A meta-model for usable secure requirements engineering.

FAILY, S. and FLÉCHAIS, I.

2010

A Meta-Model for Usable Secure Requirements Engineering

Shamal Faily
Oxford University Computing Laboratory
Wolfson Building
Oxford OX1 3QD, UK
shamal.faily@comlab.ox.ac.uk

Ivan Fléchais
Oxford University Computing Laboratory
Wolfson Building
Oxford OX1 3QD, UK
ivan.flechais@comlab.ox.ac.uk

ABSTRACT

There is a growing recognition of the need for secure software engineering approaches addressing both technical and human factors. Existing approaches to secure software engineering focus on the need for technical security to the detriment of usability. This paper presents the IRIS (Integrating Requirements and Information Security) meta-model, a conceptual model for usable secure requirements engineering. We describe a practical application of the meta-model through a case study in the Critical Infrastructure domain.

1. INTRODUCTION

Human factors are still not sufficiently addressed in the process of engineering secure software systems, and this is not a new problem. Auguste Kerckoffs wrote in 1883 that systems should be *easy to use and require neither mental strain nor the observance of many rules* [1]. Over 35 years ago, Saltzer & Schroeder [2] penned the principle of Psychological Acceptability, stating that the human interface should be designed such that users routinely and automatically apply protection mechanisms correctly. More recent work in HCISec (Human Computer Interaction in Security) [3, 4] describes how unusable controls may introduce vulnerabilities if circumvented or used incorrectly. In spite of this, while it is accepted wisdom that security concerns should be treated as early as possible (preferably when eliciting and specifying requirements), usability concerns remain largely ignored.

Considering human factors during secure systems engineering requires additional attitudes and techniques. Assumptions which may seem reasonable from a security standpoint may be unwarranted from a usability standpoint. For example, when considering how to approach usability design for security in the Critical Infrastructure (e.g. power, water and gas utilities, etc.), Anderson et al. [5] argue that we should think about designing security that *Homer [Simpson] can use safely.* While from a security standpoint this highlights the importance of security in the face of incompetence, it is counterproductive from a usability perspective. It is important to ground usability decisions in information gathered about real people, potentially including them in the design process; comparing these stakeholders to *Homer* marginalises them and undermines their contribution. These caricatures may also introduce unwarranted biases into the specification and design process.

As well as designing for the people who will use them, secure systems also need to be designed for the environments they will be used in. Even though environmental changes may undermine the security in system designs, approaches for specifying requirements remain grounded in the assumption of a single environment representing the real world [6]. Unfortunately, the real world is complex: a system may be situated in different countries, or be used by people with different cultural backgrounds and different perceptions of security and its importance. Environments within which a system will run are often the least understood aspect of a proposed design, and tools and techniques for reasoning about these lack maturity [7].

Approaches to specifying requirements for systems which are usable and secure need to align concepts from within Usability Engineering and Security with those used to elicit and specify requirements. Encapsulating these concepts and their associations within a conceptual model is a first step towards developing Requirements Engineering processes for usable security, and tool-support for managing such a process. This paper presents the IRIS (Integrating Requirements and Information Security) meta-model, 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 in different contexts. In section 2, we describe the related work our meta-model is founded upon. In section 3, we present the meta-model and its component parts, which we validate in section 4 using a Critical Infrastructure case study. In section 5, we briefly describe how the meta-model supports inter-related qualitative and quantitative analysis of risks and tasks.

2. BACKGROUND

2.1 Security Requirements Engineering Meta-Models

Existing meta-models for Security Requirements Engineering [8, 9, 10] do not explicitly consider usability needs. It is, therefore, important to determine the associations relat-

ing Security Requirements Engineering concepts with those used in Usability Engineering. We cannot, however, devise any rational conceptual model without thinking about the techniques used to elicit and refine data into the requisite concepts. A number of such techniques have been proven useful in Usability Engineering as well as Security Requirements Engineering. Task analysis is a core technique for understanding the impact of work performance on human agents [11]. Tasks can be modelled using scenarios, which are arguably the most common representation shared by Usability and Requirements Engineering for eliciting empirical data. Scenarios have been used extensively to help elicit and model requirements [12], and have also been used to describe how a system may be attacked or misused [13]. Closely allied to tasks are the goals a human agent wishes to achieve by carrying out specific tasks. Goal Oriented Requirements Engineering techniques are useful for validating the completeness of requirements [14] and modelling stakeholder rationale [15], as well as building threat trees [16] and modelling vulnerabilities and their effect [17].

Mayer's work on the Information System Security Risk Management (ISSRM) modelling language [10] 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 [18], and goal-modelling approaches [19]. This work has also attempted to integrate existing efforts on meta-modelling for risk analysis. On first inspection, this 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 certain concepts are missing and others need re-evaluating.

2.2 Contexts of Use

For systems to be situated in their environments, they need to be designed with their *contexts of use* in mind; these are the users, tasks, equipment, and social and physical environments where a system is used [20]. While the notion of *Context of Use* is key within Usability Engineering, it is largely unused in Requirements Engineering. For example, Ali et al. [21] considered how the notion of context can be used to integrate goal, Problem Frame, and feature models, but this work is limited to using a textual context description to guide analysis decisions.

Attempts have been made by the HCI (Human Computer Interaction) community to integrate data about context of use into the larger design process [22]. This work found that different models are better at representing different aspects of context of use than others, and that contexts of use should be modelled as entities in their own right.

2.3 Concept Alignment Challenges

A key issue in aligning Security Requirements Engineering with Usability Engineering is understanding how a given concept may have a different importance or meaning depending on whether a usability, security or requirements perspective is taken. For example, we may agree that a *goal* represents a statement of intent; but intent with respect to what and whom? Does a goal represent the intentions of somebody using the system, somebody designing the system, or someone with the task of turning a goal into working software? Different levels of emphasis may be placed on these potentially ambiguous terms. A security engineer may be happy to assume that a *user* operates a control, a requirements engineer may want to know more about the roles fulfilled by that user, but a usability engineer needs to understand the human characteristics of the user, his or her behaviour, and the context within which the system is used.

A further area that needs clarification is the relationship between people and their values, and goals. Contemporary thinking in HCI [23] argues that while human values remain core to usability, our changing relationship with technology means determining what these might be is harder than ever. The interaction between people and the system works on many different levels, each of which provides different opportunities for interaction. These opportunities are, however, deeply contextual and based not only on the person, but the physical or social environment surrounding the interaction. While goal-oriented approaches are useful for reasoning about system requirements, they capture little contextual information, and cast people as abstract actors and roles. We would argue that this is a good state of affairs: goals are a useful vehicle for refining and specifying requirements, but we do not believe they are the right tool for capturing empirical usability data.

Hinds [24] suggests that we should consider empirical data as clues to the true nature of requirements; these requirements exist in the problem domain independently of stakeholders. This position implies that concepts relating to the elicitation of empirical data should be disjoint from those used to specify goals and requirements. However, stakeholders still need to understand the rationale behind requirement specification decisions, which necessitates traceability from empirical usability data to goals and requirements. Therefore, we need to devise a conceptual model which describes how usability artifacts relate to requirements artifacts.

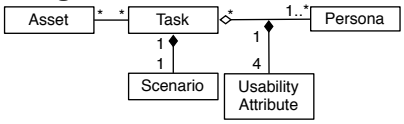
3. IRIS META-MODEL

The IRIS meta-model is a conceptual model for usable secure requirements engineering. This meta-model extends existing work in Security Requirements Engineering in two ways. First, concepts are included which allow the usability of tasks, and the usability impact of security design decisions to be modelled. Second, the meta-model explicitly defines concepts and associations which allow a context of use to be modelled.

Two guiding principles were followed when devising the IRIS meta-model. First, where possible, we extend related meta-models in Security Requirements Engineering so as to reuse existing concepts. Second, we apply as much parsimony as possible when declaring model concepts; in some cases, this involves simplifying relationships to make the conceptual associations clearer. For example, in the risk meta-model (section 3.3), we define a risk as a combination of a single threat and vulnerability. This differs from the ISSRM concept of risk, which combines a single cause with one or more impacts, where a cause is associated with a single threat and one or more vulnerabilities. As section 3.3 will illustrate, supplemental concepts are used to capture the knowledge lost by removing these concepts.

For clarity, we have sub-divided the meta-model into five views. Four of these – Task, Goal, Risk, and Responsibility – correspond to different perspectives of a secure system's context of use. The fifth centres on the axial concept of *En-*

Figure 1: Task Meta-Model



vironment. By introducing the environment as a conceptual type, and allowing other concepts to be associated with it, we can develop complete models of a context of use.

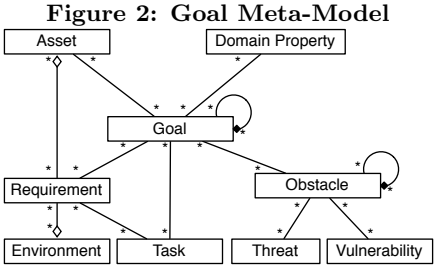
Cross-cutting each of these views is the concept of *Asset*, which represents artifacts which must be safeguarded by the system being specified; these may also represent components forming part of the system. Because stakeholders value certain properties of an asset over others, we associate one or more security attributes to each asset; these attributes reflect the security properties that stakeholders wish to preserve in the system being specified; these attributes are Confidentiality, Integrity, Availability, and Accountability. Each attribute is assigned a value of Low, Medium, or High, and each type of value should be agreed before being used. Closely associated with Asset is the additional cross-cutting concept of *Task*, which represents work being carried out within a context of use. Subsequent sections will illustrate how information about tasks can be used to measure the usability impact of security design decisions.

3.1 Task Meta-Model

The Task Meta-Model, illustrated in figure 1, 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*. A persona is a descriptive model of how indicative users behave, think, what they wish to accomplish, and why [25]. Personas sensitise participants 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. Personas are developed from rich empirical data about stakeholders 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 being specified. 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 (how efficient does the persona find the task), effectiveness (how effective is the task at meeting its objective), and satisfaction (how happy does the persona feel about the task). These attributes are elicited by assigning categorical values 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 tasks. 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 2: Goal Meta-Model



3.2 Goal Meta-Model

Many concepts in the Goal Meta-Model, illustrated in figure 2, are based on the KAOS (Knowledge Acquisition in autOmated Specification) method [26]. KAOS was chosen over alternative goal-oriented notations for two reasons. First, unlike i* [15] derived approaches, KAOS defines goals as prescriptive descriptions 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 reports [14], the KAOS modelling notation is compliant with UML and, by extension, compatible with modelling notations commonly found in industry.

In IRIS, we define a goal as a prescriptive statement of intent that the system should satisfy through the co-operation of its agents [14]. 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 3.1) and security attributes (section 3.3). In KAOS, requirements are refined goals under the responsibility of a single agent. We choose to make the distinction between goal and requirement explicit by treating the latter as an independent concept. This has a number of benefits. First, requirements may arise during analysis independent of any goal refinement or discussion about tasks. 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.

Domain properties are descriptive statements about the problem world [14]. In IRIS, we use these to capture assumptions satisfied by the system, or requirements 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 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 the Problem Frames approach where requirements reference certain phenomena in problem diagrams [27]; this reference indicates that a requirement references or constrains the related domain in the context being modelled. In certain cases, however, require-

Figure 3: Risk Meta-Model

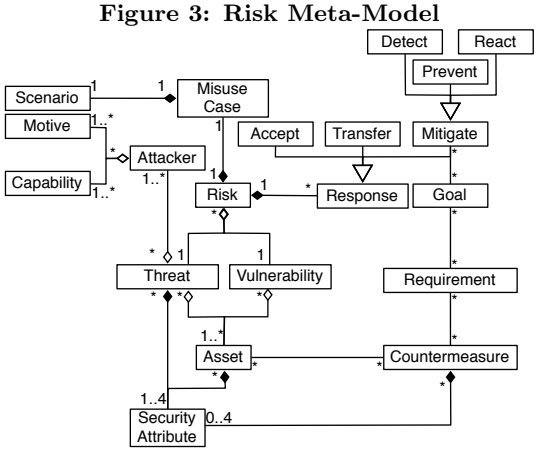
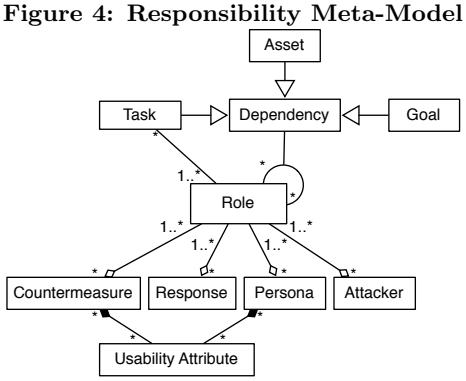


Figure 4: Responsibility Meta-Model



ments may be elicited which do not relate to assets. These may be requirements refined from a goal, non-functional requirements which affect the system in general, or a requirement implying functionality which does not yet exist. 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, goals in KAOS may conflict with obstacles; obstacles are conditions representing undesired behaviour and prevent an associated goal from being achieved [28]. By refining obstacles, candidate threats and vulnerabilities may be defined. This approach is similar, but not identical to, Van Lamsweerde's concept of anti-goals [29]. While anti-goals arise exclusively from malicious intent, obstacles may arise from accidental error as well.

3.3 Risk Meta-Model

The Risk Meta-Model, illustrated in figure 3, models the elements contributing to the definition of risks; it also incorporates concepts for responding and mitigating these risks.

A single risk aggregates a single threat and vulnerability. This reflects a threat exploiting a vulnerability; 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 the many reports in the literature and media. The attackers behind a threat target assets with respect to the properties they want to exploit. 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; these are used to describe risk impact. 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, it also *sanity checks* 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. Risks can be accepted if we are prepared to accept the consequence of its impact, transferred if the responsibility for dealing with it is out of scope, or mitigated if treating the risk has a bearing on the system specification. Strategies for mitigating a risk may be preventing or avoiding it, detecting it, or reacting to it. Choosing to mitigate a response is synonymous to intentionally specifying that the system shall manage the risk as part of its design. Consequently, we can associate goals with mitigating responses. As section 3.2 indicates, these goals can be analysed and refined to requirements mitigating this goal. Countermeasures can then be defined to meet these requirements, and, 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.

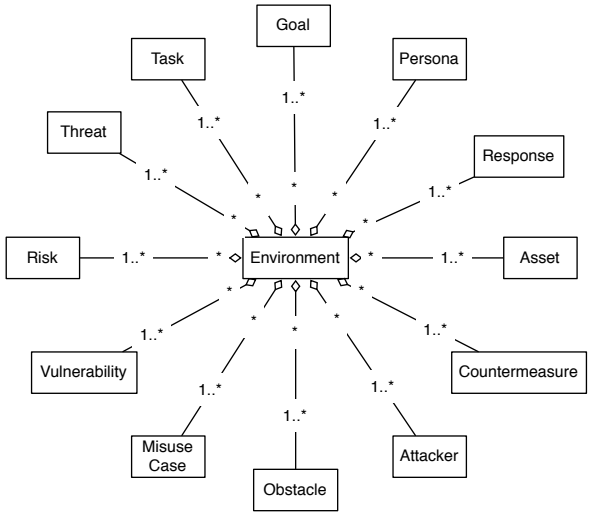
3.4 Responsibility Meta-Model

Roles are a pervasive concept within IRIS. 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 Meta-Model, illustrated in figure 4.

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 context of use is

Figure 5: Environment Meta-Model



shared by a persona participating in one or more tasks in the same context of use. 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 [30].

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 on another (the dependee), for a task, goal, or asset (the dependum). The dependency relationships in IRIS are based on the dependency links used by i* [15]; Van Lamsweerde [14] describes how these relationships can also be used in KAOS to supplement agent responsibility modelling.

3.5 Environment Meta-Model

Many concepts specified in the IRIS Meta-Model are situated within one or more environments as illustrated in figure 5. Together, these concepts represent a Context of Use. 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 in 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 environments.

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

4. CASE STUDY

Water and sewage treatment is controlled by a substantial amount of control software. This software runs on many different devices and locations across a wide geographic area. As part of their responsibility for maintaining the water network, instrument technicians often make software modifications to telemetry outstations, PLCs (Programmable Logic Controllers), and SCADA (Supervisory Control and Data Acquisition) workstations. Without a central strategy for controlling such software, water treatment integrity may be compromised if software is lost, or incorrect software is accidentally, or deliberately, installed on critical instrumentation. However, because maintaining the water network can be physically and mentally demanding, any new technology needs to be situated for the contexts within which these technicians work.

We validated the IRIS Meta-Model by using it to support the specification of requirements for a central repository for control software; this repository will be used to support instrument technicians at a UK water company.

After holding an initial scoping workshop, empirical data was collected by interviewing various stakeholders, and observing real instrument technicians. After analysing this empirical data, the behavioural characteristics of potential users were identified based on the work they need to use the software repository for. Several participatory workshops were then used to elicit the different elements of the meta-model; participants included instrument technicians, software engineers, IT support staff, and information security officers. The elicited data was captured by a software tool building upon this meta-model.

After analysing the empirical data, two different environments were elicited. The first of these, *Planned*, encapsulated a repository context of use during working hours when infrastructure modifications are scheduled in advance. The second environment, *Unplanned*, encapsulated a repository context of use when the infrastructure is modified due to an out-of-hours emergency. For reasons of brevity, this section focuses only on the Planned environment. Because the page limit forbids a detailed report of the design process used, further details of this case study will be elaborated in a future publication.

4.1 Task Analysis

Based on the empirical data, three personas were elicited. Barry represented an instrument technician who modifies software as part of this day-to-day work. Alan represented a commissioning engineer responsible for developing initial releases of PLC software. Eric represented an engineer working in a 2nd line support capacity. Of these representations, Barry was the primary persona the repository needed to be designed for.

Several tasks in the case study involved Barry making infrastructure changes to plant equipment; this work led to control software modifications and, consequently, interaction with the software repository. Although conceptually very similar, tasks were distinct enough for some to be less usable than others. For example, from Barry's perspective, outstation modifications are easier, quicker, involve fewer file changes, and considered less operationally critical than changes to PLCs. Knowledge about these differences was not initially known by many workshop participants. Consequently, authoring the scenario for this task, and defining the associated assets led to discussion on the usability impact of some design decisions.

4.2 Goal and Requirement Elicitation

The scope of the case study was limited to configuration management of control software, but the tasks assumed certain software tools were installed on laptops used by instrument technicians. To ensure this interface requirement was not lost, this was captured as a domain property. Requirements were also specified which were not directly associated with an asset. For example, a requirement was specified and associated with the Planned environment; this indicated that software tools to interface with plant hardware – an out-of-scope asset – needed to be installed as a pre-requisite for repository access.

One of the goals the repository needed to support was the download of particular items of control software; this refined to a sub-goal stating that an instrument technician needs to be authorised to download software for a given geographic area. Participants chose to obstruct this goal by stipulating that a user can download software he is not authorised for. Refining this obstacle led to the identification of a number of vulnerabilities; these included the sharing of login credentials, and unwarranted access to a technician's laptop.

Some goals were refined on a top-down basis. However, many were refined by defining the tasks, followed by the goals and requirements which have to be specified before personas can carry them out securely and without undue hindrance.

4.3 Risk Analysis

Before the assets were defined, values of *Low*, *Medium*, and *High* were associated with *No affect on performance, quality, or pollution*, *Part failure does not affect quality of site or flow*, and *Critical to the business or the quality of the site* respectively. Assets were elicited from task analysis, goal modelling, and general discussion. Assets for different types of software were associated with security attributes based on their criticality; some software assets were associated with low integrity, availability, and accountability attributes, while others were defined as being high.

Attackers were defined based on participants concerns. For example, the fear of malicious insiders led to the definition of a disgruntled instrument technician attacker. Based on their domain knowledge, several motives and capabilities were associated with this inside attacker.

When the PLC software asset was originally defined, participants indicated that the security attributes they wished to preserve for this asset were integrity and availability. Later in the analysis, a risk was defined which explored how an inside attacker might plant a logic bomb in this software to compromise the water treatment process. In addition to tampering with the software, the attacker was keen on covering his tracks, thereby introducing an accountability attribute to the related threat. When a risk incorporating this threat was defined, together with its associated Misuse Case, the importance of PLC software accountability was noted, and an accountability attribute was retrospectively added to it. This risk was treated with a detective mitigation response, such that occurrences of this risk would be detected after the event and, based on this, an associated goal was defined. Following goal-refinement, requirements for peer-reviewing PLC software changes were elicited. Based on these requirements, a software component linking the work-system to the repository was defined as a countermeasure; this ensured that the instrument technician making these change would not be assigned as a peer reviewer. A new asset was defined based on this countermeasure, and an accountability security attribute associated with it.

4.4 Responsibility Modelling

Because personas perform tasks, and certain roles took responsibility for implementing countermeasures, several role-responsibility associations were present.

Based on the restricted scope of the system being specified, the roles in this study were limited to Instrument Technicians and Engineers providing 2nd-line support. Dependency relationships were modelled where an Engineer depends on an Instrument Technician for the different software modification tasks. Engineers are responsible for performing the task of auditing software changes, but this is not possible until an Instrument Technician completes the work requiring audit.

The countermeasure defined in section 4.3 not only mitigated the logic bomb threat, it helped improve the usability of the software modification task to Barry. This improvement arose because submitting a software modification to the repository also led to data being automatically updated in the work scheduling system, thereby reducing the demand of Barry's software modification task.

5. CONCLUSION

There is a need for support when specifying systems which need to be secure and situated for their many contexts of use. Risk analysis supports the specification of such systems, but this analysis needs to be better informed by human factors, and better integrated into the requirements engineering process. To meet this need, we have presented a meta-model for usable secure requirements engineering. We have built upon existing conceptual risk-based Security Requirements Engineering meta-models by introducing concepts which facilitate the modelling of a system's different contexts of use. We have also validated this contribution by applying it to a case study, which demonstrates how our work can be used to tackle contemporary security problems of national and international interest.

Although the results of risk mitigation in section 4 had a positive outcome to the security and usability of the system, assets arising from countermeasures may introduce new vulnerabilities, become open to new or existing threats, or complicate other tasks. In related work [30], we have devised an approach for qualitatively and quantitatively analysing the results of risk and task analysis. By leveraging the conceptual relationships in the IRIS Meta-Model, together with values assigned to tasks, assets, threats, vulnerabilities, and countermeasures, we can infer qualitative categories for risk, and quantitative scores for risks and tasks; these can be used to support visualisation of on-going security and usability analysis, especially as models become more developed.

6. ACKNOWLEDGEMENTS

The research described in this paper was funded by EPSRC CASE Studentship R07437/CN001. We are very grateful to Qinetiq Ltd for their sponsorship of this work. We are also grateful to the Oxford University Computing Laboratory Security Reading Group and the anonymous reviewers for their insightful comments when reviewing this work.

7. REFERENCES

[1] A Kerckhoffs, "La cryptographie militaire", *Journal des Sciences Militaires*, pp. 5–38, 1883.

[2] J.H. Saltzer and M.D. Schroeder, "The protection of information in computer systems", *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, Sept. 1975.

[3] A Adams and MA Sasse, "Users are not the enemy", *Communications of the ACM*, vol. 42, pp. 41–46, 1999.

[4] Alma Whitten and J. D. Tygar, "Why Johnny can't encrypt: a usability evaluation of PGP 5.0", in *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, Berkeley, CA, USA, 1999, pp. 14–14, USENIX Association.

[5] Ross Anderson and Shailendra Fuloria, "Security economics and critical national infrastructure", in *Eight Workshop on the Economics of Information Security (WEIS 2009)*, 2009.

[6] Pamela Zave and Michael Jackson, "Four dark corners of requirements engineering", *ACM Trans. Softw. Eng. Methodol.*, vol. 6, no. 1, pp. 1–30, 1997.

[7] Betty H. C. Cheng and Joanne M. Atlee, "Research directions in requirements engineering", in *FOSE '07: 2007 Future of Software Engineering*, Washington, DC, USA, 2007, pp. 285–303, IEEE Computer Society.

[8] D. Firesmith, "Specifying reusable security requirements", *Journal of Object Technology*, vol. 3, no. 1, pp. 61–75, 2004.

[9] Daniel Mellado, Eduardo Fernández-Medina, and Mario Piattini, "A common criteria based security requirements engineering process for the development of secure information systems", *Computer Standards & Interfaces*, vol. 29, no. 2, pp. 244 – 253, 2007.

[10] Nicolas Mayer, *Model-based Management of Information System Security Risk*, PhD thesis, University of Namur, 2009.

[11] D. Diaper, "Understanding Task Analysis for Human-Computer Interaction", in *The Handbook of Task Analysis for Human-Computer Interaction*, Dan Diaper and Neville A. Stanton, Eds., pp. 5–47. Lawrence Erlbaum Associates, 2004.

[12] Ian. F. Alexander and Neil Maiden, Eds., *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, John Wiley & Sons Ltd, 2004.

[13] Ian Alexander, "Negative scenarios and misuse cases", in *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, Ian. F. Alexander and Neil Maiden, Eds. John Wiley & Sons Ltd, 2004.

[14] A. van Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*, John Wiley, Hoboken, NJ, 2009.

[15] Eric Yu, *Modeling Strategic Relationships for Process Reengineering*, PhD thesis, University of Toronto, 1995.

[16] Axel van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models", in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, Washington, DC, USA, 2004, pp. 148–157, IEEE Computer Society.

[17] Golnaz Elahi, Eric Yu, and Nicola Zannone, "A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities", *Requirements Engineering*, vol. 15, no. 1, pp. 41–62, 2010.

[18] R. Matulevičius, N. Mayer, and P. Heymans, "Alignment of misuse cases with security risk management", *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pp. 1397–1404, March 2008.

[19] Raimundas Matulevičius, Nicolas Mayer, Haralambos Mouratidis, Eric Dubois, Patrick Heymans, and Nicolas Genon, "Adapting Secure Tropos for Security Risk Management in the Early Phases of Information Systems Development", in *CAiSE '08: Proceedings of the 20th international conference on Advanced Information Systems Engineering*, Berlin, Heidelberg, 2008, pp. 541–555, Springer-Verlag.

[20] *ISO/IEC 13407: Human-Centered Design Processes for Interactive Systems*, ISO/IEC, 1999.

[21] Raian Ali, Yijun Yu, Ruzanna Chitchyan, Armstrong Nhlabatsi, and Paolo Giorgini, "Towards a Unified Framework for Contextual Variability in Requirements", *3rd International Workshop on Software Product Management (IWSPM09), Atlanta, USA*, 2009.

[22] Gilbert Cockton, "Grounded design: Integrating models and evaluation", *Interacting in the large: Developing a framework for integrating models in HCI, ACM CHI'99 Workshop*, 1999.

[23] Abigail Sellen, Yvonne Rogers, Richard Harper, and Tom Rodden, "Reflecting human values in the digital age", *Commun. ACM*, vol. 52, no. 3, pp. 58–66, 2009.

[24] Chris Hinds, "The case against a positivist philosophy of requirements engineering", *Requirements Engineering*, vol. 13, no. 4, pp. 315–328, 2008.

[25] Alan Cooper, Robert Reimann, and David Cronin, *About Face 3: The Essentials of Interaction Design*, Wiley, 2007.

[26] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas, "Goal-directed requirements acquisition", *Science of Computer Programming*, vol. 20, no. 1-2, pp. 3 – 50, 1993.

[27] M. A Jackson, *Problem frames : analysing and structuring software development problems*, Addison-Wesley/ACM Press, Harlow, England, 2001.

[28] A. van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering", *Software Engineering, IEEE Transactions on*, vol. 26, no. 10, pp. 978–1005, 2000.

[29] Axel van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models", in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, Washington, DC, USA, 2004, pp. 148–157, IEEE Computer Society.

[30] Shamal Faily and Ivan Fléchais, "Analysing and Visualising Security and Usability in IRIS", in *Availability, Reliability and Security, 2010. ARES 10. Fifth International Conference on*, 2010.