 8.5 were incorporated into the Task Meta-Model to support the Assumption Persona Argumentation and Misusability Case techniques. This revised model is illustrated in Figure 10.1.
- As recommended by Section 10.1.2, the Responsibility Meta-Model was updated to incorporated associations between vulnerabilities and responsible roles. This revised model is illustrated in Figure 10.2.

10.1.4.2 Process framework

Figure 10.3 (left) summarises the current relationship between IRIS concepts and its three perspectives. With the exception of the Attacker concept, the existing IRIS concepts remain in their original perspectives. Based on the findings in Section 10.1.2 and the meta-model revision in Figure 10.2, attackers were moved to the intersection of the Security and Requirements perspectives. This change is necessary if attacker–responsibility relations are to inform objective system design. This figure also shows that the references, characteristics, and artifacts are situated in the intersection between Security and Usability. This is because argumentation models support security and usability design artifacts, but these are not directly used to specify systems in IRIS. The introduction of misusability cases has also implicitly introduced use cases to the IRIS meta-model. Use cases are situated in the Requirements perspective because IRIS treats these as objective system scenarios.

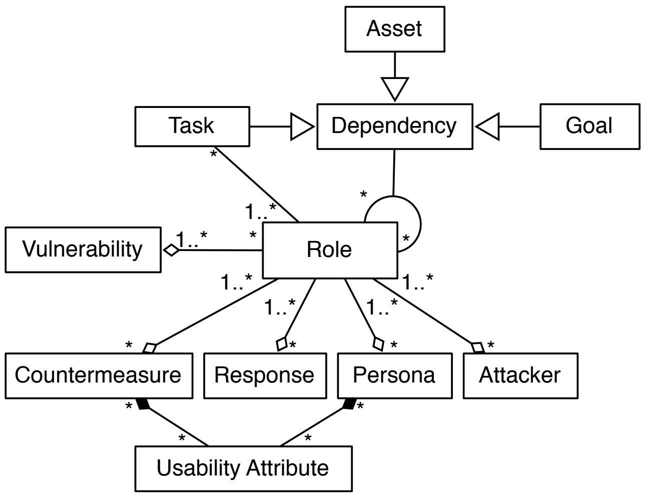


Figure 10.2: Updated Responsibility Meta-model

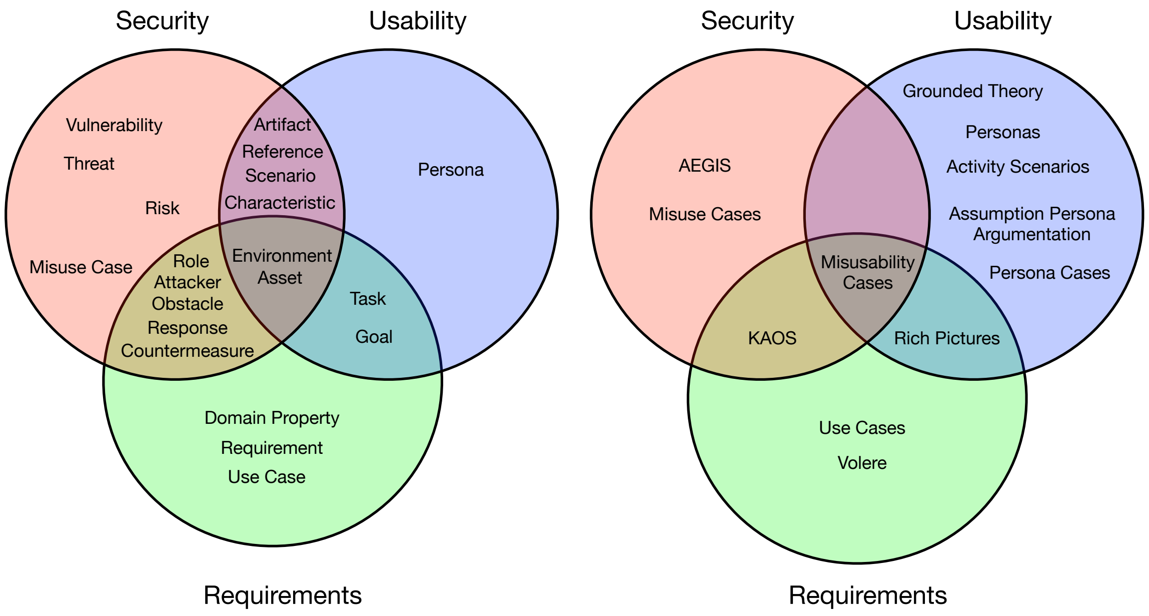
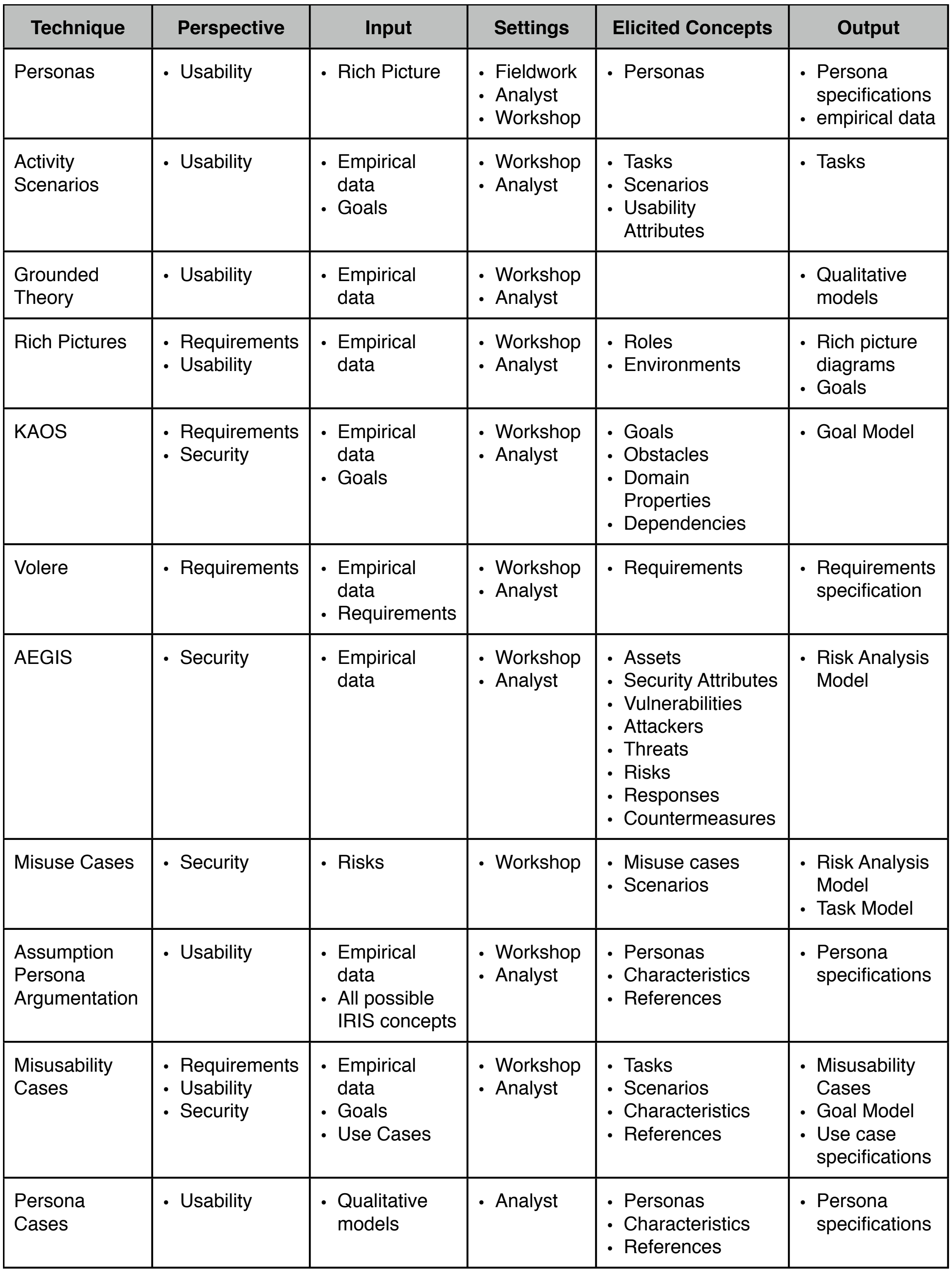


Figure 10.3: Revised IRIS perspective concepts (left) and techniques (right)



Technique	Perspective	Input	Settings	Elicited Concepts	Output
Personas	• Usability	• Rich Picture	• Fieldwork • Analyst • Workshop	• Personas	• Persona specifications • empirical data
Activity Scenarios	• Usability	• Empirical data • Goals	• Workshop • Analyst	• Tasks • Scenarios • Usability Attributes	• Tasks
Grounded Theory	• Usability	• Empirical data	• Workshop • Analyst		• Qualitative models
Rich Pictures	• Requirements • Usability	• Empirical data	• Workshop • Analyst	• Roles • Environments	• Rich picture diagrams • Goals
KAOS	• Requirements • Security	• Empirical data • Goals	• Workshop • Analyst	• Goals • Obstacles • Domain Properties • Dependencies	• Goal Model
Volere	• Requirements	• Empirical data • Requirements	• Workshop • Analyst	• Requirements	• Requirements specification
AEGIS	• Security	• Empirical data	• Workshop • Analyst	• Assets • Security Attributes • Vulnerabilities • Attackers • Threats • Risks • Responses • Countermeasures	• Risk Analysis Model
Misuse Cases	• Security	• Risks	• Workshop	• Misuse cases • Scenarios	• Risk Analysis Model • Task Model
Assumption Persona Argumentation	• Usability	• Empirical data • All possible IRIS concepts	• Workshop • Analyst	• Personas • Characteristics • References	• Persona specifications
Misusability Cases	• Requirements • Usability • Security	• Empirical data • Goals • Use Cases	• Workshop • Analyst	• Tasks • Scenarios • Characteristics • References	• Misusability Cases • Goal Model • Use case specifications
Persona Cases	• Usability	• Qualitative models	• Analyst	• Personas • Characteristics • References	• Persona specifications

Figure 10.4: Revised overview of IRIS techniques

As Figure 10.3 (right) shows, the existing IRIS techniques remain in their original perspectives. The Assumption Persona Argumentation and Persona Case techniques are situated in the Usability perspective as these are primarily used to support the creation of personas. The Use Case technique, like its associated concept, is situated in the Requirements perspective; this is because use cases are used as a specification technique rather than an context exploration technique like activity scenarios, or a risk validating technique like misuse cases. Misusability cases are, unlike other techniques, grounded in all three perspectives. Although the technique is, in theory, only a nexus between the Usability and Requirements perspectives, their underlying artifacts and references are informed by investigations into how misusability can lead to security exploitation.

The contribution of the new techniques to the IRIS framework is summarised in the revised table of IRIS techniques in Figure 10.4.

10.2 Evaluation

In Chapter 1 we claimed that **the IRIS (Integrating Requirements and Information Security) framework is an exemplar for integrating existing techniques and tools towards the design of usable and secure systems**.

Figure 10.5 provides a diagrammatic model of the thesis' main claim, and the propositions upon which this based. For the thesis to hold, we claim that each of these concepts or statements are valid. The arrows between boxes indicate that the tail concept acts as grounds for the head. For example, the validation of the thesis is based on the validity of cases 1, 2, and 3. Lessons learned from cases 1 and 2 contribute to the propositions upon which the design of cases 2 and 3 are based. The validity of all three cases is contingent on the soundness of the case study method used to design the cases, the process framework used to design the processes used in each study, and the CAIRIS tool used to support the techniques used in each process.

The figure illustrates how the motivation for integrating techniques and tools towards the design of usable and secure systems also motivated the design of the literature review. Given the broadness of HCI, Information Security, and Requirements Engineering, the notion of *design*, and how each field dealt with it, was used to drive this review. Based on this review, limitations were identified; these pointed to the lack of framework support for the technique and tool integration we require. Using the literature review as a scope of analysis, we explored possible methodological approaches for conducting research and validating the research contributions. This review was used to devise a methodology for developing and theoretically validating IRIS, together with a separate methodology for performing a more robust real-world validation of the framework. Before commencing any further research, we presented a specification exemplar for justifying and explaining the various parts of the IRIS framework.

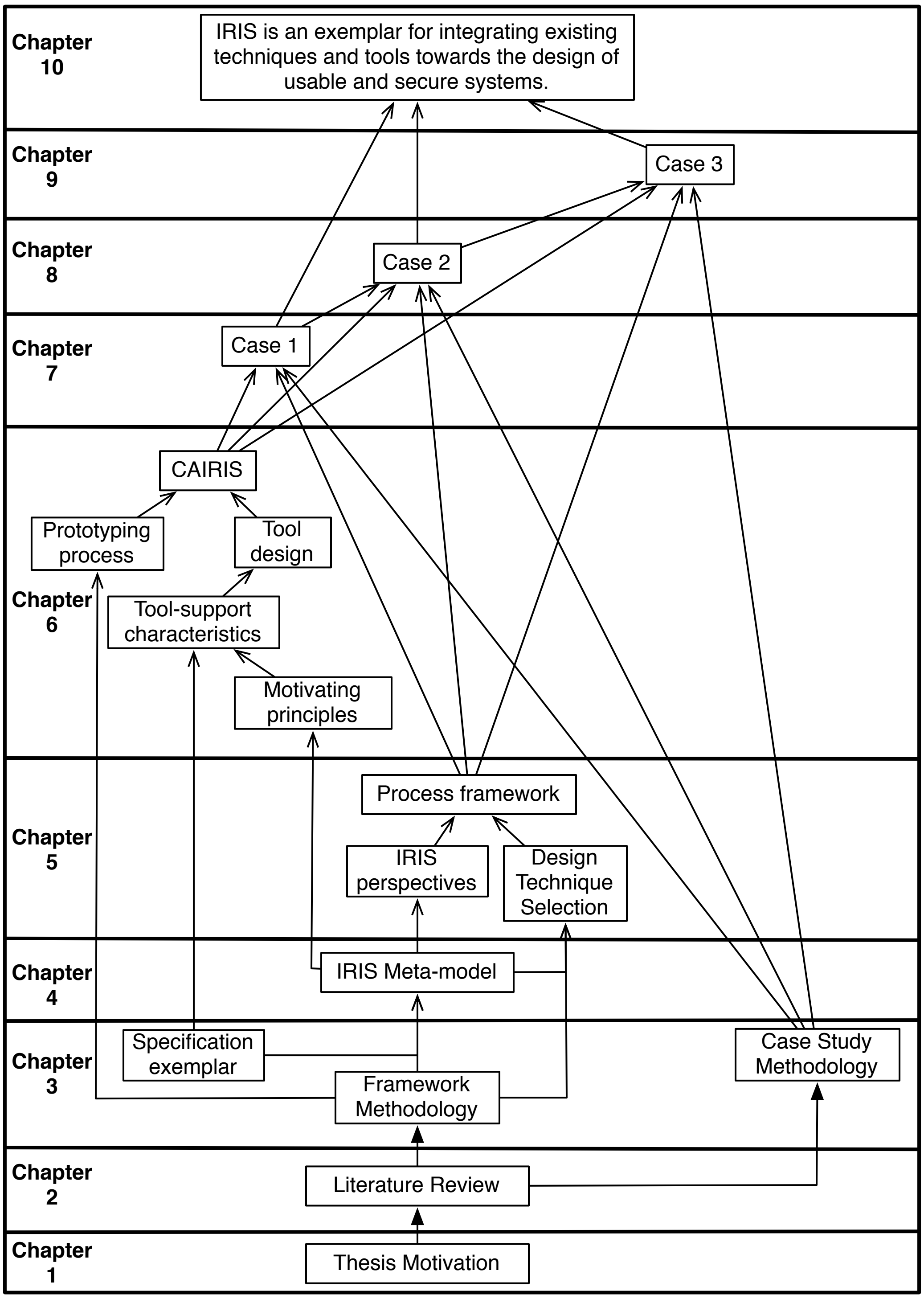


Figure 10.5: Thesis and supporting propositions

Guided by the framework methodology, we used the relevant theory from HCI, Information Security, and Requirements Engineering to develop the IRIS meta-model; this model was illustrated using representative examples for the specification exemplar. The IRIS meta-model provided the foundation for the remaining components of the IRIS framework in two respects. First, the meta-model suggested the presence of Usability, Security, and Requirements perspectives which characterised different approaches to the activity of design. Guided by both the framework methodology and these perspectives, several design techniques were selected which, when applied, provided elicitation coverage of the IRIS concepts. Second, the meta-model, and the theory it builds upon, suggested design principles that tool-support for IRIS should entail. By considering these principles using the specification exemplar, specific characteristics for tool-support could be derived. These characteristics informed the architecture and design of the CAIRIS prototype presented in Chapter 6.

The IRIS Framework was validated collectively using the case studies described in Chapters 7, 8, and 9. Each study was designed using the methodology presented in Section 3.2.2.3 and, in each case, the objective of the intervention was to devise and apply an IRIS design process for eliciting system requirements. The stages in both the planned and executed design processes were compared and contrasted, before each process was critically analysed and, based on this analysis, lessons learned were drawn. The lessons learned from the studies in Chapters 7 and 8 were subsequently used to inform the intervention plan described in Chapters 8 and 9 respectively.

In the following sections, we critically analyse the validity of the thesis in more detail, together with the component research questions described in Section 1.2 upon which this is based.

10.2.1 The IRIS framework is an exemplar for integrating existing techniques and tools towards the design of usable and secure systems

The success of the case studies validates that the IRIS framework is an exemplar for integrating existing tools and techniques for specifying systems which are both secure and situated in their contexts of use. For each collection of case specific factors, an IRIS process satisfying the criteria set in Section 5.4 was defined, and the process was adequately supported by software tools. Although new techniques were proposed in Chapters 8 and 9, these were grounded in existing design techniques rather than being *engineered* specifically for IRIS. As such, each of the proposed techniques stands alone; a testament of this are the peer-reviewed publications presenting each of these techniques independently of the IRIS framework [89, 91, 92].

One criticism which could be levelled at arguing for IRIS' validity based on the results of these case studies is that these processes could have been devised by serendipity or as a natural consequence of the general diagnosis phase carried out at the start of each intervention. There are, however, three counter-arguments for why this is not the case.

First, the number of employable techniques from HCI, Requirements Engineering, and Information Security is significant, and, as Section 1.1 highlighted, their order of application is also variable. An informed analyst may instead choose to apply an existing design framework such as those described in Section 2.4, but the security or usability of the resulting specification may fall foul of the issues also identified, e.g. lack of traceability between elicited concepts in RESCUE, and an assumption that security is a primary system concern in SQUARE.

Second, the switching between perspectives in each of the devised IRIS processes was transparent to the participating stakeholders, be they focus group attendees, developers in design sessions, or reviewers validating the final analysis by walking through various IRIS models and requirements. This suggests that IRIS processes have a conceptual integrity that, despite them being a composite of multiple techniques, allows both the process and its artifacts to be easily digested by non-experts.

Third, the elicitation and analysis activities were carried out in close proximity with CAIRIS. To date, we are unaware of existing work purporting to collectively support all of the techniques used in a single tool. In the case of complementary tools used for qualitative data analysis, we are unaware of any prior work on reconciling (Chapter 7) or integrating (Chapter 9) the output from qualitative data analysis tools with Requirements Management tools.

One weakness that does undermine some of IRIS' soundness is its "toolbox-like" nature. In particular, it is left to the analyst to perform a diagnostic analysis of the context of intervention and, on the basis of these, orchestrate the various IRIS process stages that conform to the criteria stated in Section 5.4. Because IRIS is a toolbox rather than a unified methodology, IRIS is contingent on the capabilities of the designer to choose the correct combination of techniques. Although guidance is provided about how precisely one technique works with another, the sense-making activities carried out to devise the different clusters of techniques could have been better supported.

10.2.2 Which concepts from Usability, Security, and Software Engineering can be harmonised to support the design of secure and usable systems?

Based on the issues reported in Chapter 2 and Chapter 4, concepts from User-Centered Design, Information Security, and Requirements Engineering were proposed which, collectively, were necessary for specifying the elements of a secure system situated in its designed contexts of use. In Chapter 5, we described how viewing the design process from Security, Usability, and Requirements perspectives informed the selection of techniques for eliciting these concepts. We demonstrated both the completeness of these concepts, and a harmonised approach for eliciting them in the case study chapters.

Although the meta-model was in a state of flux during the early stages of the development of CAIRIS, only minor modifications to the meta-model concepts and associations were made during the first case study. Despite several CAIRIS changes in subsequent case studies, no further meta-model elements or associations were changed. The only significant changes made were to support the new techniques introduced in Chapters 8 and 9.

While the concepts elicited were both necessary and sufficient for applying IRIS in the validating case studies, there are two particular weaknesses in the meta-model.

The first relates to the notion of user goals and weak support for agent-oriented goal modelling. Such approaches are an active research topic in Requirements Engineering; exploring how IRIS concepts might interrelate with these approaches may lead to new insights into how usability artifacts can be used. In particular, determining techniques for eliciting data about persona rationale and goal dependencies may lead to ways of contextualising personas without relying solely on tasks.

The second weakness relates to the concepts supported for risk analysis. Given the substantial work carried out by Mayer towards aligning Requirements Engineering with Information Security [165], it would have been inappropriate to naively reproduce this work. It was, therefore, decided to simplify the risk analysis approach adopted by IRIS to indicate how a representative approach to risk analysis might align with requirements and usability concepts. In doing so, opportunities were lost for modelling sophisticated risk analysis strategies, for example, attacks leading to context changes as indicated in the evaluation section of the first case study, or blended threats where an attacker might exploit two seemingly innocuous vulnerabilities, leading to potentially catastrophic results.

10.2.3 What are the characteristics of tool-support needed to support the design of secure and usable systems?

Based on the weaknesses found in the literature, design principles were inferred and, from these, four characteristics for software tools needed to design secure and usable systems were identified: Automated Analysis, Model Visualisation, Model Reusability, and Model Externalisation. These characteristics were validated in two ways. First, the CAIRIS software tool was developed to embody these characteristics, and the NeuroGrid exemplar was used to illustrate how these characteristics appealed to the design principles originally identified. Second, CAIRIS was used to support the IRIS processes in the three validating case studies. During the evaluation phase of each intervention, only minor, incremental improvements were necessary.

In all three case studies, the author was the sole user of CAIRIS. However, applying the tool in a multi-user environment introduces a number of additional requirements for the Model Externalisation characteristic. These include:

- Additional information about contributors needs to be added to several concepts, such as document references and goals.

- An alternative externalisation format is needed for goals and requirements if a team decides to treat these synonymously. This may include adopting an alternative requirement labelling scheme based on a goal's position in the hierarchy, or publishing requirements to an alternative output format, such as a wiki-compatible ASCII text format.
- As well as supporting security model re-use, fine grained externalisation of IRIS model concepts is needed; this is necessary for merging contributions from different CAIRIS users into a single CAIRIS model. Satisfying this requirement entails additional functionality for the exporting and importing XML based model data.

The identification of these tool features does raise one criticism of the identified characteristics. These characteristics inform how tools should be *designed* to support secure and usable system design, but do not inform designers about using the tools to *design* secure and usable systems. While the Action Research evaluation provided some measure of how effective CAIRIS was towards this latter question, these results are less convincing than an independent expert analysis based on a set of design heuristics.

It is also unclear what design heuristics might be most suitable for such an analysis, and how useful existing design heuristics might be for this purpose. For example, many of the heuristics proposed by Olsen's criteria for evaluating User Interface systems work [190] might be applicable for evaluating CAIRIS, e.g. *Expressive Leverage* such that designers can accomplish more tasks with less, and *Simplifying Interconnection*. However, heuristics like *Inductive Combination*, which is concerned with how design primitives combine to form more complex designs, is more suited to a user interface design tool than a Requirements Management tool like CAIRIS.

10.2.4 How can user-centered design techniques be improved to support the design of usable and secure systems?

This dissertation has made three contributions towards improved User-Centered Design approaches for designing secure systems. First, we provided guidance towards how personas and activity scenarios could be used in secure system design. Although there have been sporadic reports of the usefulness of both approaches for designing security, the first case study provides concrete guidance for how these can be used to help design secure systems. Second, based on the need to better identify assumptions in Chapter 8, we described how the elicitation and validation of assumption personas could be improved using the Toulmin Model of argumentation to structure persona characteristics. We also demonstrated how activity scenarios could be better used to explore the systemic impact to security of poor usability design by building on the argumentation model. Third, we build upon these structured persona characteristics by describing how they could be used to model the sense-making activities underpinning Grounded Theory analysis. Using this, we demonstrated how persona characteristics could be semi-systematically generated directly from Grounded Theory models.

While the dissertation made contributions towards the improved User-Centered Design approaches, these were primarily based on the individualistic persona technique. In Chapter 9, we saw an example of how multiple personas collaborated in an activity where the goals of both personas were being satisfied. Even though the policy's focus in this study was based on the impact to a single persona, the example arose because the study stakeholders wanted to examine the impact of policy changes to a collaborative activity. Given an renewed interest in CSCW design approaches in the HCI literature, understanding how IRIS could have contributed to improved collaborative design techniques might have led to valuable insights into the IRIS meta-model, techniques for eliciting such data, and tool-support.

The techniques used in the dissertation case studies have been actively used in other real-world case studies. However, CSCW techniques remain largely untested in software engineering contexts. Moreover, as Section 2.2.2.4 suggests, certain tensions may be inevitable when trying to apply CSCW techniques in practice. For example, the lack of end-user access in the case study of Chapter 8 and the severe time constraints in the case study of Chapter 9 both reduced the opportunities for collecting observational data and engaging in participatory design activities. Considering these tensions in the context of the specific interventions suggests that further work is needed on adapting work from CSCW into general design contexts before attempting to scale these up to addressing secure systems developed specifically by integrating pre-existing design techniques. Failure to do so may, as Bannon recently noted, lead to problems due to a misalignment of motives and interests between developers and sociologists [25].

One possible approach for integrating CSCW techniques into the IRIS framework might involve determining how the activities carried out in using them might be complementary with the activities carried out by techniques within the same (Usability) perspective, or other Requirements or Security techniques. The Persona Case technique is an example of such an approach.

10.2.5 Research question evaluation

The evaluation of this thesis confirms that the original research question posed in Section 1.2, **How can existing techniques and tools be integrated and improved to support the design of usable and secure systems?**, has been answered by illustrating how the IRIS framework demonstrates the integration of existing techniques and tools in Chapters 7, 8, and 9.

The research contributions arising from this dissertation have also answered the sub-questions associated with the main question.

RQ1 was answered by presenting the meta-model in Chapter 4, which described the relationships between design concepts from Usability, Security, and Software Engineering, and the process framework in Chapter 5, which described how techniques can be harmoniously combined to elicit them.

RQ2 was answered by stating the tool characteristics needed to support the design of secure and usable systems in Chapter 6; these characteristics were illustrated by developing a software tool, CAIRIS, embodying these.

RQ3 was answered by describing the technique changes in Chapter 5, and presenting improved techniques for applying personas (Assumption Persona Argumentation and Persona Cases) and scenarios (Misusability Cases) in Chapters 8 and 9.

10.3 Future work

In this section, we propose three possible areas for future research that build upon the contributions arising from this dissertation. These areas are based on the contributions IRIS can make towards modelling an organisation's security assurance level, improving the attacker models used in security design techniques, and aligning IRIS artifacts towards known design patterns.

10.3.1 Modelling security assurance

Many organisations, especially Critical Infrastructure providers like ACME, are under increasing pressure to develop and adopt organisational assurance strategies. This has led to the development of maturity models such as the OWASP-sponsored Software Assurance Maturity Model (OpenSAMM) [45]. These models are frameworks which can be used to select security practices for different business functions. In particular, Brownsword et al. [39] recently demonstrated a framework for evaluating the assurance of organisational systems using strategic planning techniques such as system dynamics modelling [234] and long-term scenario planning [215].

Given the scope of this dissertation, it is reasonable to ask how the lessons learned developing and applying IRIS might be useful for developing assurance, rather than requirements, models. These frameworks, like the specification frameworks described in Section 2.4, are agnostic to how data contributing to the various models is specified, and also appear to lack an agreed, objective perspective that models and techniques from other perspectives can inter-relate to. For example, Brownsword's framework uses different techniques to develop different views of assurance data, but it is unclear how different views in the modelling framework inform each other. To determine how IRIS might be used to develop assurance models, the concepts associated with [39] need to be aligned with the IRIS meta-model. Based on this, candidate techniques can be considered for eliciting the concepts necessary to develop an objective, assurance model.

One interesting aspect of Brownsword's framework was the inclusion of innovation techniques to examine the assurance maturity of a system. The Misusability Case technique introduced in Section 8.3.2 also demonstrated the opportunities IRIS affords for stimulating innovation in design. Recent work has examined how creativity and innovation techniques can be usefully employed for designing secure systems [90]. Like misusability cases, these techniques rely on the grounding of empirical data, and the application of HCI tools such as Defamiliarisation [30]. This work raises the question of what further role IRIS can play in fostering innovation in design. For example, given the potentially disruptive nature of innovation, might a further perspective be necessary to consider how innovation models and techniques integrate with other design techniques?

10.3.2 Attacker models

As the review in Section 2.3 indicates, there has been much research on how Requirements Engineering techniques can be used to improve the design of secure systems. Notwithstanding this dissertation's research contributions, there has been less work on how Usability Engineering techniques can similarly contribute to design techniques situated within the IRIS Security perspective.

One possible contribution that IRIS can make is in explicating assumptions about existing security design artifacts. In particular, the adversarial nature of secure systems design demands that design activities need to factor in the role of the attacker. However, doing so is problematic because we do not know, with the certainty that we would like, precisely who the attackers are.

The idea of adopting an adversary-centered, as opposed to user-centered, approach to security design has been proposed [233], which involves creating personas for attackers. However, simply re-casting User-Centered Design for attackers rather than users is problematic for two reasons. First, even if we could definitively state who the attackers are, we usually don't have convincing data about how attackers have exploited, or may exploit, the systems we wish to build. Second, in lieu of convincing data, we may fall foul of our own stereotypes about who a system's attackers are, and the capabilities they have. For example, attempting to mitigate against attacks precipitated by a *super-hacker* may lead to design decisions which are over-commensurate to the risks the system faces in reality.

IRIS can make two contributions to this aforementioned problem. The first of these involves re-using the Assumption Persona Argumentation technique introduced in Chapter 8 to ground the assumptions being made about attackers. Exposing these may lead to the identification of possible fallacies as well as additional assumptions; eventually, this may lead to an attacker persona which is both better grounded and more believable. An approach for developing attacker personas based on these ideas is currently being developed as on-going work in the EU FP7 webinos project [259]. The results of this approach will be the subject of a future publication.

The second possible contribution towards developing attacker models builds on the success of the Persona Case technique introduced in Chapter 9, which suggests that other qualitative models can be independently developed and re-purposed for the design of secure systems. In particular, as Section 9.6.1 suggests, this technique may scale to more general qualitative models.

Previous work has reported how criminal profiling techniques can be used identify different categories of possible internet-facing attackers [50]. More recent work has modelled the different motivations for a similar taxonomy of attackers and, using Criminology models such as Social Learning Theory, attempted to justify the motivations for attackers within these categories [202]. Using the Persona Case technique, it should be possible to go one stage further and, using qualitative data, develop grounded personas to support security design techniques. In the short term, such qualitative models could be developed via a critical review of the related cybercrime literature using Grounded Theory as a design technique, as described in Section 5.4.1. If, however, the intent is to develop a catalogue of generic attacker persona cases, then careful thought needs to be given for sensitising questions to guide subsequent analysis. In the medium to long term, pre-existing qualitative models or sources of empirical data about attackers need to be found, or research approaches devised for collecting meaningful data to support qualitative data analysis.

10.3.3 Design pattern alignment

As highlighted in Chapter 1, *design* in this dissertation was restricted to the elicitation and specification of system requirements, and the activities necessary to support them. Although architectural components could be modelled as IRIS assets or countermeasures from a security analysis perspective, further techniques would need to be applied to consider other architectural concerns.

Section 3.1.3 highlighted the research challenges associated with fitting known solutions to problems at hand. Although there has been little work in this area at the level of this dissertation's abstractions, there are several examples of how design techniques from Requirements Engineering, HCI-Sec, and Information Security can foster re-use from an architectural viewpoint. These include work on specification patterns for security goals [254], security design patterns [214], and the design patterns and guidelines for usable secure controls reviewed in Section 2.1.2.1.

Future work should consider how CAIRIS can be developed to support the application of design patterns. One possible direction might involve aligning these to IRIS countermeasures such that introducing these within CAIRIS leads to the pattern elements being added to the same environment as the countermeasures. Based on the security pattern structure defined by [214], this would entail the introduction of the assets and asset associations realising the structural pattern elements, and additional requirements corresponding to the design criteria needing to be satisfied for the pattern to be usefully applied.

Future work also needs to consider the additional work necessary to contextualise patterns introduced into an environment. For example, introducing the Packet Filter Firewall pattern described in [214] to an environment also introduces assets corresponding to the firewall software, an abstract asset for an external host server, and component for storing fire-wall rules. Indications for what the security properties associated with these assets can be considered on an a-priori basis, but cannot be confirmed until they have been introduced. Similarly, a-priori requirements associated with patterns need to be re-evaluated once introduced into a model to ensure they do not duplicate or conflict with existing goals and requirements.

In the longer term, the information stored within the CAIRIS database may lead to useful insights from an Empirical Software Engineering perspective. For example, analysing how the artifacts elicited from security, requirements, and usability analysis map onto pre-existing design patterns, and what types of vulnerabilities might be exposed by these patterns may provide data about pattern improvements for mitigating these vulnerabilities.

Appendix A

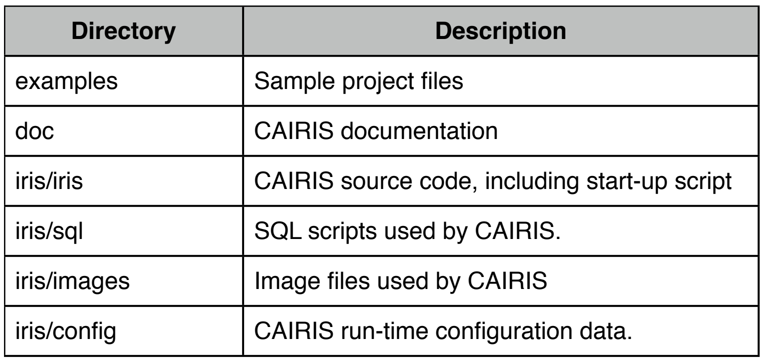
CAIRIS

This appendix contains some practical details necessary for installing and starting up CAIRIS. For more detailed instructions, readers should consult the CAIRIS User Manual [85].

A.1 Pre-requisites

Before CAIRIS can be used, recent versions of the following open-source applications need to be installed:

- Python
- MySQL Community Server and Client
- MySQLdb
- GraphViz
- pyparsing
- PyDot
- NumPy
- wxPython
- Glade
- DocBook
- dblatex



Directory	Description
examples	Sample project files
doc	CAIRIS documentation
iris/iris	CAIRIS source code, including start-up script
iris/sql	SQL scripts used by CAIRIS.
iris/images	Image files used by CAIRIS
iris/config	CAIRIS run-time configuration data.

Figure A.1: CAIRIS installation directories

In theory, CAIRIS will run on any platform which supports Python and the above pre-requisites. In practice, however, running all of these pre-requisites on any operating system other than Linux might be challenging. Therefore, it is recommended that CAIRIS is used only on Linux. CAIRIS has been tested using recent versions of Red Hat, SuSE, and Ubuntu. In particular, all of the pre-requisites are available via Ubuntu's Synaptic Package Manager.

A.2 Obtaining CAIRIS

A source distribution of CAIRIS can be downloaded by clicking on the Download link at `http://www.cs.ox.ac.uk/cairis`.

Following checkout, the directories described in Figure A.1 are created beneath the root iris directory.

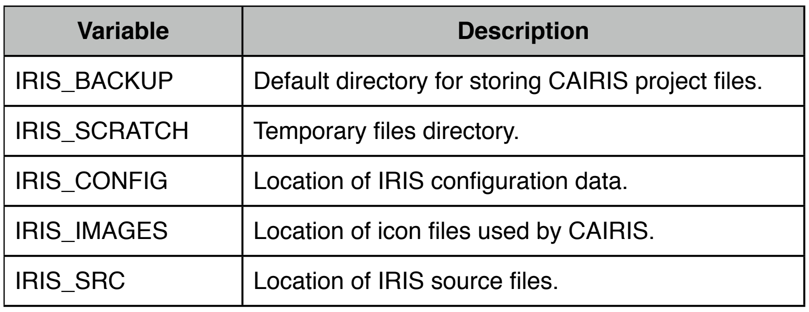
A.3 Installation

A number of environment variables are used by CAIRIS at run-time; these are defined in Figure A.2. These must to be set before CAIRIS can be started.

The IRIS_CONFIG, IRIS_IMAGES, and IRIS_SRC environment variables should be set with the current locations of iris/config, iris/images, and iris/iris respectively. It is recommended that /tmp is set for the location of IRIS_SCRATCH.

CAIRIS stores its data in a MySQL database. An empty database needs to be created, and an account associated with it, before CAIRIS can be started. When this database is created, a file called *db.ini* must be created. This file must contain the following 5 lines:

- The database server network location, e.g. 127.0.0.1



Variable	Description
IRIS_BACKUP	Default directory for storing CAIRIS project files.
IRIS_SCRATCH	Temporary files directory.
IRIS_CONFIG	Location of IRIS configuration data.
IRIS_IMAGES	Location of icon files used by CAIRIS.
IRIS_SRC	Location of IRIS source files.

Figure A.2: CAIRIS environment variables

- The database server port, e.g. 3306
- The database user, e.g. irisuser
- The database password
- The database name, e.g. iris

This file needs to be saved in the iris/iris directory. A sample db.ini file is included in the iris/iris directory of the CAIRIS source distribution.

Using mysql command line tool, it is also necessary to run two SQL scripts in the iris/sql directory. The init.sql script creates the CAIRIS database table, and populates the database with meta-data needed by the tool. The procs.sql script creates a number of stored procedures and database functions used by CAIRIS. These scripts can be run from the mysql prompt.

Finally, because a number of the CAIRIS stored procedures are recursive, it is necessary to modify the database server to support recursion. This support can be added by starting the mysql command line tool using the root account and setting the global max_sp_recursion_depth variable to 255. Once the variable has been set, the tables and privileges need to be flushed.

A.4 Running CAIRIS

To start CAIRIS, run the iris.py script in the iris/iris directory.

To save the contents of this current CAIRIS database instance, click on the Save project toolbar button and, from the save project data dialog box, enter the name of the project file and the location to store the data. The data is stored as an ASCII text SQL dump file.

To open a pre-existing project file, click on the Open project toolbar button and select the dump file to open.

A.5 Sample project

A model file for the NeuroGrid specification exemplar, completeExample.sql, can be found in the examples directory.

A.6 Model notation

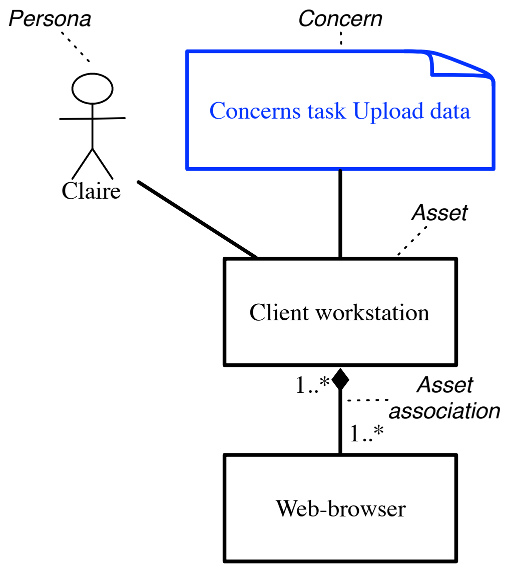


Figure A.3: Asset model notation

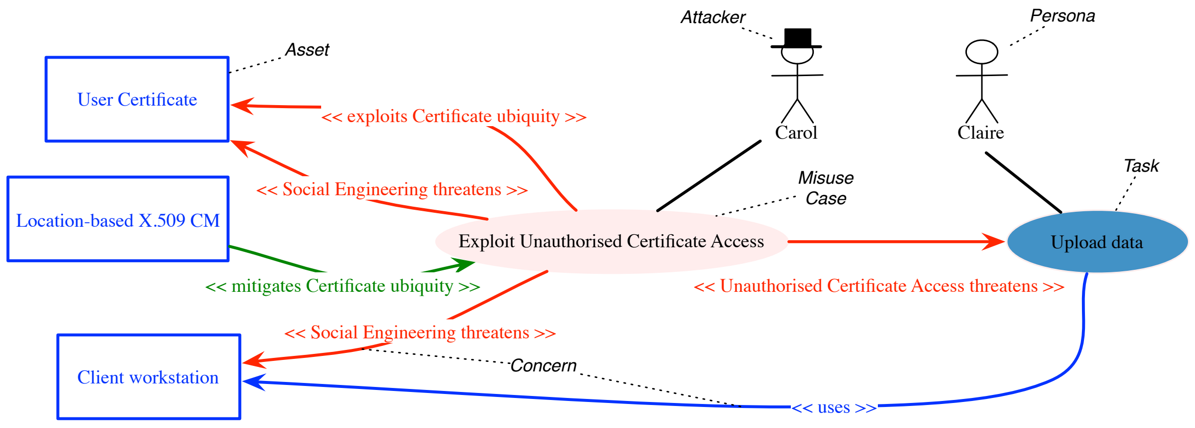

Attacker
Persona
Asset
User Certificate
<< exploits Certificate ubiquity >>
Carol
Claire
Task
<< Social Engineering threatens >>
Misuse Case
Location-based X.509 CM
Exploit Unauthorised Certificate Access
Upload data
<< mitigates Certificate ubiquity >>
<< Unauthorised Certificate Access threatens >>
<< Social Engineering threatens >>
Client workstation
Concern
<< uses >>

Figure A.4: Task model notation

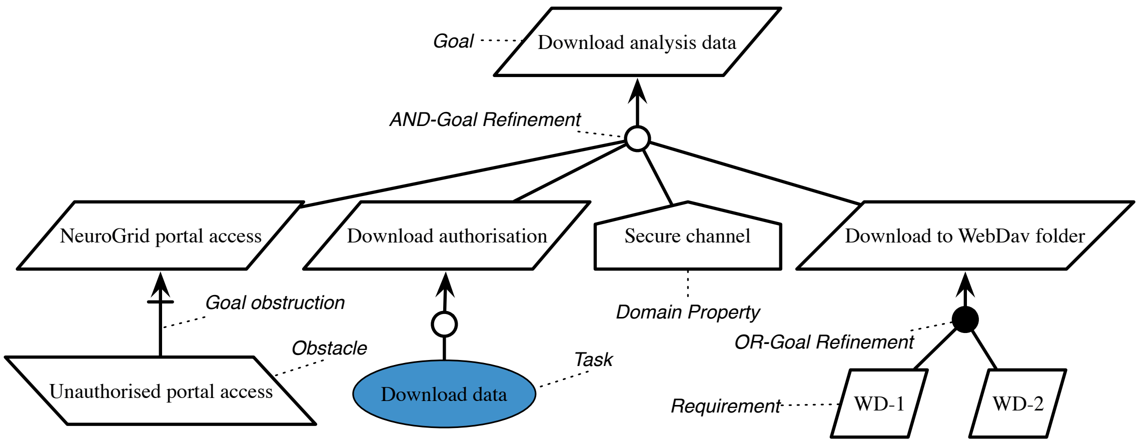


Figure A.5: Goal model notation

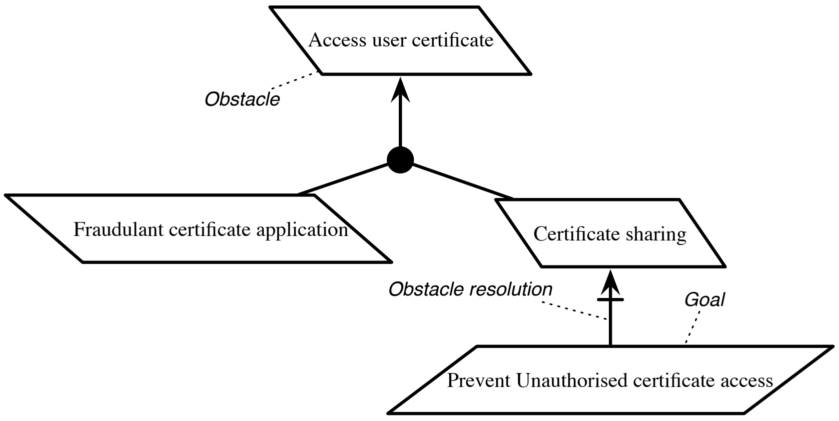

Access user certificate
Obstacle
Fraudulant certificate application
Certificate sharing
Obstacle resolution
Goal
Prevent Unauthorised certificate access

Figure A.6: Obstacle model notation

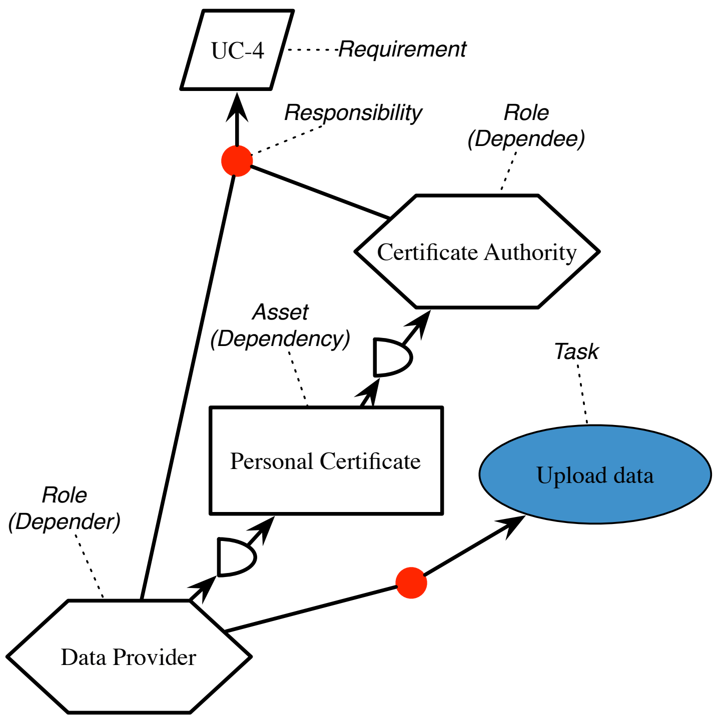


Figure A.7: Responsibility model notation

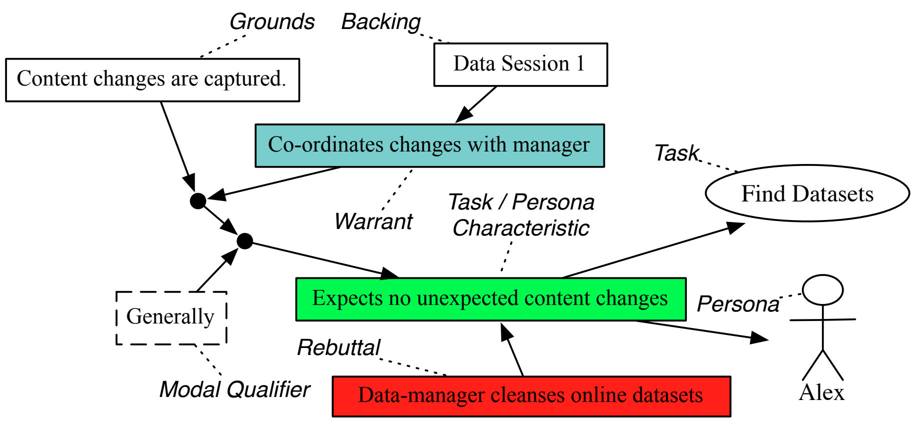

Grounds
Backing
Content changes are captured.
Data Session 1
Co-ordinates changes with manager
Task
Find Datasets
Warrant
Task / Persona
Characteristic
Generally
Expects no unexpected content changes
Persona
Rebuttal
Modal Qualifier
Data-manager cleanses online datasets
Alex

Figure A.8: Assumption persona and task characteristic model notation

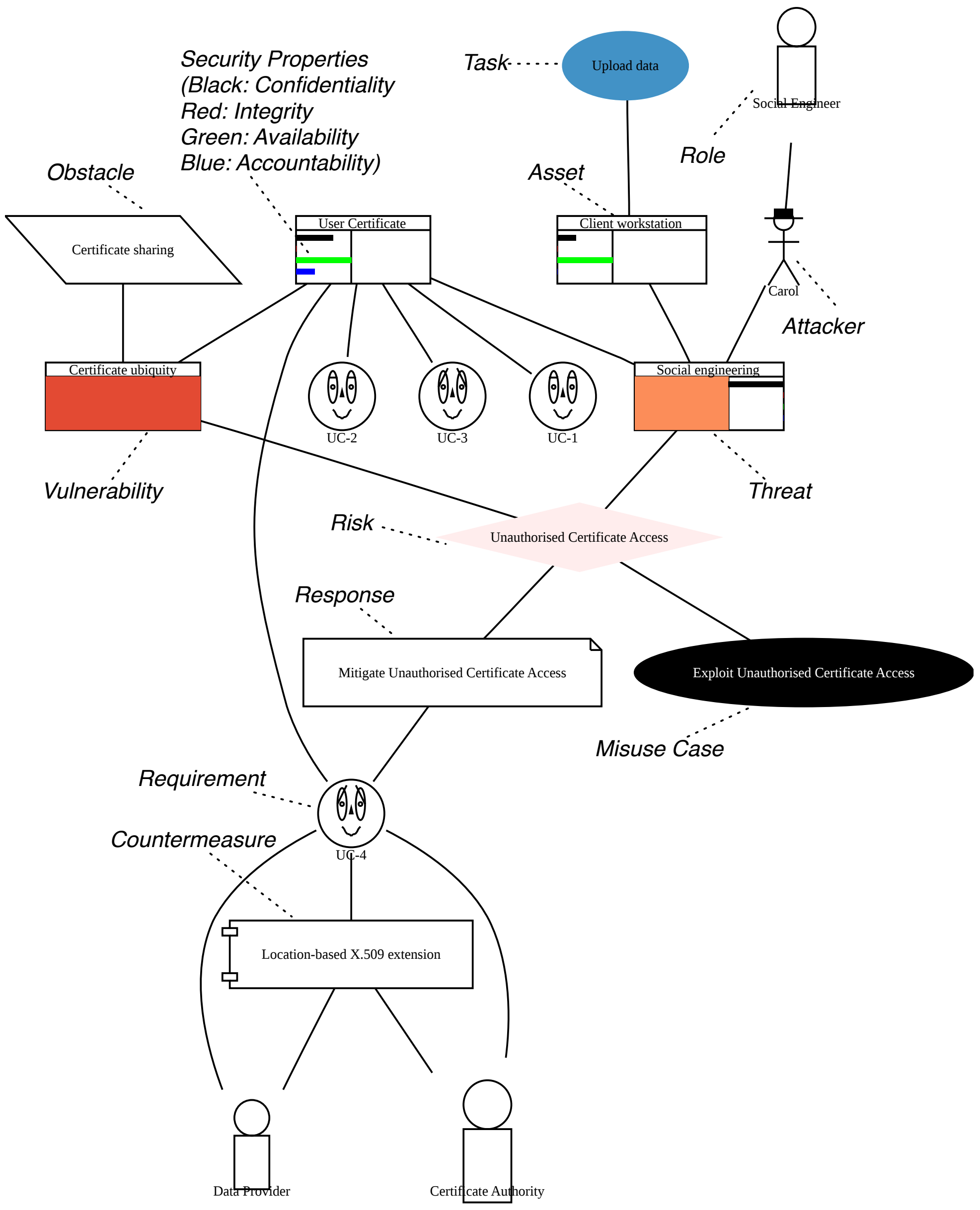


Figure A.9: Risk analysis model notation

Bibliography

[1] *International Standard IEC 1025 Fault Tree Analysis (FTA)*. International Electrotechnical Commission, 1990. 47

[2] "Design, 6. a." OED Online. `http://dictionary.oed.com/cgi/entry/50061846`, July 2010. 4

[3] "Framework, 1. a." OED Online. `http://dictionary.oed.com/cgi/entry/50089406`, Sept. 2010. 38

[4] "Technique, 3" OED Online. `http://dictionary.oed.com/cgi/entry/50248085`, Sept. 2010. 73

[5] Mark S. Ackerman. The intellectual challenge of CSCW: the gap between social requirements and technical feasibility. *Human-Computer Interaction*, **15**(2):179–203, September 2000. 21

[6] Anne Adams. *Users' Perceptions of Privacy in Multimedia Communications.* PhD thesis, University College London, 2001. 76

[7] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, **42**:41–46, 1999. 10

[8] Hans Akkermans and Jaap Gordijn. What is this science called requirements engineering? In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 273 –278. IEEE Computer Society, 2006. 48

[9] Ian Alexander. Initial industrial experience of misuse cases in trade-off analysis. In *Proceedings of the IEEE International Requirements Engineering Conference*, pages 61–68. IEEE Computer Society, 2002. 41

[10] Ian Alexander. Semiautomatic tracing of requirement versions to use cases. In *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering*, 2002. Unpublished workshop proceedings. 41

[11] Ian Alexander. Misuse cases: use cases with hostile intent. *IEEE Software*, **20**(1):58 – 66, 2003. 142

[12] Ian Alexander. Negative scenarios and misuse cases. In Ian. F. Alexander and Neil Maiden, editors, *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*. John Wiley & Sons Ltd, 2004. 82

[13] Ian Alexander and Ljerka Beus-Dukic. *Discovering requirements: how to specify products and services*. John Wiley & Sons, 2009. 23, 78, 138

[14] John Annett. Hierarchical Task Analysis. In Dan Diaper and Neville A. Stanton, editors, *The Handbook of Task Analysis for Human-Computer Interaction*, chapter 3, pages 67–82. Lawrence Erlbaum Associates, 2004. 18, 19

[15] Anonymous. MySQL web site. `http://www.mysql.com`, February 2011. 86

[16] Anonymous. NumPy web site. `http://numpy.scipy.org`, February 2011. 86

[17] Anonymous. Open Web Application Project (OWASP) web site. `http://www.owasp.org`, January 2011. 108

[18] Anonymous. Python web site. `http://www.python.org`, February 2011. 86

[19] Anonymous. wxPython web site. `http://www.wxpython.org`, February 2011. 86

[20] Mikio Aoyama. Persona-Scenario-Goal Methodology for User-Centered Requirements Engineering. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*, pages 185 –194. IEEE Computer Society, 2007. 37

[21] AT&T. Graphviz web site. `http://www.graphviz.org`, March 2011. 91

[22] David E. Avison and Trevor Wood-Harper. *Multiview: An Exploration in Information Systems Development*. McGraw-Hill, 1990. 38

[23] James Backhouse and Gurpreet Dhillon. Structures of responsibility and security of information systems. *European Journal of Information Systems*, **5**(1):2–9, 1996. 9

[24] Liam J. Bannon. From human factors to human actors: the role of psychology and human-computer interaction studies in system design. In Joan M. Greenbaum and Morten Kyng, editors, *Design at work: cooperative design of computer systems*, pages 25–44. L. Erlbaum Associates, 1991. 21

[25] Liam J. Bannon. Approaches to software engineering: A human-centered perspective. In *Proceedings of the 3rd Conference on Human-Centered Software Engineering*, **LNCS 6409**, pages 1–5. Springer, 2010. 196

[26] Scott Barman. *Writing information security policies.* New Riders, 2002. 9, 10

[27] Richard L. Baskerville. Investigating information systems with action research. *Communications of the Association for Information Systems*, **2**(article 19):1–32, 1999. 50

[28] BBC News. Thefts puncture Paris bike scheme. `http://news.bbc.co.uk/1/hi/world/europe/7881079.stm`, 10 February 2009. 86

[29] Kent Beck and Cynthia Andres. *Extreme programming explained: embrace change.* Addison-Wesley, 2nd edition, 2005. 36

[30] Genevieve Bell, Mark Blythe, and Phoebe Sengers. Making by making strange: Defamiliarization and the design of domestic technologies. *ACM Transactions on Computer-Human Interaction*, **12**(2):149–173, 2005. 198

[31] Daniel M. Berry. The safety requirements engineering dilemma. *Proceedings of the Ninth International Workshop on Software Specification and Design*, pages 147–149, 1998. 47

[32] Hugh Beyer and Karen Holtzblatt. *Contextual design: defining customer-centered systems.* Morgan Kaufmann Publishers Inc., 1998. 18, 21, 46, 49, 117, 164

[33] Paul Beynon-Davies. *Information Systems: An Introduction to Informatics in Organisations.* Palgrave, 2002. 9

[34] Colin Birge. Enhancing research into usable privacy and security. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 221–226. ACM, 2009. 12

[35] Andrew Blyth. Using stakeholders, domain knowledge, and responsibilities to specify information systems' requirements. *Journal of Organizational Computing and Electronic Commerce*, **9**(4):287–296, 1999. 9

[36] John B. Bowles. The personification of reliability, safety, and security. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 161–166. IEEE Computer Society, 2007. 41

[37] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, **8**(3):203–236, 2004. 31

[38] Sacha Brostoff and Martina Angela Sasse. Safe and sound: a safety-critical approach to security. In *Proceedings of the 2001 New Security Paradigms Workshop*, pages 41–50. ACM, 2001. 10, 47

[39] Lisa Brownsword, Carol C. Woody, Christopher J. Alberts, and Andrew P. Moore. A Framework for Modeling the Software Assurance Ecosystem: Insights from the Software Assurance Landscape Project. Technical Report CMU/SEI-2010-TR-028, Software Engineering Institute, 2010. 197

[40] Janet E. Burge, John M. Carroll, Raymond McCall, and Ivan Mistrik. *Rationale-Based Software Engineering.* Springer, 2008. 138

[41] Shawn A. Butler. Security design: Why it's hard to do empirical research. In *Workshop on Using Multidisciplinary Approaches in Empirical Software Engineering Research. Available at http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/secure.butler.pdf*, 2002. Unpublished workshop proceedings. 48

[42] William Buxton. *Sketching User Experiences: getting the design right and the right design.* Elsevier, 2007. 49

[43] Paul Cairns and Anna L. Cox. Applying old research methods to new problems. In Paul Cairns and Anna L. Cox, editors, *Research Methods for Human-Computer Interaction*, chapter 11, pages 212–220. Cambridge University Press, 2008. 46

[44] John W. Castro, Silvia T. Acuna, and Natalia Juristo. Integrating the personas technique into the requirements analysis activity. In *Proceedings of the 2008 Mexican International Conference on Computer Science*, pages 104–112. IEEE Computer Society, 2008. 37, 41

[45] Pravir Chandra. OpenSAMM Web Site. `http://www.opensamm.org`, March 2011. 197

[46] Christopher N. Chapman and Russell P. Milham. The persona's new clothes: Methodological and practical arguments against a popular method. *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting. Available at http://cnchapman.files.wordpress.com/2007/03/chapman-milham-personas-hfes2006-0139-0330.pdf*, pages 634–636, 2006. 20, 138, 163

[47] Peter Checkland and Jim Scholes. *Soft systems methodology in action.* John Wiley & Sons, 1990. 10, 78

[48] Betty H. C. Cheng and Joanne M. Atlee. Research directions in requirements engineering. In *Future of Software Engineering 2007*, pages 285–303. IEEE Computer Society, 2007. 48

[49] Herman Chernoff. The Use of Faces to Represent Points in K-Dimensional Space Graphically. *Journal of the American Statistical Association*, page 68, 1973. 104

[50] Raoul Chiesa, Stefania Ducci, and Silvio Ciappi. *Profiling hackers: the science of criminal profiling as applied to the world of hacking.* Auerbach Publications, 2009. 199

[51] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering.* Kluwer Academic, 2000. 29

[52] Alistair Cockburn. *Writing Effective Use Cases.* Addison-Wesley, 2001. 34, 35, 39

[53] Gilbert Cockton. Value-centred HCI. In *Proceedings of the 3rd Nordic conference on Human-computer interaction*, pages 149–160. ACM, 2004. 15

[54] Gilbert Cockton. Revisiting usability's three key principles. In *CHI '08 extended abstracts on Human factors in computing systems*, pages 2473–2484. ACM, 2008. 20

[55] Mike Cohn. *User stories applied: for agile software development.* Addison-Wesley, 2004. 36

[56] Larry L. Constantine. Activity modeling: Towards a pragmatic integration of activity theory with usage-centered design. Technical report, Laboratory for Usage-centered Software Engineering, 2006. 22

[57] Larry L. Constantine and Lucy A. D. Lockwood. *Software for use: a practical guide to the models and methods of usage-centered design.* Addison-Wesley, 1999. 18, 22

[58] Greg Conti. *Security Data Visualization: Graphical Techniques for Network Analysis.* No Starch Press, 2007. 44

[59] Control Engineering UK. 'Stuxnet' Trojan Targets Siemens WinCC. `http://www.controlenguk.com/article.aspx?ArticleID=35267`, 20 July 2010. 161

[60] Alan Cooper. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity (2nd Edition).* Pearson Higher Education, 1999. 19, 20

[61] Alan Cooper, Robert Reimann, and David Cronin. *About Face 3: The Essentials of Interaction Design.* John Wiley & Sons, 2007. 18, 20, 37, 60, 87, 117, 139, 163

[62] John R. Cooper, Seok-Won Lee, Robin A. Gandhi, and Orlena Gotel. Requirements engineering visualization: A survey on the state-of-the-art. In *Proceedings of the 4th International Workshop on Requirements Engineering Visualization*, pages 46 –55. IEEE Computer Society, 2009. 44

[63] Juliet M. Corbin and Anselm L Strauss. *Basics of qualitative research : techniques and procedures for developing grounded theory.* Sage Publications, Inc., 3rd edition, 2008. 76, 164

[64] Steven L. Cornford. Managing Risk as a Resource using the Defect Detection and Prevention process. In *Proceedings of the 4th International Conference on Probabilistic Safety Assessment and Management*, pages 1609–1614. Springer, 1998. 43

[65] Joëlle Coutaz and Gaëlle Calvary. Hci and software engineering: Designing for user interface plasticity. In Andrew Sears and Julie A. Jacko, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, pages 1107–1125. Lawrence Erlbaum Associates, 2008. 15

[66] Charles Crichton, Jim Davies, Jeremy Gibbons, Steve Harris, Andrew Tsui, and James Brenton. Metadata-driven software for clinical trials. In *Proceedings of the 2009 ICSE Workshop on Software Engineering in Health Care*, pages 1–11. IEEE Computer Society, 2009. 135

[67] Adele da Veiga and Jan H. P. Eloff. An information security governance framework. *Information Systems Management*, **24**(4):361–372, 2007. 9

[68] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, **20**(1-2):3 – 50, 1993. 28

[69] John M. Darley and Bibb Latané. Norms and normative behaviour: field studies of social interdependence. In Leonard Berkowitz and Jacqueline Macaulay, editors, *Altruism and Helping Behaviour*. Academic Press, 1970. 145

[70] Folker den Braber, Ida Hogganvik, Mass S. Lund, Ketil Stølen, and Fredrik Vraalsen. Model-based security analysis in seven steps - A guided tour to the CORAS method. *BT Technology Journal*, **25**(1):101–117, 2007. 2, 42, 82, 108, 181

[71] Detica. *The Cost of Cyber Crime: A Detics Report in Partnership with the Office of Cyber Security and Information Assurance in the Cabinet Office*. UK Cabinet Office, 2011. 1

[72] James A Dewar. *Assumption-based planning: a tool for reducing avoidable surprises*. Cambridge University Press, 2002. 144

[73] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, **5**(1):4–7, 2001. 57

[74] Dan Diaper. Understanding Task Analysis for Human-Computer Interaction. In Dan Diaper and Neville A. Stanton, editors, *The Handbook of Task Analysis for Human-Computer Interaction*, pages 5–47. Lawrence Erlbaum Associates, 2004. 18

[75] DocBook Technical Committee. DocBook Schema Specification. `http://www.docbook.org/schemas/5x`, 2008. 111

[76] John Dowell and John Long. Towards a conception for an engineering discipline of human factors. *Ergonomics*, **32**(11):1513–1535, 1989. 21

[77] Anthony Dunne and Fiona Raby. *Design Noir: The Secret Life of Electronic Objects*. August / Birkhaeuser, 2001. 142

[78] Steve Easterbrook, Eric Yu, Jorge Aranda, Yuntian Fan, Jennifer Horkoff, Marcel Leica, and Rifat Abdul Qadir. Do viewpoints lead to better conceptual models? an exploratory case study. In *Proceedings of the 13th IEEE International Requirements Engineering Conference*, pages 199–208. IEEE Computer Society, 2005. 33

[79] Golnaz Elahi and Eric Yu. Trust trade-off analysis for security requirements engineering. In *Proceedings of the 17th IEEE International Requirements Engineering Conference*, pages 243 –248. IEEE Computer Society, 2009. 32

[80] Golnaz Elahi, Eric Yu, and Nicola Zannone. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. *Requirements Engineering*, **15**(1):41–62, 2010. 32, 33

[81] Hans-Erik Eriksson and Magnus Penker. *Business modeling with UML: business patterns at work*. John Wiley & Sons, 2000. 22

[82] Shamal Faily. Towards requirements engineering practice for professional end user developers: a case study. In *Proceedings of the 3rd International Workshop on Requirements Engineering Education and Training*, pages 38–44. IEEE Computer Society, 2008. 88

[83] Shamal Faily. Integrating requirements and information security. BCS Requirements Engineering Specialist Group Postgraduate Workshop, London, UK (Oral Presentation), 2009. 86

[84] Shamal Faily. Integrating requirements and risk management to analyse contexts of use. Oxford University Computing Laboratory Cake Seminar, Oxford, UK (Oral Presentation), 2009. 86

[85] Shamal Faily. *CAIRIS User Manual*. University of Oxford, Oxford, UK, 2010. 87, 201

[86] Shamal Faily. CAIRIS web site. `http://www.comlab.ox.ac.uk/cairis`, June 2011. 86

[87] Shamal Faily and Ivan Fléchais. Context-Sensitive Requirements and Risk Management with IRIS. In *Proceedings of the 17th IEEE International Requirements Engineering Conference*, pages 379–380. IEEE Computer Society, 2009. 86

[88] Shamal Faily and Ivan Fléchais. Designing and aligning e-science security culture with design. *Information Management and Computer Security*, **18**(5):339–349, 2010. 10, 52

[89] Shamal Faily and Ivan Fléchais. The secret lives of assumptions: Developing and refining assumption personas for secure system design. In *Proceedings of the 3rd Conference on Human-Centered Software Engineering*, **LNCS 6409**, pages 111–118. Springer, 2010. 192

[90] Shamal Faily and Ivan Fléchais. To boldly go where invention isn't secure: applying Security Entrepreneurship to secure systems design. In *Proceedings of the 2010 New Security Paradigms Workshop*, pages 73–84. ACM, 2010. 198

[91] Shamal Faily and Ivan Fléchais. Eliciting Usable Security Requirements with Misusability Cases. In *Proceedings of the 19th IEEE International Requirements Engineering Conference*. IEEE Computer Society, 2011. To Appear. 192

[92] Shamal Faily and Ivan Fléchais. Persona cases: a technique for grounding personas. In *Proceedings of the 29th international conference on Human factors in computing systems*, pages 2267–2270. ACM, 2011. 192

[93] Jon Fairclough, Michael Jones, Uffe Mortensen, Bryan Melton, Adriaan Scheffer, Daniel de Pablo, and Gianfranco Alvisi. ESA software engineering standards and ISO 9001: Theory and practice. *European Space Agency, (Special Publication) ESA SP*, (377):303–308, 1996. 80

[94] Martin S. Feather, Steven L. Cornford, James D. Kiper, and Tim Menzies. Experiences using visualization techniques to present requirements, risks to them, and options for risk mitigation. In *Proceedingss of the 1st International Workshop on Requirements Engineering Visualization*, pages 10–19. IEEE Computer Society, 2006. 43

[95] Martin S. Feather, Stephen Fickas, Anthony Finkelstein, and Axel van Lamsweerde. Requirements and specification exemplars. *Automated Software Engineering*, **4**(4):419–438, 1997. 51

[96] Donald G. Firesmith. Engineering security requirements. *Journal of Object Technology*, **2**(1):53–68, 2003. 23

[97] Ivan Fléchais. *Designing Secure and Usable Systems*. PhD thesis, University College London, 2005. 48, 50, 57, 76, 108, 129, 136, 181

[98] Ivan Fléchais, Cecilia Mascolo, and Martina Angela Sasse. Integrating security and usability into the requirements and design process. *International Journal of Electronic Security and Digital Forensics*, **1**(1):12–26, 2007. 13

[99] Ivan Fléchais and Martina Angela Sasse. Stakeholder involvement, motivation, responsibility, communication: How to design usable security in e-science. *International Journal of Human-Computer Studies*, **67**(4):281 – 296, 2009. 23, 158

[100] Ivan Fléchais, Martina Angela Sasse, and Stephen M. V. Hailes. Bringing security home: a process for developing secure and usable systems. In *Proceedings of the 2003 New Security Paradigms Workshop*, pages 49–57. ACM, 2003. 2

[101] José Fonseca. XDot web site. `http://code.google.com/p/jrfonseca/wiki/XDot`. 91

[102] Institute for Information Infrastructure Protection. 1st Software and Usable Security Aligned for Good Engineering (SAUSAGE) Workshop. `http://www.thei3p.org/events/sausage2011.html`, April 2011. 3

[103] Apache Software Foundation. Subversion web site. `http://subversion.apache.org`, January 2011. 86

[104] Batya Friedman, Peyina Lin, and Jessica K. Miller. Informed consent by design. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and Usability: Designing Secure Systems that People Can Use*. O'Reilly Media, 2005. 77

[105] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995. 11, 49, 92

[106] Robin A. Gandhi and Seok-Won Lee. Visual analytics for requirements-driven risk assessment. In *Proceedings of the 2nd International Workshop on Requirements Engineering Visualization*, pages 46–55. IEEE Computer Society, 2007. 43

[107] Simson L. Garfinkel. *Design principles and patterns for computer systems that are simultaneously secure and usable*. PhD thesis, Cambridge, MA, USA, 2005. Adviser-David D. Clark and Adviser-Robert C. Miller. 10

[108] John Geddes, Sharon Lloyd, Andrew Simpson, Martin Rossor, Nick Fox, Derek Hill, Joseph V. Hajnal, Stephen Lawrie, Andrew McIntosh, Eve Johnstone, Joanna Wardlaw, Dave Perry, Rob Procter, Philip Bath, and Ed Bullmore. NeuroGrid: Using Grid Technology to advance Neuroscience. In *Proceedings of 18th IEEE Symposium on Computer-Based Medical Systems*, pages 570–572. IEEE Computer Society, 2005. 52

[109] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of software engineering*. Prentice Hall, 2nd edition, 2003. 2, 15

[110] Kentaro Go and John M. Carroll. Scenario-Based Task Analysis. In Dan Diaper and Neville A. Stanton, editors, *The Handbook of Task Analysis for Human-Computer Interaction.* Lawrence Erlbaum Associates, 2004. 18

[111] Simon Godik and Tim Moses. EXtensible Access Control Markup Language (XACML) version 1.1, committee specification, August 2003. *http://www.oasis-open.org*, 5 May 2005. 52

[112] Dieter Gollmann. *Computer security.* John Wiley & Sons, 2nd edition, 2006. 9

[113] John D. Gould and Clayton Lewis. Designing for usability: key principles and what designers think. *Communications of the ACM*, **28**(3):300–311, 1985. 17

[114] Jan Gulliksen, Bengt Goransson, Inger Boivie, Jenny Persson, Stefan Blomkvist, and Asa Cajander. Key principles for user-centered systems design. In Ahmed Seffah, Jan Gulliksen, and Michel C. Desmarais, editors, *Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle*, pages 17–35. Springer, 2005. 22

[115] Brent Hailpern and Peri Tarr. Model-driven development: The good, the bad, and the ugly. *IBM Systems Journal*, **45**(3):451–461, 2006. 42

[116] Charles B. Haley. *Arguing Security: A Framework for Analyzing Security Requirements.* PhD thesis, The Open University, 2007. 38

[117] Charles B. Haley, Robin Laney, Jonathan D. Moffett, and Bashar Nuseibeh. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Transactions on Software Engineering*, **34**(1):133–153, 2008. 23, 24, 25, 26, 29

[118] Denis Hatebur, Maritta Heisel, and Holger Schmidt. A Security Engineering Process based on Patterns. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, pages 734–738. IEEE Computer Society, 2007. 25

[119] Knut Haukelid. Theories of (safety) culture revisited-an anthropological approach. *Safety Science*, **46**(3):413–426, 2008. 23

[120] Tuija Helokunnas and Rauno Kuusisto. Information security culture in a value net. In *Proceedings of the 2003 Engineering Management Conference*, pages 190–194. IEEE Computer Society, 2003. 9, 23

[121] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, **28**(1):75–105, 2004. 49

[122] Thomas T. Hewett, Ronald Baecker, Stuart Card, Tom Carey, Jean Gasen, Marilyn Mantei, Gary Perlman, Gary Strong, and William Verplank. *ACM SIGCHI Curricula for Human-Computer Interaction*, chapter 2. ACM, 1996. 14

[123] Matthias Hoffmann, Nikolaus Kuhn, Matthias Weber, and Margot Bittner. Requirements for Requirements Management Tools. In *Proceedings of the 12th IEEE International Requirements Engineering Conference*, pages 301–308. IEEE Computer Society, 2004. 23

[124] Ida Hogganvik. *A graphical approach to security risk analysis.* PhD thesis, University of Oslo, 2007. 43

[125] Karen Holtzblatt and Sandra Jones. Contextual inquiry: a participatory technique for systems design. In D. Schuler and A. Namioka, editors, *Participatory Design: Principles and Practice*, pages 177–210. Lawrence Erlbaum Associates, 1993. 14

[126] Karen Holtzblatt, Jessamyn B. Wendell, and Shelley Wood. *Rapid contextual design: a how-to guide to key techniques for user-centered design.* Elsevier, 2005. 18

[127] Paco Hope, Gary McGraw, and Annie I. Antón. Misuse and abuse cases: getting past the positive. *IEEE Security & Privacy*, **2**(3):90–92, May-June 2004. 35

[128] Siv Hilde Houmb, Shareeful Islam, Eric Knauss, Jan Jürjens, and Kurt Schneider. Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. *Requirements Engineering*, **15**(1):63–93, 2010. 42

[129] IBM. IBM Rational DOORS, 2010. 23, 41

[130] IEC. *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. Parts 1-7.* International Electrotechnical Commission, Switzerland, 1998-2005. 96

[131] IEEE. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, 1990. 22

[132] IEEE. IEEE recommended practice for software requirements specifications. *IEEE Std 830-1998*, Oct 1998. 80

[133] Juhani Iivari, Rudy Hirschheim, and Heinz K. Klein. A paradigmatic analysis contrasting information systems development approaches and methodologies. *Information Systems Research*, **9**(2):164–193, 1998. 38

[134] International Telecommunication Union. *X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks.* International Telecommunication Union, 2005. 52

[135] Magnus Irestig, Henrik Eriksson, and Toomas Timpka. The impact of participation in information system design: a comparison of contextual placements. In *Proceedings of the 8th conference on Participatory design*, pages 102–111. ACM, 2004. 14

[136] ISO. ISO 9241-11. Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11 Guidance on usability. Technical report, 1998. 15, 18, 57, 60, 93

[137] ISO. *ISO/IEC 13407: Human-Centered Design Processes for Interactive Systems*. ISO/IEC, 1999. 2, 17

[138] ISO. *ISO/IEC 27001: Information Technology – Security Techniques – Requirements*. ISO/IEC, 2005. 9, 58, 65

[139] ISO. *ISO/IEC 15408 : Common Criteria for Information Technology Security Evaluation Part 1 : Introduction and general model Version 3.1 Revision 1*. ISO/IEC, 2006. 42

[140] ISO. *ISO/IEC 27002: Information Technology – Security Techniques – Code of Practice for Information Security Management*. ISO/IEC, 2007. 8, 9

[141] ISO. *ISO/IEC 15408 : Common Criteria for Information Technology Security Evaluation Part 2 : Security Functional Components*. ISO/IEC, 2008. 42

[142] ISO. *ISO/IEC 27005: Information Technology – Security Techniques – Information security risk management*. ISO/IEC, 2008. 66

[143] Michael Jackson. *Problem frames: analysing and structuring software development problems*. Addison-Wesley, 2001. 24, 78

[144] Ivar Jacobson. *Object-oriented software engineering: a use case driven approach*. Addison-Wesley, 1992. 34

[145] Helen L. James. Managing information systems security: a soft approach. In *Proceedings of the Information Systems Conference of New Zealand*, pages 10–20. IEEE Computer Society, 1996. 10, 50

[146] Uwe Jendricke and Daniela Gerd tom Markotten. Usability meets security - the identity-manager as your personal security assistant for the internet. In *Proceedings of the 16th Annual Computer Security Applications Conference*, pages 344–353. IEEE Computer Society, 2000. 13

[147] Jan Jürjens. *Secure systems development with UML*. Springer, Berlin, 2005. 42

[148] Carl Kalapesi, Sarah Willersdorf, and Paul Zwillenberg. *The Connected Kingdom: How the Internet is Transforming the U.K. Economy*. Boston Consulting Group, 2010. 1

[149] Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *IEEE Software*, **14**(5):67 –74, Sept. 1997. 40

[150] David Kieras. GOMS Models for Task Analysis. In Dan Diaper and Neville A. Stanton, editors, *The Handbook of Task Analysis for Human-Computer Interaction*, chapter 4, pages 83–116. Lawrence Erlbaum Associates, 2004. 18

[151] Eric Knauss, Daniel Lubke, and Sebastian Meyer. Feedback-driven requirements engineering: The heuristic requirements assistant. In *ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, pages 587–590, Washington, DC, USA, 2009. IEEE Computer Society. 42

[152] Philippe Kruchten. *The Rational Unified Process: An Introduction.* Addison-Wesley, 3rd edition, 2003. 34, 36, 38

[153] Soren Lauesen. *User interface design: a software engineering perspective.* Pearson Addison Wesley, 2005. 15

[154] Nancy Leveson. *Safeware: System Safety and Computers.* Addison-Wesley, Reading, Mass., 1995. 28

[155] Kurt Lewin. Action research and minority problems. *Journal of Social Issues*, **2**(4):34–46, 1946. 48, 50

[156] Jonathan Liebenau and James Backhouse. *Understanding Information: an Introduction.* Macmillan, 1990. 9

[157] Neil Maiden and Sara Jones. The RESCUE Requirements Engineering Process: An Integrated User-Centered Requirements Engineering Process. Version 4.1. Technical report, City University, 2004. 39

[158] Neil Maiden, Sara Jones, Cornelius Ncube, and James Lockerbie. Using i* in Requirements Projects: Some Experiences and Lessons. In Eric Yu, editor, *Social Modeling for Requirements Engineering.* MIT Press, 2011. 34

[159] Andrew Martin, Jim Davies, and Steve Harris. Towards a framework for security in escience. *IEEE e-Science 2010 Conference*, 2010. 135, 136

[160] Tara Matthews, Steve Whittaker, Thomas Moran, and Sandra Yuen. Collaboration personas: A new approach for designing workplace collaboration tools. In *Proceedings of the 29th international conference on Human factors in computing systems*, pages 2247–2256. ACM, 2011. 21

[161] Raimundas Matulevičius, Nicolas Mayer, and Patrick Heymans. Alignment of misuse cases with security risk management. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security*, pages 1397–1404. IEEE Computer Society, 2008. 36

[162] Douglas Maughan. The need for a national cybersecurity research and development agenda. *Communications of the ACM*, **53**(2):29–31, 2010. 2

[163] Alistair Mavin and Neil Maiden. Determining socio-technical system requirements: Experiences with generating and walking through scenarios. In *Proceedings of 11th IEEE Interational Conference on Requirements Engineering*, pages 213–222. IEEE Computer Society, 2003. 39

[164] Roy A. Maxion, Thomas A. Longstaff, and John McHugh. Why is there no science in cyber science? In *Proceedings of the 2010 New Security Paradigms Workshop*, pages 1–6. ACM, 2010. 48

[165] Nicolas Mayer. *Model-based Management of Information System Security Risk.* PhD thesis, University of Namur, 2009. 36, 42, 65, 82, 194

[166] John McDermott and Chris Fox. Using abuse case models for security requirements analysis. In *Proceedings of the 15th Annual Computer Security Applications Conference*, ACSAC '99, pages 55–, Washington, DC, USA, 1999. IEEE Computer Society. 35

[167] Nancy R. Mead. Benefits and challenges in the use of case studies for security requirements engineering methods. *International Journal of Secure Software Engineering*, **1**(1):74–91, 2010. 40

[168] Nancy R. Mead, Eric D. Hough, and Theodore R. Stehney II. Security Quality Requirements Engineering (SQUARE) Methodology. Technical Report CMU/SEI-2005-TR-009, Carnegie Mellon Software Engineering Institute, 2005. 40

[169] Per H. Meland, Daniele G. Spampinato, Eilev Hagen, Egil T. Baadshaug, Krister-Mikael Krister, and Ketil S. Velle. SeaMonster: Providing tool support for security modeling. *Norsk informasjonssikkerhetskonferanse, Universitetet i Agder, Kampus Gimlemoen*, November 2008. 42

[170] Bill Moggridge. *Designing interactions.* MIT Press, 2007. 17

[171] Daniel Laurence Moody, Patrick Heymans, and Raimundas Matulevicius. Improving the effectiveness of visual representations in requirements engineering: An evaluation of i* visual syntax. In *Proceedings of the 17th IEEE International Requirements Engineering Conference*, pages 171–180. IEEE Computer Society, 2009. 33

[172] Fred Moody. *I Sing the Body Electronic: A Year with Microsoft on the Multimedia Frontier.* Penguin USA, 1996. 20

[173] Haralambos Mouratidis and Paolo Giorgini. Secure Tropos: A Security-Oriented Extension of the Tropos Methodology. *International Journal of Software Engineering and Knowledge Engineering*, **17**(2):285–309, 2007. 33

[174] Thomas Muhr. *User's Manual for ATLAS.ti 5.0.* ATLAS.ti Scientific Software Development GmbH, Berlin, 2004. 77

[175] Enid Mumford. *Designing Human Systems for New Technology: The ETHICS Method.* Manchester Business School, 1983. 38

[176] Lisa P. Nathan, Predrag V. Klasnja, and Batya Friedman. Value scenarios: a technique for envisioning systemic effects of new technologies. In *CHI '07: extended abstracts on Human factors in computing systems*, pages 2585–2590. ACM, 2007. 19, 142

[177] Jakob Nielsen. Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. In Randolph G. Bias and Deborah J. Mayhew, editors, *Cost-Justifying Usability*, pages 242–272. Morgan Kaufmann, 1994. 2

[178] Jakob Nielsen and Robert L. Mack. *Usability inspection methods.* John Wiley & Sons, 1994. 13

[179] Helen Nissenbaum. Accountability in a computerized society. In Batya Friedman, editor, *Human Values and the Design of Computer Technology.* Cambridge University Press, 1997. 59

[180] Joan C. Nordbotten and Matha E. Crosby. The effect of graphic style on data model interpretation. *Information Systems Journal*, **9**(2):139–156, 1999. 33

[181] Donald A. Norman. Human-centered design considered harmful. *Interactions*, **12**(4):14–19, 2005. 20

[182] Donald A. Norman. Ad-hoc personas & empathetic focus. In John Pruitt and Tamara Adlin, editors, *The persona lifecycle: keeping people in mind throughout product design*, pages 154–157. Morgan Kaufmann, 2006. 20

[183] Bashar Nuseibeh. Weaving together requirements and architectures. *Computer*, **34**(3):115–117, 2001. 70

[184] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 35–46. ACM, 2000. 22

[185] Bashar Nuseibeh, Charles B. Haley, and Craig Foster. Securing the Skies: In Requirements We Trust. *IEEE Computer*, **42**(9):64 –72, Sept. 2009. 156

[186] University of East London. SecTro website. `http://sectro.securetropos.org`, 2010. 42

[187] National Institute of Standards and Technology. *System Protection Profile - Industrial Control Systems v1.0.* National Institute of Standards and Technology, Gaithersburg, Maryland, 2004. 108, 121

[188] University of Toronto. i* web-site. `http://www.cs.toronto.edu/km/istar`, 2010. 29

[189] University of Trento. Si*tool: Security and dependability tropos tool. `http://sesa.dit.unitn.it/sistar_tool/home.php?7`. 42

[190] Dan R. Olsen, Jr. Evaluating user interface systems research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 251–258, New York, NY, USA, 2007. ACM. 195

[191] Mairie De Paris. velib': Dossier de presse Anglais. `http://www.velib.paris.fr/content/download/777/5528/version/1/file/Dossier+de+presse+Anglais.pdf`. 86

[192] David L. Parnas and Paul C. Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, **12**(2):251–257, 1986. 38

[193] Colin Potts, Kenji Takahashi, and Anne I. Antón. Inquiry-Based Requirements Analysis. *IEEE Software*, **11**(2):21–32, Mar 1994. 36

[194] Jenny Preece, Yvonne Rogers, and Helen Sharp. *Beyond Interaction Design: Beyond Human-Computer Interaction.* John Wiley & Sons, 2nd edition, 2007. 49

[195] The GNOME Project. PyGTK web site. `http://www.pygtk.org`, February 2011. 86

[196] John Pruitt and Tamara Adlin. *The persona lifecycle: keeping people in mind throughout product design.* Elsevier, 2006. 19, 37, 77, 154

[197] Dave Randall, Richard Harper, and Mark Rouncefield. *Fieldwork for design: theory and practice.* Springer, London, 2007. 21

[198] Trygve Reenskaug. Models-views-controllers. Technical report, Xerox Palo Alto Research Center, 1979. 86, 89

[199] Respect-IT. Objectiver. `http://www.objectiver.com`, 2007. 42

[200] James Robertson and Suzanne Robertson. Volere Requirements Specification Template: Edition 14 - January 2009. `http://www.volere.co.uk/template.htm`, 2009. 23, 39, 78

[201] Suzanne Robertson and James Robertson. *Mastering the requirements process*. Addison-Wesley, 2nd edition, 2006. 80

[202] Marcus K. Rogers. The psyche of cybercriminals: A psycho-social perspective. In Sumit Ghosh and Elliot Turrini, editors, *Cybercrimes: A Multidisciplinary Analysis*. Springer-Verlag Berlin Heidelberg, 2010. 199

[203] Mary Beth Rosson and John M. Carroll. *Usability engineering: scenario-based development of human-computer interaction*. Academic Press, 2002. 18, 77

[204] Mary Beth Rosson and John M. Carroll. Scenario-based design. In Andrew Sears and Julie A. Jacko, editors, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, chapter 53, pages 1041–1060. CRC Press, 2nd edition, 2008. 60, 77

[205] Lillian Røstad. An extended misuse case notation: Including vulnerabilities and the insider threat. In *Proceedings of the 12th International Working Conference on Requirements Engineering*. Essener Informatik Beiträge, 2006. 35

[206] Royal Society of London. *Risk Assessment: A Study Group Report*. Royal Society, 1983. 9

[207] Tobias Ruighaver, Sean Maynard, and Shanton Chang. Organisational security culture: Extending the end-user perspective. *Computers & Security*, **26**(1):56–62, 2 2007. 23

[208] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2nd edition, 2005. 13, 68

[209] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, **63**(9):1278–1308, Sept. 1975. 10

[210] Martina Angela Sasse, Sacha Brostoff, and Dirk Weirich. Transforming the 'weakest link' – a human/computer interaction approach to usable and effective security. *BT Technology Journal*, **19**(3):122–131, July 2001. 11

[211] Holger Schmidt. Threat- and risk-analysis during early security requirements engineering. In *Proceedings of the 5th International Conference on Availability, Reliability and Security*, pages 188–195. IEEE Computer Society, 2010. 26, 29

[212] Bruce Schneier. *Secrets & Lies : Digital Security in a Networked World*. John Wiley & Sons, 2000. 2, 10, 47

[213] Bruce Schneier. *Beyond Fear: Thinking Sensibly about Security in an Uncertain World.* Springer-Verlag New York, 2003. 9

[214] Markus Schumacher, Eduardo Fernandez, Duane Hybertson, and Frank Buschmann. *Security Patterns: Integrating Security and Systems Engineering.* John Wiley & Sons, 2005. 8, 199

[215] Peter Schwartz. *Art of the Long View.* Currency Doubleday, 1991. 197

[216] Michael Scriven. The methodology of evaluation. In Ralph W. Tyler, Robert M. Gagne, and Michael Scriven, editors, *Perspectives of Curriculum Education*, pages 39–83. Rand McNally, 1967. 46

[217] Ahmed Seffah, Jan Gulliksen, and Michel C. Desmarais. *Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle.* Springer, 2005. 22

[218] Ahmed Seffah, Jan Gulliksen, and Michel C. Desmarais. An introduction to human-centered software engineering: Integrating usability in the development process. In Ahmed Seffah, Jan Gulliksen, and Michel C. Desmarais, editors, *Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle*, pages 3–14. Springer, 2005. 21

[219] Ahmed Seffah and Eduard Metzker. The obstacles and myths of usability and software engineering. *Communications of the ACM*, **47**(12):71–76, 2004. 22, 41, 42

[220] Ahmed Seffah, Jean Vanderdonckt, and Michel C. Desmarais, editors. *Human-Centered Software Engineering: Software Engineering Models, Patterns and Architectures for HCI.* Springer, 2009. 22

[221] Abigail Sellen, Yvonne Rogers, Richard Harper, and Tom Rodden. Reflecting human values in the digital age. *Communications of the ACM*, **52**(3):58–66, 2009. 1

[222] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society. 43

[223] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, **10**(1):34–44, 2005. 35, 36

[224] Supriya Singh and Kylie Bartolo. Grounded Theory and User Requirements: a challenge for qualitative research. *Australasian Journal of Information Systems*, **12**(90–102), 2005. 41

[225] SINTEF. Fifth International Workshop on Secure Software Engineering. `http://www.sintef.no/secse`, 2011. 3

[226] Jill Slay and Michael Miller. Lessons learned from the maroochy water breach. In Eric Goetz and Sujeet Shenoi, editors, *IFIP series in Critical Infrastructure Protection*, **253**, pages 73–82. Springer, 2007. 115

[227] Ian Sommerville. *Software Engineering.* Pearson Education Limited, 8th edition, 2007. 38

[228] Ian Sommerville and Pete Sawyer. *Requirements engineering: a good practice guide.* John Wiley & Sons, 1999. 23

[229] Robert Spence. *Information Visualization: Design for Interaction.* Pearson Prentice Hall, 2nd edition, 2007. 108

[230] John M. Spivey. *The Z notation: a reference manual.* Prentice Hall International, 1992. 28

[231] British Standard. BS ISO 7498-2:1989: Information processing systems – Open systems – Interconnection –Basic reference model – Part 2: Security architecture. , 1989. 58

[232] Susan L. Star and James. R. Griesemer. Institutional ecology, "translations" and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39. *Social Studies of Science*, **19**(3):387–420, 1989. 13

[233] Adam Steele and Xiaoping Jie. Adversary Centered Design: Threat Modeling Using Anti-Scenarios, Anti-Use Cases and Anti-Personas. In *Proceedings of the 2008 International Conference on Information & Knowledge Engineering*, pages 367–370. CSREA Press, 2008. 198

[234] John Sterman. *Business Dynamics: System Thinking and Modeling for a Complex World.* McGraw-Hill, 2000. 197

[235] Anselm L Strauss and Juliet M. Corbin. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory.* Sage Publications, Thousand Oaks, 2nd edition, 1998. 48

[236] Ros Strens and John Dobson. How responsibility modelling leads to security requirements. In *Proceedings of the 1992-1993 New Security Paradigms Workshop*, pages 143–149. ACM, 1993. 9, 158

[237] Alistair Sutcliffe. Convergence or competition between software engineering and human computer interaction. In Ahmed Seffah, Jan Gulliksen, and Michel C. Desmarais, editors, *Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle.* Springer, 2005. 22, 24

[238] Frank Swiderski and Window Snyder. *Threat modeling.* Microsoft Press, 2004. 2

[239] John Taylor. Presentation at e-Science Meeting by the Director of the Research Councils, Office of Science and Technology, UK. *Available at: http://www.nesc.ac.uk/nesc/define.html*, 2001. 135

[240] Mary F. Theofanos and Shari L. Pfleeger. Guest Editors' Introduction: Shouldn't All Security Be Usable? *IEEE Security & Privacy*, **9**(2):12–17, 2011. x, 12

[241] Harold Thimbleby. *Press on: principles of interaction programming.* MIT Press, 2007. 15

[242] Harold Thimbleby. User-centered methods are insufficient for safety critical systems. In *HCI and Usability for Medicine and Health Care, Third Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society*, **4799 LNCS**, pages 1–20. Springer, 2007. 20, 22, 43

[243] Harold Thimbleby and Will Thimbleby. Internalist and externalist HCI. In *Proceedings of the 21st British HCI Group Annual Conference*, pages 111–114. British Computer Society, 2007. 21

[244] Kerry-Lynn Thomson and Rossouw von Solms. Information security obedience: a definition. *Computers and Security*, **24**(1):69–75, 2005. 9

[245] Kerry-Lynn Thomson, Rossouw von Solms, and Lynette Louw. Cultivating an organizational information security culture. *Computer Fraud and Security*, **2006**(10):7–11, 2006. 9

[246] Daniela Gerd tom Markotten. User-Centered Security Engineering. In *Proceedings of the 4th EurOpen/USENIX Conference*, 2002. Unpublished workshop proceedings. 12

[247] Inger A Tøndel, Martin G. Jaatun, and Per H. Meland. Security requirements for the rest of us: A survey. *IEEE Software*, **25**(1):20–27, Jan.-Feb. 2008. 23

[248] Stephen Toulmin. *The uses of argument.* Cambridge University Press, updated edition, 2003. 138

[249] Edward R. Tufte. *Envisioning information.* Graphics Press, 1990. 104

[250] Edward R. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative.* Graphics Press, 1997. 107

[251] Phil Turner and Susan Turner. Is stereotyping inevitable when designing with personas? *Design Studies*, **32**(1):30 – 44, 2011. 138

[252] Carnegie Mellon University. Symposium On Usable Privacy and Security. `http://cups.cs.cmu.edu/soups`, 2011. 3

[253] Jenny Ure, Frank Rakebrandt, Sharon Lloyd, and Ali A. Khanban. Usability, the tri-wizard challenge: Recurring scenarios in the design of a healthgrid portal. In *Proceedings of the 2008 Conference on Human System Interactions*, pages 298–305. IEEE Computer Society, 2008. 52

[254] Axel van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th International Conference on Software Engineering*, pages 148–157. IEEE Computer Society, 2004. 28, 199

[255] Axel van Lamsweerde. *Requirements Engineering: from system goals to UML models to software specifications*. John Wiley & Sons, 2009. 27, 28, 62, 68, 79

[256] Axel van Lamsweerde and Emmanuel Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering*, **26**(10):978–1005, 2000. 28

[257] John Viega. Critical infrastructure. In *The myths of security: what the computer security industry doesn't want you to know*, pages 229–230. O'Reilly, 2009. 161

[258] Mark Vincent. Communicating requirements for business: UML or problem frames? In *Proceedings of the 3rd international workshop on Applications and advances of problem frames*, pages 16–22. ACM, 2008. 26

[259] webinos Consortium. webinos web site. `http://webinos.org`, February 2011. 198

[260] Alma Whitten and Doug Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, pages 169–184. USENIX Association, 1999. 10

[261] Roel Wieringa. Design science as nested problem solving. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, pages 1–12. ACM, 2009. 49

[262] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, **11**:102–107, 2005. 48

[263] William M. Wilson, Linda H. Rosenberg, and Lawrence E. Hyatt. Automated quality analysis of natural language requirement specifications. In *Proceedings of Fourteenth Annual Pacific Northwest Software Quality Conference. Available at http://www.pnsqc.org/proceedings/pnsqc1996.pdf*, pages 140–151, 1996. 104

[264] Ka-Ping Yee. Guidelines and strategies for secure interaction design. In Lorrie Faith Cranor and Simson Garfinkel, editors, *Security and Usability: Designing Secure Systems that People Can Use*, pages 247–273. O'Reilly Media, 2005. 10, 11

[265] Yu Yijun, Jan Jürjens, and Jörg Schreck. Tools for traceability in secure software development. In *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 503–504. IEEE Computer Society, 2008. 42

[266] Robert K Yin. *Case study research: design and methods.* Sage Publications, 3rd edition, 2003. 53

[267] Eric Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, pages 226–235. IEEE Computer Society, 1997. 29, 31, 33

[268] Pamela Zave. Classification of research efforts in requirements engineering. *ACM Computing Surveys*, **29**(4):315–321, 1997. 22

[269] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in hci. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 493–502. ACM, 2007. 46, 49

[270] Mary Ellen Zurko. User-centered security: stepping up to the grand challenge. In *Proceedings of the 21st Annual Computer Security Applications Conference*, pages 14–27. IEEE Computer Society, 2005. 13

[271] Mary Ellen Zurko and Richard T. Simon. User-centered security. In *Proceedings of the 1996 New Security Paradigms Workshop*, pages 27–33. ACM, 1996. 11