Threat Analysis in Goal-Oriented Security Requirements Modelling

Per Håkon Meland, SINTEF ICT, Norway*
Elda Paja, University of Trento, Italy
Erlend Andreas Gjære, SINTEF ICT, Norway
Stéphane Paul, Thales, France
Fabiano Dalpiaz, Utrecht University, the Netherlands
Paolo Giorgini, University of Trento, Italy

Abstract

Goal and threat modelling are important activities of security requirements engineering: goals express why a system is needed, while threats motivate the need for security. Unfortunately, existing approaches mostly consider goals and threats separately, and thus neglect the mutual influence between them. In this paper, we address this deficiency by proposing an approach that extends goal modelling with threat modelling and analysis. We show that this effort is not trivial and a trade-off between visual expressiveness, usability and usefulness has to be considered. Specifically, we integrate threat modelling with the socio-technical security modelling language (STS-ml), introduce automated analysis techniques that propagate threats in the combined models, and present tool support that enables reuse of threats facilitated by a threat repository. We illustrate our approach on a case study from the Air Traffic Management (ATM) domain, from which we extract some practical challenges. We conclude that threats provide a useful foundation and justification for the security requirements that we derive from goal modelling, but this should not be considered as a replacement to risk assessment. The usage of goals and threats early in the development process allows raising awareness of high-level security issues that occur regardless of the chosen technology and organizational processes.

Keywords: requirements engineering; threats; security; ATM; analysis; reuse

1. Introduction

Modern systems are becoming more and more complex and dynamic, as they involve a multitude of autonomous subsystems and human actors that interact in unpredictable manners (Sommerville et al., 2012). These large and complex systems are highly exposed to malicious intents, and they need to exhibit adaptive behaviour at runtime to continue delivering their purpose without failing (Dalpiaz, Giorgini, & Mylopoulos, 2013).

As in any engineering discipline, early awareness and analysis of potential problems is beneficial to system design, enabling the development of more robust systems. We investigate the usage of threat modelling and analysis in goal-oriented security requirements engineering. This helps not only the elicitation of security requirements, but also the definition of adaptation triggers, i.e., the circumstances under which a system shall adapt.

* Corresponding author

Threat modelling is typically regarded as the analysis of how a system can be exploited in malicious ways. However, as there is no well-accepted standard for conducting threat modelling, the chosen technique is subject to trade-offs that take into account the analysis' purpose (Meland & Gjære, 2012). Threat modelling can, for instance, be asset-centric, attacker-centric, or software-centric (Shostack, 2008). Though a number of somewhat overlapping threat modelling techniques and approaches exist, there is general consensus that (i) threat awareness is of great benefit for performing risk assessment and for eliciting security requirements in the early phases of the software development lifecycle, and (ii) threat modelling and analysis should be repeated as more information about the system becomes available.

Goal modelling is the a state-of-the-art technique in requirements engineering (E. Yu & Mylopoulos, 1998) to understand *why* a certain requirement exists and how it is related to the goals and needs of stakeholders. Moreover, goal modelling comes with refinement mechanisms that support the clarification process, and offers techniques to identify conflicts early in the system development.

Goal models have been extensively used in security requirements too (Giorgini, Massacci, Mylopoulos, & Zannone, 2005), (Mouratidis & Giorgini, 2007), (Liu, Yu, & Mylopoulos, 2003), (Lamsweerde, 2004). However, their combined usage with threats has not been adequately investigated yet and typically goal modelling and threat modelling are conducted as independent activities.

The research question we address in this paper is "*to what extent should we include threats in goal-oriented modelling?"* We believe there is no straightforward answer to this question, and we argue that risk assessment shall be conducted as a separated activity, and not as part of goal modelling (as, for instance, in (Asnar, Giorgini, & Mylopoulos, 2011)), for different reasons. Firstly, when adding additional concepts to a modelling language, we need to consider the impact on its complexity and usability (Moody, 2009). For instance, Moody defines *visual expressiveness* to be the number of visual variables used in a notation. Having a rich vocabulary is of great value when you want to describe necessary details, improving usefulness, but requires more effort to learn; hence it could also affect usability. Secondly, risk assessment deals with tangible assets such as processes and systems, while goal modelling represents the motivational component of the stakeholders, which is of abstract and intangible nature.

Our baseline is the socio-technical security modelling language (STS-ml), a goal-oriented language for the security requirements of complex socio-technical systems (Dalpiaz, Paja, & Giorgini, 2011), (Paja, Dalpiaz, & Giorgini, 2013). We made this choice because STS-ml (i) contains primitives that support complex systems, (ii) includes a well-defined methodology that accompanies the modelling language, (iii) comes with a stable modelling and analysis tool that supports end users, and (iv) has been refined using an iterative development process that has considerably considered evaluations with industrial partners (Trösterer et al., 2012).

The contributions of this paper are as follows:

- revised version of the STS-ml language that includes modelling primitives to represent threats and analysis techniques that combine goals and threats;
- tool support for threat propagation as well as the reuse of threats stored in a threats repository;

- a case study in the domain of Air Traffic Management (ATM), where we discuss benefits and limitations from our experience.

The rest of the paper is structured as follows. In Section 2, we present the ATM case study. In Section 3, we describe our baseline, the STS-ml language. In Section 4, we introduce our modelling extension and the support to threats reuse. In Section 5, we detail the automated analysis that involves threat propagation. In Section 6, we discuss related work, extensively report on our experience in the ATM domain, and present future directions. In Section 7, we conclude the paper.

2. Case study: Air Traffic Management

In this paper we have chosen to illustrate the use of threats in goal modelling with a case study from the Air Traffic Management (ATM) domain. Like many large IT systems, ATM systems are increasingly complex, pervasive and important in our everyday lives, rising stakeholders' concerns about security risks in proportion. Unlike IT systems from other domains, the European Air Navigation Services and their supporting systems are currently undergoing a small revolution, a paradigm shift, most notably through the introduction of the System Wide Information Management (SWIM) (Eurocontrol, 2013).

ATM was a closed world, a fortress isolated from all threatening forces. Security studies were deemed useless because all ecosystem actors were trusted, and because technical systems only had ad-hoc point-to-point communications with other trusted technical systems. Security awareness in the ATM domain really started in 2001, after the 9/11 events in the US: suddenly it became obvious that all ecosystem actors (for example pilots) could not be blindly trusted. SWIM takes ATM security to a new level. In the close future, SWIM will enable collaboration and data exchange between numerous and untrusted players and/or technical systems, typically through a client-server paradigm. For instance, pilots are now starting to use iPads or Windows Surface 2 tablets as Electronic Flight Bags (EFB), connecting this general-purpose untrusted equipment to other trusted ATM systems. SWIM changes the relations between traditional ATM stakeholders, but it also opens the door to newcomers that will offer new services. The result will hopefully maximize the efficiency of the airspace, but at the expense of an increased security risk, if this small revolution is improperly managed.

The lack of industrial experience and background in security engineering in the ATM domain and the tremendous complexity of all the trades involved in the ATM system of systems make security need elicitation a hot topic amongst ATM stakeholders today.

Thus, we applied goal modelling to the SWIM case-study, in order to gain experience, as well as to validate our goal modelling language, methodology and tool support on a real-life case.

3. Background: STS-ml

The Socio-Technical Security modelling language (STS-ml) (Dalpiaz, et al., 2011) is a goal-oriented modelling language for security requirements engineering. The language represents a socio-technical system as a set of interacting actors - representing humans, organizations, or technical systems. The notion of actor (graphically represented as a circle) is a generic one, which is specialised into two refined concepts, namely that of *role* (representing generic actors), and that of *agent* (representing actual known participants). For instance, student and professor are examples or roles, while John and Prof. Doe are examples of agents. In the air traffic management case study, Pilot, Meteorological (meteo or MET for short) service provider, Aircraft system, and SWIM access point are all actors represented as roles, there are no agents already known at design time in this case study scenario.

STS-ml supports representing actors' important *assets*, being inline with principles in computer and information security, so to capture security requirements for the protection of these valuable assets. In a socio-technical system, actors interact with one another to achieve their desired objectives, and they exchange information necessary to accomplish them. Therefore, actors' *strategic objectives* (represented by the notion of goal - a state of affairs) and their *information* (represented by the notion of *information*, together with that of *document* representing the said information) constitute one's assets. Objectives (goals) represent actors' intentions (for instance, meteorological conditions accessed); as such, we refer to them as *intentional assets*, as opposed to *informational assets*. Objectives (goals) are graphically represented as rectangles with rounded corners, while informational assets are regular rectangles (see Figure 1).

Actors achieve their main goals (*root goals*) by refining them into lower-level goals (*subgoals*) through decompositions (and/or-decompositions). For instance, Meteo service provider has the main goal of processing the MET request (*MET request processed*), which is further refined via AND-decomposition into the goals *Customer verified* and *MET info published* (see Figure 1).

Actors may possess documents, they may *need*, *modify*, or *produce* documents while achieving their goals. Information can be represented by one or more documents (through TangibleBy), and on the other hand one or more information entities can be part of some document. Information and documents can be composed of other information or documents through part of relationships. For instance, the *Met service provider* needs information represented by document *Secure MET request* to achieve goal *Customer verified*, it needs document *MET conditions data* for goal *MET info published*, while it produces document *MET info object* (Figure 1).

Actors interact with one another by delegating goals and exchanging information (transmitting documents). For example, the Pilot delegates goal *MET conditions accessed* to Aircraft system, while the latter transmits document *Secure MET request* to the SWIM access point (Figure 1).

4. Modelling and reusing threats

4.1. Expressing threats in STS-ml

We have enriched STS-ml with the primitive *event* (graphically represented as a triangle), which represents things that happen in the world. While events are in general a fundamental abstraction to represent knowledge (Kowalski & Sergot, 1986), they play a role in security requirements engineering too, for they represent occurrences that may endanger the system under design.

We decided to use events (instead of threats) to promote the generality of the modelling language. Indeed, an event per se does not indicate whether the happened thing has a positive or negative impact. We capture this relationship through the relationship *threaten*, which relates an event to another element in the model. The semantics of the relationship depends on the connected element. Particularly, when an event relates to:

- Agent, the overall purpose of the linked agent is threatened;
- Role, the role's purpose is at risk, regardless of the specific agent playing the role;

- Goal, the fulfilment of the goal is endangered;
- Document, the availability and/or integrity of the document are at risk;
- Document transmission, the transmission of the document from the sender to the receiver is threatened;
- Goal delegation, the delegation of the goal is at risk.

The identification of events and of the threaten relationships to other elements indicates reasons that motivate the introduction of security requirements for the socio-technical system

Figure 1 shows an STS-ml model for part of the ATM scenario enriched with threat modelling. The model shows how a pilot is supported by the aircraft system and SWIM in obtaining updated meteo information from a meteo service provider.

In the model, the SWIM access point maintains a registry of service providers that offer MET observations. Hence, the aircraft system does not invoke any meteo service directly, but uses SWIM as an intermediary broker. SWIM forwards the request to the correct service provider(s). SWIM access points implement standardised interfaces through which all such information requests and responses are routed. A SWIM access point can forward the MET info request, which is then handled by a meteo service provider and returns a MET info object. These exchanges of information are expressed as document transmissions.

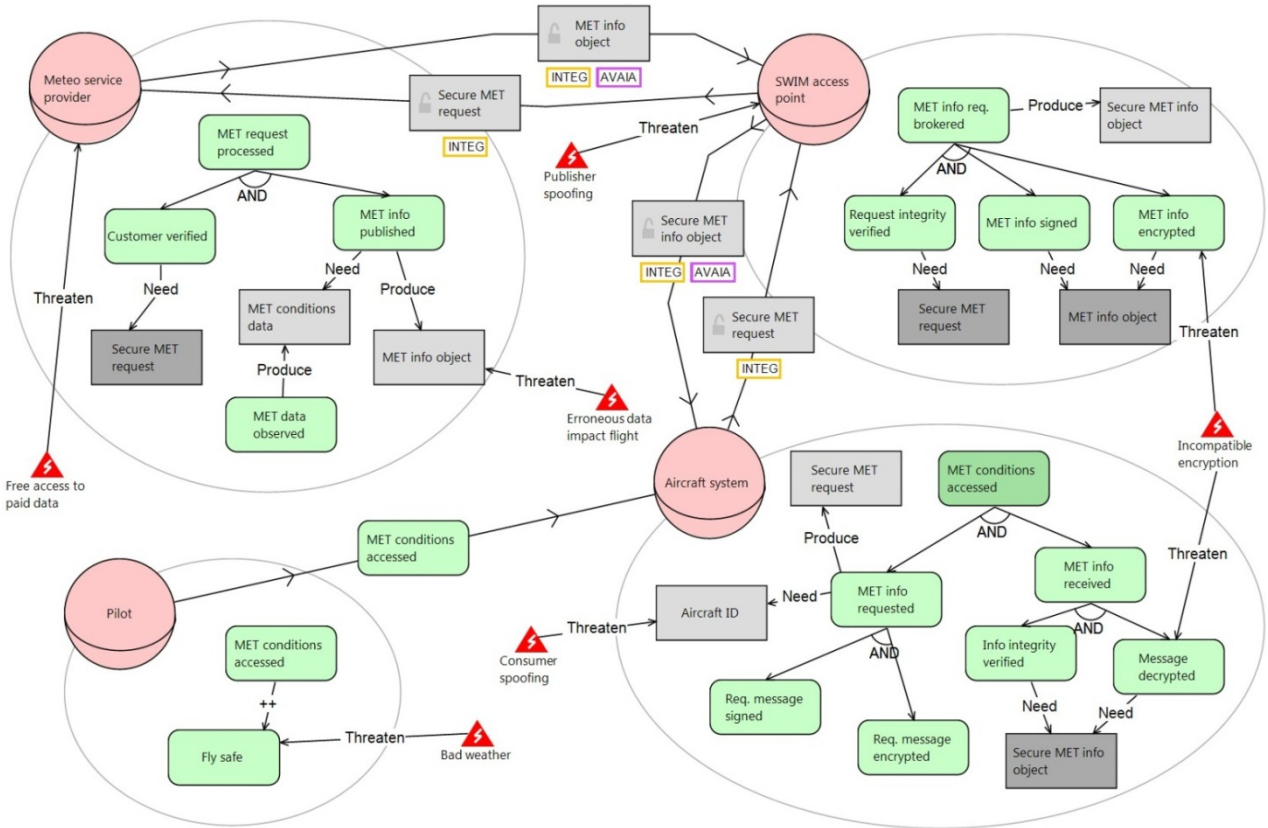

Figure 1. An STS-ml model for the ATM scenario with events and *threaten* relationships

The goals that the roles have or aim to achieve are at different levels of abstraction: while some are at a high-level (for instance, the pilot's goal to *Fly safe*), others are technical, such as the meteo service provider's goal *MET info encrypted*. Some goals are delegated to other actors. For example, goal *MET conditions accessed* is delegated from the pilot to the aircraft system.

We illustrate how we enriched the model by adding events and threaten relationships:

- *Publisher spoofing* is an event that threatens the SWIM access point actor, for SWIM users need confidence in the information brokering performed here. The threat would justify a need for SWIM access points to verify all publishers, even if this is not modelled explicitly here.
- The event *free access to paid data* is also a threat to be considered, for it threatens the meteo service provider role. From the perspective of SWIM, we do not have much knowledge of the inner workings of meteo services. We do however have requirements for the interfaces they provide. Additionally, we know through contractual obligations towards the service providers that consumption of paid services leave us with some responsibility when it comes to handling these requests securely. At this stage, however, we have not detailed a solution to this.
- Another event that we consider is *consumer spoofing*, which threatens the availability and integrity of document *aircraft ID*.

4.2. Methodology

The STS-ml language is supported by a methodology that is known as the STS modelling approach (Aniketos, 2013). This methodology, which relies upon the usage of the STS-Tool for modelling and analysing STS-ml models, outlines the recommended steps for creating the STS-ml models in a systematic way.

The methodology can be used in early security engineering, where business / operational staff can be involved without introducing the complexities of technical security solutions. We stress that this methodology is not designed to be a fully-fledged risk assessment method; it is risk agnostic, and tries to fill the identified holes in other methodologies by focusing on the security needs at the organisational level.

The methodology has a specific activity for describing the parameters and elements of the domain, where previous attack history and knowledge of adversaries and threat agents should be analysed. This is a part of the methodology's initial context study, where actors are identified and one uses the threat information to specify the circumstances that may adversely impact a role or an agent. Figure 2 shows how two threat events can be linked to an agent (*Wilbur Wright*, which is an individual and known from history as an aviation pioneer) and a role (*Pilot*).

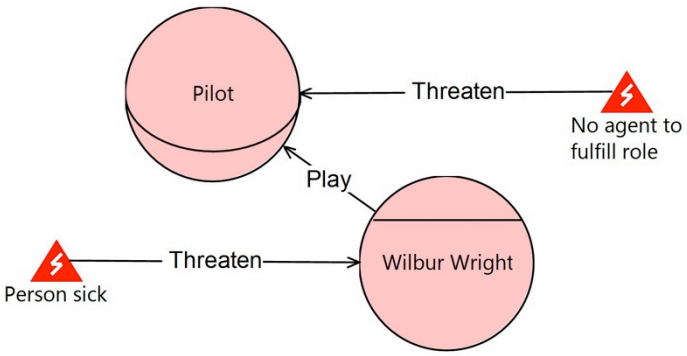

Figure 2. Threats to a role and an agent

The next main step of the methodology is a study of the assets (in STS-ml, goals and documents). Having these in place in the model allows adding events and threaten relationships that affect them. Such events typically arise from a complementing risk assessment or other more dedicated threat modelling methods. Figure 3 shows the event *denial of service* threatening the goal *MET conditions accessed* and *data tampering* threatening the document MET info object.

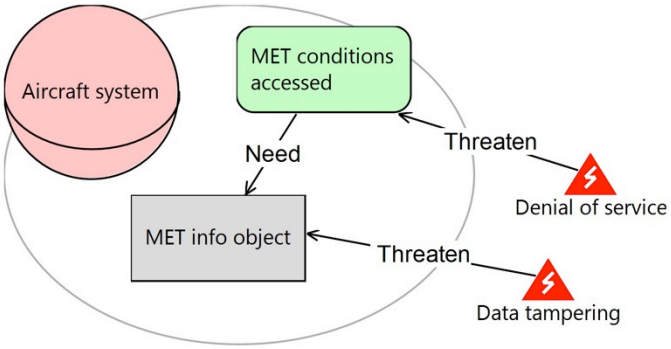

Figure 3. Threats to a goal and a document

The STS modelling approach contains a number of steps that are not directly related to threats. The interested reader can find a complete description in an open report from the Aniketos project (Aniketos, 2013).

4.3. Supporting threat reuse

As pointed out by Oladimeji et al. (2006), *"common threats and attack patterns, as well as experiences with security violations (a changing phenomenon) can all be documented in a threat catalog for later reuse"* and *"this catalog becomes useful source document during threat elicitation"*. This observation has been adopted by the works of Sindre et al. (2003) and Meland et al. (2010), where they have shown how to reuse threats between different threat models in order to improve the quality of the models.

We have extended the STS-Tool with a functionality that enables searching for threats from an existing online repository (Meland et al., 2009) and adding the selected threats directly into the model. The threat repository has a web service interface which can be used by any tool to search, import and add security information. Since the information is linked together, it is possible to find related threats (for example a set of more specific threats based on a high level threat). These are typical threats for a given vulnerability and/or suggestions for countermeasures.

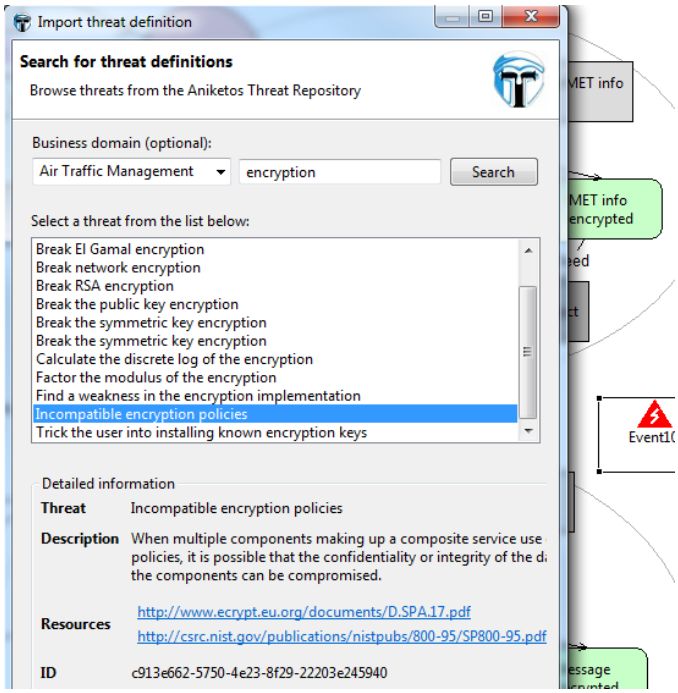

Figure 4. The "Import threat" plugin for the STS Tool

During the modelling, a search interface is available whenever the user creates an *Event* element. This interface is shown in Figure 4, where the key word *encryption* reveals a list of associated threats from the repository. In addition to specifying a text-based search filter, the STS-Tool and threat repository support domain selection, which retrieves a list of all relevant threats inside a specific domain. This way, the modeller is assisted in the discovery of potential domain-specific threats, as well as finding those already known.

The repository stores the threats with associated tags that allow mapping them to one or more specific domains. If a suitable domain specific threat cannot be found in the repository, one can import a generic threat and manually adapt it to become domain specific. Table 1 shows some examples of generic and domain specific threats for the ATM domain. In some cases there is no mapping, while in other cases several domain-specific versions of the generic threat do exist. The generic threats have been extracted from threat models uploaded to the threat repository over the years, while the domain specific ones have been direct input from people working with the ATM domain.

Table 1. Generic threats and ATM domain-specific threats

Generic threat	Domain specific threat
Theft of business information	**Theft of passenger lists**
Data tampering	**Manipulation of passenger info**
Masquerading	**Spoofing of an aircraft**
Incompatible encryption policies	**N/A**
Denial of service	**Denial of service during take-off**
Denial of service	**Denial of service en-route**
Eavesdropping	**Track position of aircraft for attack**
Hazardous environment	**Volcano ashes in the air**
Theft of physical resources	**Terrorist hijacking**
Blackmail user to reveal password	**N/A**

5. Automated analysis: threat propagation

STS-ml offers analysis capabilities (Paja et al. 2013) over the models, which include verifying the consistency and validity of the models, reasoning over security requirements, as well as analysing the impact of events threatening actors and their assets.

As described in Section 2, events may threaten actors, their supporting assets, which are represented in terms of goals or documents, as well as their interactions (goal delegations and document transmissions). As such, the principle behind threat analysis is to answer the question: *how does an event threatening any of these elements of an STS-ml model affect the rest of the model* (including assets, actors)? To answer this question, we calculate how the impact of the event is propagated over goal decompositions, goal-document relationships internal to the actor (need, modify, and produce), as well as the social relationships involving goals and documents (goal delegation, and document transmission).

Therefore, starting from the specified events (threats) we identify the impact of these threats over the rest of the model. The algorithm to calculate how the impact of threatening events is propagated in an STS-ml model, given the set of events is presented in Table 2.

The analysis starts with the known events, identifying the elements they threaten in a STS-ml model (line *1*) and propagates their impact over goal decomposition trees, documents and social relationships. Results are kept in a list (line *2*). The newly discovered elements are treated as threatened elements or objects. The analysis ends when all elements have been visited and no new elements are found (lines 3 - 5). The propagation rules are the following:

- If an event threatens an actor (line 6) then all its assets (goals and documents), as well as interactions (goal delegations and document transmissions) are threatened (line 7)
- If an event threatens a goal delegation (lines *9* and *10*), then the goal of the delegator is threatened as well (it might not be achieved since delegation might not take place)
- If an event threatens a document transmission (lines *11* and *12*), then the document of the receiver is threatened (it might not make it to the receiver)
- If an event threatens a goal (lines *13* and *17*), then the threat is propagated to:
    - the father goal if the goal is an AND-subgoal (lines *14* and *15)*;
    - the goal of the delegator if the goal is being delegated (line *16)*;
    - documents being produced by the goal if any (line *17)*.
- If an event threatens a document (lines *18* and *19*), then the threat is propagated to the documents needed or modified by the goal, as well as to the receiver's document (in case of a document transmission).

Table 2. Threat propagation algorithm

```
THREATPROPAGATION (Event e)
1.  Set<Element> tObj ← getElementsThreatenedBy(e)
2.  Set<Element> result
3.  while tObj contains elements that are not visited
4.    obj = tObj.GETNEXTUNVISITEDELEMENT()
5.    obj.SETVISITED()
6.    if obj.getType() = Actor then
7.       tObj += obj.GETGOALS() + obj.GETDOCS() + obj.GETDELEGS() + obj.GETTRANSMS()
8.    else result += {obj}
9.    if obj.getType() = Delegation then
10.      tObj += obj. GETSOURCEGOAL()
11.   if obj.getType() = Transmission then
12.      tObj += obj. GETTARGETDOCUMENT()
13.   if obj.getType() = Goal then
14.      for each parent in obj.GETGOALPARENTS()
15.         if parent.GETTYPE() = AND then tObj += {parent}
16.         tObj += obj.GETDELEGATEDFROM()
17.         tObj += obj.GETDOCUMENTSPRODUCED()
18.   if obj.getType() = Document then
19.      tObj += obj.GETGOALSMODIFYING() + obj.GETGOALSNEEDING() +
                 obj.GETTRANSMISSIONSTO()
20.   return result
```

5.1. Visualisations

The threat propagation algorithm is implemented in STS-Tool and identifies the impact of threatening events over STS-ml models while showing the results of the analysis by annotating the models in red. The details of the visualisation are, however, not discussed here for simplicity. Figure 5 shows how this works in practice for our case study model, outlining in red the affected relations, goals, documents and actors. For instance, the document *MET info object* is threatened by *Erroneous data impact flight*, as a consequence, any goal needing/using or modifying that document is indirectly threatened (as seen inside SWIM access point). Moreover, SWIM access point's goal *MET info req. brokered* is threatened since its AND-subgoal *MET info encrypted* is threatened. But, *MET info req. brokered* produces the document *Secure MET info object*, which is considered threatened as well. Finally, the Aircraft system's goal *MET conditions accessed* is threatened, as a result Pilot's same goal is threatened (being the delegator of the goals).

In a nutshell, looking at these examples we can see how the impact in one end is propagated throughout all parts of the system. We see how the goal *MET conditions accessed* of the Pilot is also affected (as expected), although not directly connected to any of the events.

The visual presentation of the results of the analysis, makes it clear in which parts of the socio-technical system particular assets are bound to be processed or retained. One should, however, be aware of how these propagated threats are realised throughout the system, as they can enter through trusted components before finally manifesting themselves and create real impact elsewhere.

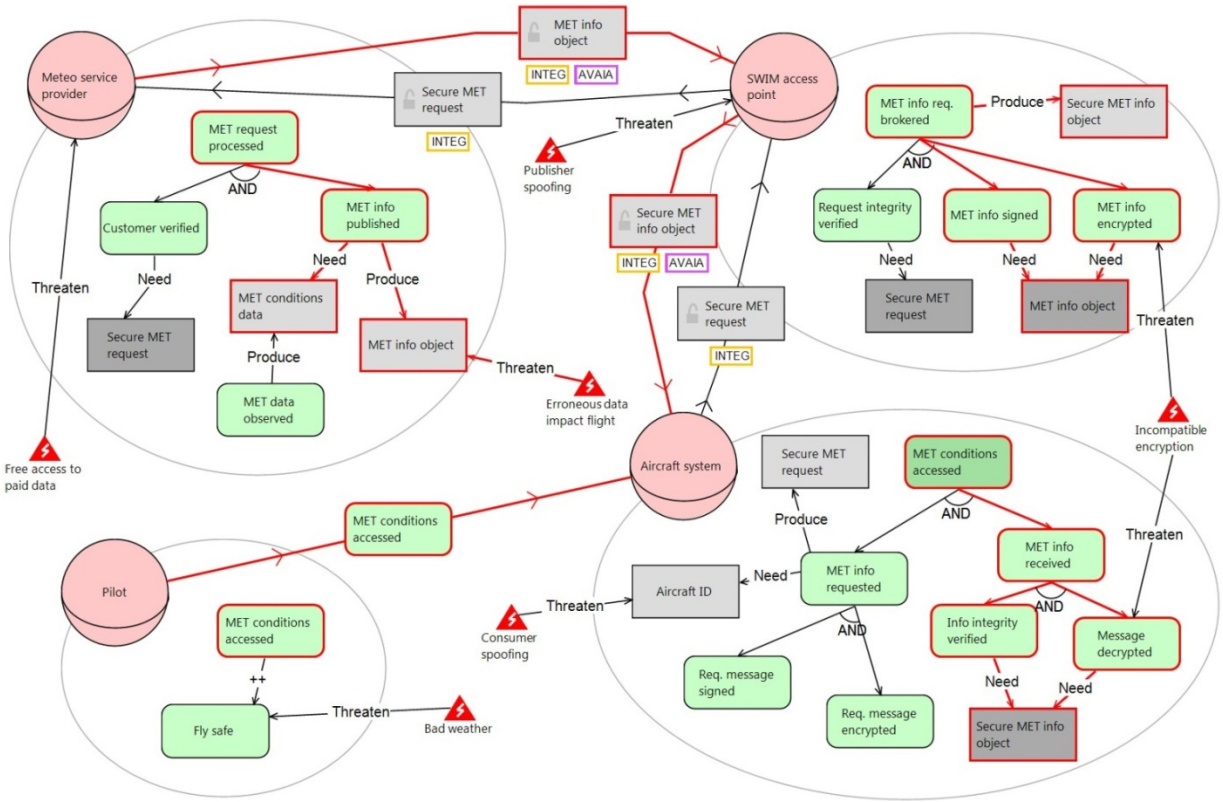

Figure 5. Threat propagation analysis as output from the generated security requirements document (extract)

5.2. Output and usefulness

The STS-Tool offers the possibility to generate a security requirements document (in PDF format) to the user. This document describes actors and their assets in the model, as well as all security requirements derived. In addition, the document contains the detailed results of the threat analysis, describing each event and listing all threatened and impacted entities (Figure 6). The methodology foresees a review of the requirements, and initial feedback suggests that automated support for validation of the requirements is desirable, in particular with respect to validity of delegations.

The security requirements document generator can look up the modelled threats in the threat repository (they were inserted in the diagram through the threat import interface). By using threat IDs stored in the model, the generator can download and include additional information in the security requirements document, including known countermeasures.

Since analysts wish to mitigate the impacts of threats that materialise in a system, not only at design time, but also at runtime, we need to maintain the threat information in later design and implementation stages. For this purpose the STS-Tool outputs a Security Requirements Specification file (SRS), which contains machine-readable XML. This file enables structured input to BPMN diagrams, in which we can define runtime behaviour of composite services. Through runtime monitoring for threats in our service components, we can listen to notifications related to these services and threats, and for instance trigger a recomposition as described by Meland and Gjære (2012).

Text	Description
Impact of event Erroneous data impact flight in the diagram	The event Erroneous data impact flight threatening MET info object, threatens also Secure MET info object, Message decrypted, MET info object, Info integrity verified, MET request processed, MET info signed, Secure MET info object, MET info published, MET conditions accessed, MET conditions accessed, MET info received, MET info encrypted, MET info req. brokered and MET conditions data.
Impact of event Consumer spoofing in the diagram	The event Consumer spoofing threatening Aircraft ID, threatens also Secure MET info object, MET request decrypted, Message decrypted, Info integrity verified, MET request processed, Secure MET info object, Secure MET request, Secure MET request, MET conditions accessed, Secure MET request, MET conditions accessed, MET info received, MET info requested, Request integrity verified and MET info req. brokered.
Impact of event Incompatible encryption in the diagram	The event Incompatible encryption threatening Message decrypted and MET info encrypted, threatens also Secure MET info object, MET conditions accessed, MET conditions accessed, Info integrity verified, MET info received, MET info req. brokered and Secure MET info object.

Figure 6. Threat propagation analysis as output from the generated security requirements document (extract)

6. Discussion

6.1. Related Work

The literature on goal-modelling in requirements engineering is comprehensive, and an overview of the major efforts before 2001 has been given by Lamsweerde (2001). The more recent developments still build upon formalisms defined in the early years, for instance the *i** framework (E. S.-K. Yu, 1996), KAOS (Dardenne, Lamsweerde, & Fickas, 1993), NFR (Chung & Leite, 2009) and Tropos (Castro, Kolp, & Mylopoulos, 2001), with goal refinements and agent responsibilities as common concepts.

As shown by Elahi et al. (2007) there are many approaches for representing various security aspects with goal oriented and agent techniques. Many of these are extensions to traditional goal modelling, for instance Secure Tropos (Mouratidis, Giorgini, Manson, & Philp, 2002) adds security features to Tropos. Among existing candidates, we have chosen STS-ml for the expressiveness, methodological, and tooling advantages that we have highlighted earlier in this paper.

Several approaches have considered threats within goal modelling. These works differ in whether they model the goals of the adversary and how they can be achieved, or how a goal is threatened by an event. Attack trees (Schneier, 1999) are an example of the former approach, where the main goal of the attacker is refined into sub-goals. An example of the latter approach is given by Oladimeji et al. (2006), who have extended the NFR framework and use negative softgoals to represent an event that may have some negative impact.

As pointed out by Quartel et al. (2009), both KAOS and *i** include notions for negative influence of the satisfaction of a goal, such as conflicts and obstacles. Building on KAOS with goals and anti-goals (as obstacles), Salehie et al. (2012) propose a framework for assembling assets, threats and security requirements. By further incorporating risk and utility nodes, the

result is a causal network which can list and rank security configurations suited for runtime adaptation.

Secure Tropos also contains a construct for threats that can be associated with a security feature element through a negative (-) contribution relationship. Work by Mahdy and Rojas (2011) describes another approach on how to integrate threat modelling as a part of the Secure Tropos methodology. Attacks are modelled as goals performed by an attacker role, and associated to the regular goals through an *attacks* relationship.

Our work is inspired by some of these approaches. However, we differ in that we provide a systematic approach that goes beyond modelling. Indeed, our approach also provides methodological guidance, automated reasoning to propagate the effect of threats, and threats reuse through an online repository. We take a lightweight perspective on threat modelling, which relies on a simple yet effective extension of STS-ml. Unlike Asnar et al. (2011), we argue that fully-fledged risk assessment should not be integrated within goal modelling, but it stands out as a separate activity.

By extending UML use cases with misuse cases, Sindre and Opdahl (2000) express how a threat (hostile goal) threatens business goals. McDermott and Fox (1991) have a similar approach with his abuse cases; however, he stresses that the long-term goal of an actor should be described over more than one abuse case, typically as a part of the textual description of a malicious actor.

6.2. Limitations and Future Directions

In the security engineering community, there are many definitions of *threat*. According to CNSSI-4009 (Committee on National Security Systems (CNSS), 2010), a threat is *"any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service"*. This same definition is now also in a recent revision of NIST SP-800-30 (National Institute of Standards and Technology (NIST), 2012). These definitions do support threats modelling in conjunction with goals, especially for they highlight the impact of threats on the missions and functions of organisations.

Other definitions point out different aspects. According to the French risk assessment method EBIOS (French Network and Information Security Agency, 2010), a threat is a typical means used by a threat source. This short and rather vague definition stresses the link between threat and threat source. Even if this fact is not stressed by the definition of threat given in the CNSSI-4009 glossary, reading NIST SP-800-30 in the text shows a similar understanding of threats: *"Threat events are caused by threat sources"*. These standards indicate that threat modelling should also encompass threat source modelling.

The follow-up question is: what is a threat source? Sticking to the definitions of CNSS and NIST, a threat source is *"the intent and method targeted at the intentional exploitation of a vulnerability, or a situation and method that may accidentally exploit a vulnerability"*. The key word here is *vulnerability*. The link between vulnerability, threat and threat source was well explained in the initial version of NIST SP-800-30 (Stoneburner, Goguen, & Feringa, 2002). This note has disappeared in revision 1 of NIST SP-800-30, but it is recalled here due

to its clarity: *"when for a given vulnerability there is no threat source that has the technical ability or motivation to exploit it, there is no threat; likewise, when there is no vulnerability present for which a given threat-source has the necessary skills, time and budget, this threat-source poses no threat"*. These standards indicate that threat modelling approach should ideally also encompass asset vulnerability modelling. However, this activity is part of classical risk assessment methods, which go beyond the scope of goal modelling.

As Zave and Jackson (1997) clearly point out, requirements engineering (and, thus, goal modelling) are concerned with the analysis of the problem space. On the other hand, vulnerability modelling is part of the solution space, for it concerns weaknesses that affect parts of the to-be system. This observation, along with the standards that define vulnerability modelling as essential part of threat modelling, indicate that threat modelling is incompatible with goal modelling.

Nevertheless, we argue that this is a limitation of existing methods, which can be circumvented. Stoneburner et al. (2002) observes that there is no threat when there is no vulnerability. However, threat sources exist irrespective of the existence of vulnerabilities, and vulnerabilities are often not known in advance. The existence of a threat source can be modelled as part of the problem space, for instance in conjunction with goal modelling. The threats in our model are threat sources in the sense that they denote potential vulnerabilities. An interesting future direction is to enrich our threat sources with information such as motivations and skills, to answer questions such as "*who are my potential adversaries, what's their motivation, and what are their goals? How much inside information do they have?* (Myagmar, Lee, & Yurcik, 2005). It would also be possible to add countermeasures to the model, as previously shown by Rojas and Mahdy (2011) for Secure Tropos and for attack trees by Kordy et al. (2011).

A different research path (more challenging and in need of experimental work) concerns deviations from intended goals. It is often the case that unforeseen circumstances (events) make the system deviate from its intended goals. For example, a top-level goal of an aircraft pilot could very well be *fly safely*. During the 9/11 event, it suddenly occurred that the pilot's top-level goal has been abandoned and goal *crash aircraft* has been pursued. Many measures had been taken to ensure that the *fly safely* goal could be achieved, assuming someone, i.e. the pilot, cared to achieve that goal. Threat modelling in the problem space might be interesting to capture and analyse a change in the problem definition, i.e., a change of goal, and prevent this deviation from occurring.

When it comes to reuse of threats, the online repository we have been exploiting contains many generic threats for software development, but only a few domain-specific threats. In order to improve the quality of the modelling process, there should be a development for reuse as well, which considers the domains for which reuse would pay off (creating reusable artefacts is more expensive (Tracz, 1988)). Research questions related to this have previously been raised for misuse cases by Sindre et al. (2003). There is an investment need in the creation of both more generic and domain specific threats for the repository. A substantial set of goal models that include threats would enable deriving generic threats. Creating specific threats will always require great domain knowledge and experience that is dependent on collaborations with industrial partners. The payback would of course be access to these shared resources, and tool support that could analyse trends and automatically suggest relevant threats based on heuristics.

Another important component of our approach is the tool-supported threat propagation analysis. While it reduces the need to manually investigate threat impacts, it relies upon model correctness in order to function properly. It is yet unexplored how well the tool is able to reveal gaps in the security architecture, apart from showing which goal model elements are affected.

7. Conclusion

Threat modelling is a very important activity in security requirements, for it enables preventing hazardous circumstances early in the development process. There are several different ways for including threats into goal modelling, and the choice depends on the purpose that one wants to achieve. In our experience, which is reflected in our approach, threats provide a foundation and justification for the security requirements that originate from goals.

We argue that threat modelling is not a replacement for risk assessment methods. Risk assessment is typically concerned with describing and analysing the vulnerabilities that would enable for an attacker to exploit the system. Our approach stands at a higher level of abstraction: when representing goals at the organisational level, we are still in the problem space, and system vulnerabilities are still unknown, for they reside in the solution space. Since threat sources exist irrespective of vulnerabilities, we do model these in STS-ml; the key advantage is that, thinking of threats before entering the solution space enables thinking outside the box and identifying possible threats that would not be identified when considering the system's execution flow.

Our experience has shown that integrating threat-related concepts with goal modelling increases the complexity to the language and the modelling process, but with appropriate tool support the benefits make this choice worth-while. In our approach, the tool enables propagating and visualising the impact of threats on actors, goals, and documents. This enables creating better requirements in the first place and thereby better solution designs and implementations. We have also shown how to achieve reuse of generic and domain-specific threats, and how the STS-Tool has been extended to integrate with an online threat repository.

8. Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos). It is an extension to the already published paper by Meland et al. (2013).

References

Aniketos. (2013). Deliverable 1.4 Final version of the socio-technical security modelling language tool http://www.aniketos.eu/content/deliverables.

Asnar, Y., Giorgini, P., & Mylopoulos, J. (2011). Goal-driven risk assessment in requirements engineering. *Requir. Eng., 16*(2), 101-116.

Castro, J., Kolp, M., & Mylopoulos, J. (2001). A Requirements-Driven Development Methodology, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering* (pp. 108–123). London, UK: Springer-Verlag.

Chung, L., & Leite, J. C. P. (2009). On Non-Functional Requirements in Software Engineering. In T. B. Alexander, K. C. Vinay, G. Paolo & S. Y. Eric (Eds.), *Conceptual Modeling: Foundations and Applications* (pp. 363-379): Springer-Verlag.

Committee on National Security Systems (CNSS). (2010). National, Information Assurance (IA) Glossary, Instruction No. 4009 http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf.

Dalpiaz, F., Giorgini, P., & Mylopoulos, J. (2013). Adaptive socio-technical systems: a requirements-based approach. *Requirements Engineering, 18*(1), 1-24.

Dalpiaz, F., Paja, E., & Giorgini, P. (2011, 8-8 Sept. 2011). *Security requirements engineering via commitments.* Paper presented at the Socio-Technical Aspects in Security and Trust (STAST), 2011 1st Workshop on.

Dardenne, A., Lamsweerde, A. v., & Fickas, S. (1993). Goal-directed Requirements Acquisition, *Science of computer programming* (pp. 3–50).

Elahi, G., Yu, E., & Sandhu, R. (2007). *A Goal Oriented Approach for Modeling and Analyzing Security TradeOffs.* University of Toronto, Canada.

Eurocontrol (Producer). (2013, April 15th) System Wide Information Management (SWIM). retrieved from http://www.eurocontrol.int/services/system-wide-information-management-swim

French Network and Information Security Agency (Producer). (2010) EBIOS - Expression des Besoins et Identification des Objectifs de Sécurité retrieved from http://www.ssi.gouv.fr/site_article45.html

Giorgini, P., Massacci, F., Mylopoulos, J., & Zannone, N. (2005, 29 Aug.-2 Sept. 2005). *Modeling security requirements through ownership, permission and delegation.* Paper presented at the Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on.

Kordy, B., Mauw, S., Radomirović, S. v., & Schweitzer, P. (2011). *Foundations of attack-defense trees.* Paper presented at the Proceedings of the 7th International conference on Formal aspects of security and trust, Berlin, Heidelberg.

Kowalski, R., & Sergot, M. (1986). A logic-based calculus of events. *New Gen. Comput., 4*(1), 67-95.

Lamsweerde, A. v. (2004). *Elaborating Security Requirements by Construction of Intentional Anti-Models*. Paper presented at the Proceedings of the 26th International Conference on Software Engineering.

Liu, L., Yu, E., & Mylopoulos, J. (2003, 8-12 Sept. 2003). *Security and privacy requirements analysis within a social setting.* Paper presented at the Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International.

McDermott, J., & Fox, C. (1999). *Using abuse case models for security requirements analysis.* Paper presented at the 15th Annual Computer Security Applications Conference, 1999 (ACSAC'99).

Meland, P. H., Ardi, S., Jensen, J., Rios, E., Sanchez, T., Shahmehri, N., et al. (2009). *An Architectural Foundation for Security Model Sharing and Reuse.* Paper presented at the International Conference on Availability, Reliability and Security, 2009 (ARES '09).

Meland, P. H., & Gjære, E. A. (2012). *Representing Threats in BPMN 2.0.* Paper presented at the Seventh International Conference on Availability, Reliability and Security (ARES), 2012.

Meland, P. H., Gjære, E. A., & Paul, S. (2013, 2-6 Sept. 2013). *The Use and Usefulness of Threats in Goal-Oriented Modelling.* Paper presented at the Availability, Reliability and Security (ARES), 2013 Eighth International Conference on.

Meland, P. H., Tøndel, I. A., & Jensen, J. (2010). *Idea: reusability of threat models; two approaches with an experimental evaluation.* Paper presented at the Proceedings of the Second international conference on Engineering Secure Software and Systems, Berlin, Heidelberg.

Moody, D. (2009). The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Trans. Softw. Eng., 35*(6), 756-779.

Mouratidis, H., & Giorgini, P. (2007). Secure Tropos: a Security-Oriented Extension of the Tropos Methodology. *International Journal of Software Engineering and Knowledge Engineering, 17*(2), 285-309.

Mouratidis, H., Giorgini, P., Manson, G., & Philp, I. (2002). *A Natural Extension of Tropos Methodology for Modelling Security.* Paper presented at the Proceedings of the agent oriented methodologies workshop (OOPSLA 2002), Seattle - USA.

Myagmar, S., Lee, A. J., & Yurcik, W. (2005). Threat Modeling as a Basis for Security Requirements, *In Symposium on Requirements Engineering for Information Security (SREIS)*.

National Institute of Standards and Technology (Producer). (2012) Information security, Guide for conducting risk assessments, Special Publication (SP) 800-30, revision 1. retrieved from http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf

Oladimeji, E., Supakkul, S., & Chung, L. (2006). *Security threat Modeling and Analysis: A goal-oriented approach*.

Paja, E., Dalpiaz, F., & Giorgini, P. (2013). *Managing Security Requirements Conflicts in Socio-Technical Systems.* Paper presented at the 32nd International Conference on Conceptual Modeling (ER 2013).

Quartel, D., Engelsman, W., Jonkers, H., & van Sinderen, M. (2009). A Goal-Oriented Requirements Modelling Language for Enterprise Architecture, *Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International* (pp. 3-13).

Rojas, D. M., & Mahdy, A. M. (2011). Integrating threat modeling in secure agent-oriented software development. *Int. J. Softw. Eng, 2*(3), 23–36.

Salehie, M., Pasquale, L., Omoronyia, I., Ali, R., & Nuseibeh, B. (2012). *Requirements-driven adaptive security: Protecting variable assets at runtime.* Paper presented at the Requirements Engineering Conference (RE), 2012 20th IEEE International.

Schneier, B. (1999). Attack Trees: Modeling Security Threats. *Dr. Dobb's Journal*.

Shostack, A. (2008). *Experiences Threat Modeling at Microsoft.* Paper presented at the Modeling Security Workshop, in Association with MODELS '08.

Sindre, G., Firesmith, D. G., & Opdahl, A. L. (2003). *A Reuse-Based Approach to Determining Security Requirements.* Paper presented at the In Proc. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03).

Sindre, G., & Opdahl, A. L. (2000). *Eliciting security requirements by misuse cases.* Paper presented at the 37th International Conference on Technology of Object-Oriented Languages and Systems, 2000. TOOLS-Pacific 2000.

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., et al. (2012). Large-scale complex IT systems. *Commun. ACM, 55*(7), 71-77.

Stoneburner, G., Goguen, A. Y., & Feringa, A. (2002). SP 800-30. Risk Management Guide for Information Technology Systems. Gaithersburg, MD, United States: National Institute of Standards & Technology.

Tracz, W. (1988). Software reuse myths. *SIGSOFT Softw. Eng. Notes, 13*(1), 17–21 http://doi.acm.org/10.1145/43857.43859.

Trösterer, S., Beck, E., Dalpiaz, F., Paja, E., Giorgini, P., & Tscheligi, M. (2012). Formative User-Centered Evaluation of Security Modeling: Results from a Case Study. *IJSSE, 3*(1), 1-19.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour, *Requirements Engineering. Proceedings. Fifth IEEE International Symposium on* (pp. 249-262).

Yu, E., & Mylopoulos, J. (1998). *Why Goal-Oriented Requirements Engineering.* Paper presented at the Fourth International Workshop on Requirements Engineering: Foundation for Software Quality.

Yu, E. S.-K. (1996). *Modelling strategic relationships for process reengineering.* University of Toronto, Toronto, Ont., Canada, Canada.

Zave, P., & Jackson, M. (1997). Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol., 6*(1), 1-30.