Managing Security Requirements in Practice: A Case Study
-- Extended Abstract --

Vidya Lakshminarayanan, WenQian Liu*, Charles L Chen, Dewayne E Perry

Empirical Software Engineering Lab (ESEL)
Electrical and Computer Engineering
The University of Texas at Austin, Austin TX
{vidya,clchen,perry}@ece.utexas.edu

*Software Engineering
Department of Computer Science
University of Toronto, Canada
wl@cs.toronto.edu

ABSTRACT

While security has long been a significant issue in military systems, the spread of the internet has stimulated a growing interest in, and increasing demand for, secure systems. As with any domain, there are specific issues in the security domain that must be understood to successfully engineer the needed secure software systems. We present data collected as part of a study of requirements and architecture that is relevant to managing security requirements. Our findings show that security requirements are significantly different from other requirements such as performance and reliability; that highly secure systems need to be well-engineered software systems; and that software engineering must look deeper into the security domain to build reliable secure systems.

Categories and Subject Descriptors

D.2 [Software Engineering]: Requirements, Architecture

General Terms

Security

Keywords

Security, security requirements, managing requirements, security architects, security architecture, case studies

1 INTRODUCTION

Security has long been a major issue in military and defense systems. Making sure that only the right people get access to information, that plans do not land in the wrong hands, and that communication channels are not compromised are among the top priorities for national defense. More recently, the internet boom has exacerbated the problem. By connecting everyone with everyone else, the internet has greatly enhanced our ability to exchange information, but it has also opened more doors for attackers. With the growing concern over malicious attacks compromising data integrity and privacy, security in software systems has become an increasingly important topic and led to increased software engineering research [1,4,5,6].

In our empirical research on the topic of how to go from requirements to architecture [2,3], we conducted a series of interviews with practicing architects. Our case study involves a series of carefully designed semi-structured interviews. The data collected from the interviews helped us analyze how software architects (i) manage requirements and in particular handle non-functional requirements, (ii) view software architecture, (iii) transform the requirements into architecture, and (iv) view software evolution. In our companion paper, we give a detailed description of the design of our case study [7].

Among the architects interviewed, three are primarily involved in building secure systems and managing security requirements. We have analyzed these security related interviews in depth, distilling critical comments and perceptions about how security requirements are managed in practice. In section 2, we discuss our case study in three parts: (i) delineating key aspects in security, (ii) characterizing security architects, and (iii) discussing the critical issues in managing security requirements. In section 3, we summarize our findings, draw our conclusions, relate our conclusions to current work, and indicate areas of future research.

2 CASE STUDY

In this section, we provide our insights into security issues based on the data collected from three interviews with software architects who are involved in security. We present selected interview data that reflect how these architects view security and manage such requirements in

practice. Two of our subjects consider themselves security architects and one manages security requirements. To preserve the anonymity of our subjects, we will use *A, B* and *C* instead of their real names in all the quotes.

We present the data and our analysis in three parts. First, we describe the key aspects in security. Next, we illustrate some of the critical characteristics of security architects, particularly the skills required for managing security requirements effectively in addressing these key issues. Last, we present how our subjects manage issues in security requirements in practice.

Attributions are provided in the form of quotations or summaries whenever the subject has a remark on the topic. The lack of such indicates that no relevant discussion was found in the interview data.

2.1 Key Aspects in Security

We present the following key issues with respect to security: what the problem domain is; how mature or stable it is; what difficulties and obstacles are imposed on the domain.

Problem Domain

Subject *A* suggested that security issues have typically surfaced in three areas of software engineering: communication, operating system, and cryptography. Furthermore, he distinguished three types of security problems. The first type is authentication and protection of discrete resources. He suggested that solutions to these problems are well-established.

"[A] fairly large collection of security problems, including authenticating principles and protecting discrete resources against unauthorized access and protecting content of communications at least for some time interval, [has] well-established solutions that are often fairly easy to apply and work reasonably well." [Subject A]

The second type is the protection of confidentiality in the presence of inference. He believes that finding solutions for these problems is difficult.

"The canonical example of a difficult security problem is protecting confidentiality of information in a relational database. … The reason that relational database security is hard is because a relational database is basically an engine for developing lots and lots of aliases for the same information. When you get a new reference that you have not seen before, it is difficult to tell whether that reference applies to information that you have already protected in some way, therefore, it is difficult to apply the correct policy. … Inference is known to be a hard problem, [so is] preventing unauthorized inference from a string of queries to a database." [Subject A]

The third type is intellectual property related issues for which he believes the solutions are impossible.

"There is a third set of problems for which we keep trying to solve, but for which solutions are actually probably impossible. I include Digital Rights Management among this set of problems. Not everyone agrees with me." [Subject A]

Subject *B* also mentioned that encryption algorithms have well-established and easy-to-apply solutions, which is in support of the first type that *A* mentioned. In addition, he pointed out that privacy in transit (i.e. protecting content away from the source) is a difficult issue.

"[An] FBI personnel going to CIA … should be able to read a document, but the moment [he] leaves CIA, [he] should not read the document. [This] idea ... is privacy in transit. Because usually … we enforce privacy close to the source, but once [this is taken] away from you, you can't really enforce it [anymore]. We do spend a lot of time thinking about [such] problem[s]." [Subject B]

Maturity and Stability

According to subject *A*, the security domain as a whole is immature and unstable.

"You read the newspaper. There is no possibility I am going to be out of job anytime soon. By my definition, it means that it's not a mature domain."[Subject A]

"It is an unstable domain specifically because the architectural artifacts that we have designed for security were designed with a set of assumptions in mind which are no longer true of real computer systems; so the architecture is not well matched to the real world."[Subject A]

While subject *B* did not remark on the domain in general, he indicated that many aspects within security are immature. On the other hand, he also pointed out there are some mature aspects within security that have well-established solutions.

"Security is a huge topic, and there [are] a lot of things which are immature in security. Like Federation [Identity Management] is very, very immature." [Subject B]

"There's both maturity and immaturity within the space. … Things like the encryption algorithm, pretty mature; you know how to do it. [For] user name protection, figure out what level of protection you need, [and] what are you protecting against, you have …different protocols [to solve them and] each one has pros and cons."[Subject B]

Some Difficult Problems in Security Engineering

Composition is particularly difficult in engineering secure systems because emergent properties can cause serious problems when putting two or more components together.

"It is unfortunate that lots of security problems do not compose in a mathematical sense, i.e. if X has security property 1, and if Y has security property 1 then X+Y does not [necessarily] have security property 1."[Subject A]

Subject *A* suggested that a framework approach does not work well for this problem since it is often underspecified which is undesirable in building secure systems. He indicated that it is theoretically possible to have a precisely defined set of frameworks that are specific

enough for security but abstract enough for general applications; however, practically it does not exist.

Awareness and understanding of security issues is low among people in general. According to subject *A*, this leads to difficulties in getting people to appreciate the feasibility (as in the case of Digital Rights Management) and justification of a various levels of security.

"Cost justifying security has always been a nightmare…so you can come up with a perfectly good architecture and everybody says, 'Ok, let's build one tenth of it.' " [Subject A]

"There's large number of products on the market with unsuccessful security." [Subject A]

However, some signs show that people are starting to pay more attention to security.

"[It can] make or break the deal." [Subject C]

"It's getting easier to justify it now…on the basis of reputation damage… [It's] easier to get security projects justified after your website has been defaced." [Subject A]

In order to address these key issues, we are interested in finding out what skills are required of software architects, especially those who deal with security issues. In the next section, we will present our subjects' opinions on the critical characteristics of architects.

2.2 Characteristics of Architects

During the interviews, our subjects discussed at length the characteristics required of architects in designing software systems, especially highly secure systems. We will begin with the general characteristics of architects and narrow it down to discussions of the particular skills required of security architects.

Bridging the gap between business and technology is the key in architecting software systems.

"I think bridging the gap is the key. If you talk to the business folks they always want to talk business talk, they never want to talk technology.... It is how you take [the] requirement and translate it to something that you can see, identify, and bring out some level of commonality. … That to me is the key thing for architects." [Subject B]

"The flip side… [is] applying [this] to a set of technologies. That comes through education. So I think an architect should be well versed in technologies which are available."[Subject B]

Breadth is another required characteristic.

"Breadth is very important, not [being] just focused only on security, but being able to know the other aspects of software engineering, whether it is performance, hardware, application [or] whatever it is." [Subject B]

"Generally, security people are generalists rather than specialists. They have [to] understand a lot about different parts of the system and how they work, [and] enough about each [part] of the system so that they can figure out [where] vulnerabilities [can surface]." [Subject A]

Architects should generally be involved in collecting requirements.

"I think that we do [collect requirements] often. I am not saying … that should be done all the time or not be done [at] all, but I think it's good. I strongly feel about architects going and talking to customers." [Subject B]

"It's good to conceptualize what people want. …That's why it is important to go see customers. If you don't see customers, you don't get that and I can't tell you enough to get [the] experience." [Subject B]

Architects need to have strong technical, people, leadership, and communication skills. They should be able to wear different hats at different times.

"Good technical background is the key. Good people, good leadership skills are very, very important. Because you are leading a team, … a set of people to believe what you think is right, … you got to be able to convince. … If you are a dictator, that's bad. You got to be a team player, … have some level of leadership skills and be able to listen. If you don't listen, then you will be going to your tunnel vision and do what you think is right, as opposed to what is required for the job." [Subject B]

"You work with developers who talk only [in] development language; you talk with customers … [in] their own language; you talk with the marketing people [in] their own language; you talk to the [executives] who [use] a different language … so you got to be able to balance out all of those in a good fashion." [Subject B]

"[An architect needs to be a] politician, diplomat, nursery attendant, business liaison. You have to be ... not a believer... [but a] benevolent dictator. [You] have to be technically savvy, but more so, sound. I don't think you need to know the latest version of the latest spec … [but] good sound design principles and … learn [quickly]." [Subject C]

To the question on whether security architects are born or trained, subject *A* replied, "*Generally, … majority of the security people are born, but then after that they have to be trained*." However, subject *B* suggested they can be trained and need not to be born with such skills.

"I think anybody can do anything in life if you work hard. That's my fundamental belief. Having said that…Yes, some people just don't get it. … Developers tend to be very focused. … [One] characteristic of an architect is breadth. … Typically when we try to grow somebody, the biggest problem we face is they are very focused in what they know, and they are not easy to learn the rest of the concepts." [Subject B]

To be effective in managing security requirements architects must be able to adopt the mentality of the attackers.

"The most important qualification to be a security architect is [being] able to think like the bad guys. … If you do not have an element of ... malice, [or] least an appreciation of the beauty of malice … you are just going to fail." [Subject A]

Subject *A* described three attitudes people may have in response to a new attack. Only one is appropriate for a security architect.

"[The first says] 'that is really annoying, I can't get my job done'. They are fine they are probably not dangerous, you could use them to test things or something."

"[The next says] 'oh that is really neat! I wonder how he did that'. Those will likely be good security people."

"[The last says] 'Well you know, nobody should be allowed to do that'. They have to be kept far away from security. They totally have the wrong attitude, they don't get the problem, [and] they will never be able to think that way." [Subject A]

2.3 Issues in Managing Security Requirements

So far, we have discussed key aspects in security and presented critical characteristics of architects. We now present our data on managing security requirements from three perspectives: establishing security requirements, prioritizing security requirements and architecting security requirements.

Establishing Security Requirements

Security has a fundamental difference from all the other requirements, such as reliability, safety and performance. In the latter, we usually expect to have random component failures and accidents. In the former however, failures are often caused intentionally by capable and motivated adversaries. Therefore, it is important to capture the malicious intentions, motivations, and capabilities of attackers in the security domain. Threat models are used for these considerations.

"In security the primary problem is the existence of a capable and motivated adversary who wants the system to fail. This property makes security architecture different from other disciplines." [Subject A]

"Security architecture is fundamentally based on the idea of threat models. You have to start off with the model of the threats you are trying to defend against; if the threat model incorporates the possibility of physical attacks, then you have to pay attention to physical attack. … In fact, threat analyses do include an element of characterizing adversaries in terms of capability, motivation, and desired outcomes." [Subject A]

However, subject *A* commented that security problems cannot be solved by an ontology-based approach. He suggested that a way to approach it is through generalization over past attacks and experiences.

"As soon as [one] puts together the ontology, by definition it defines everything that is there and therefore everything else is unthinkable. Unthinkable stuff [is] really bad." [Subject A]

"The way security people learn to think about things… is by studying past failures. … You look at the collection of successful attacks on past systems [and] make sure that none of those work on the current system. … Then if you really hit a dead end and want to break the system, you take somebody who doesn't have any assumptions. … [They will be] able to enter into the whole thing because [they will not try to think like the designers]. … Sometimes it is very important to be able to do that when you are designing security systems." [Subject A]

Sometimes the requirements and the problems are not presented in the right form. In such cases, it is necessary to discover the shape of the problem and identify the form of the requirements in order to proceed.

"If … we already know what the problem was and the customer is putting a twist to it we try to shift the customer or the requirements to the right direction by saying 'maybe you should think this way' or 'maybe you can do the same thing by an alternate way'. Because customers are set in their ways and they don't want to change, so they want what they have been doing. … Education helps [at] certain times. … People seem to have a narrow focus and sometimes you have to broaden them." [Subject B]

"What [is the] business problem [that] you are trying to solve? Don't come to me with 'we have to upload this spreadsheet'. [Tell me] what are you trying to solve; what are you trying to do. And when [we] don't do that [we] just end up in a rat hole." [Subject C]

There can be situations where it is not possible to accommodate all the requirements at the same time. In such cases, we do the best we can by assessing the pros and cons.

"When a requirement is outright impossible, we say that's impossible. … And sometimes we are told to do it anyway. Digital Rights Management is a perfect example. I believe it is demonstrably the case that you cannot do Digital Rights Management to meet a set of requirements that people in the entertainment industry want. I just don't think it is feasible. Nevertheless, we enable our systems for DRM and build DRM mechanisms anyway. Because people say they want them. … It filters out a number of dumb attackers. The smart attackers get in and copy things anyway." [Subject A]

"There is not a luxury to do everything that we want to do or everything that is ideal. You have to go with the requirements, go with the political nature, the business requirements, the funding aspect. … So you do pros and cons and decide what is the best..." [Subject B]

On top of the intellectual aspects, the physical aspects of security also play an important role.

"[We] really cannot afford not to pay attention to physical aspects of things. It's sort of like designing the pressure vessel of a submarine; it only has to leak in one place for you to have trouble. And if that is in the physical infrastructure then that's just as bad a problem as if you have screwed up some conceptual thing. So [we] have to pay attention to every aspect of how you might attack a system." [Subject A]

"There are physical [aspects we need to pay attention to]. How is our data safe? … What happens if a tornado hits? How secure is that data in any kind of disaster? ... That's at the macro level. Then you get into the application, and they are very sensitive about different parts of the [application]. … You could define security a contributor that cuts across every type of object in the system." [Subject C]

Prioritizing Security Requirements

Having established a set of requirements, two situations often arise: (i) there are conflicting requirements, and (ii) the cost of building a system that satisfies all the requirements is too high. Hence, there is a need to prioritize the requirements to establish which are key and which are subordinate. Deciding on how to prioritize

requirements is usually done through negotiations. For functional requirements, choice can be made through prioritization of feature sets. For non-functional requirements, it is often possible to achieve a balance without seriously compromising any of them.

"You get a good feel for the weight of the requirement.…You can generally tell, by the discussion, what's important to them especially if you push back on something. [If] it's really important to them, you'll start getting the messages, the body language, [that they are] not comfortable with that."[Subject C]

However, security levels are defined with respect to specific goals; they are either achieved or not. Thus, security requirements have to take precedence without giving any concessions. Aside from setting the level of security that is acceptable, there is not much about security requirements that can be adjusted.

"The attack succeeds or it fails. So [security] is a difficult property to subject to engineering tradeoffs. … [But] you can decide in advance that the system has to impose some specified work factor on the adversary and have that as a design goal." [Subject A]

"You can't ... really continuously tune your level of security. And this of course pisses off all the other designers in the organization because they are all sitting around saying, 'Well, you know we can tradeoff a few clock cycles here for a better user interface here or something like that' and the security guy is just sitting in the room and everybody else says, 'So what do you have to offer?' And the security guy says, 'Nothing, you have to do it my way.' " [Subject A]

However, sometimes other factors can trump security as in the case of legal issues.

"We had a certain product and it failed because of the legal implications with that product. … There is a legal ramification of issuing a certificate. That means if I am issuing a certificate to [you] then I am accountable for it if [you do] any fraud with that certificate. … There is a liability [issue] associated with that. So we spend tons of money on the product, and it was failed." [Subject B]

Architecting Security Requirements

All our subjects commonly expressed the opinion that it is important to consider other requirements in support of security requirements while building secure systems.

We observed that there is a slight disagreement on how the subjects categorize these supporting requirements[1]. For example, some of them categorize performance to be functional while others categorize it to be non-functional. Whether functional or non-functional, it is agreed that a set of supporting requirements is needed in building successful secure systems. These include performance, scalability, interoperability, availability, manageability, and maintainability.

"Typically in security the functional requirements mostly have to do with interoperability mechanism and with manageability. So it's a functional requirement that my VPN client has to be able to talk this bizarre protocol that is spoken by my mutant VPN server." [Subject A]

"A system administrator [needs] to [be able to] update the access of everybody in department X by running a script over night. [This implies] there's got to be an API level interface for the security management system and it's got to have certain kinds of authentication and authorization functions so that we can run it safely." [Subject A]

"Performance is frequently a functional requirement; you are not allowed to slow down." [Subject A]

"Kevin Mitnik should not be able to talk to the operator into giving up the password. This … is a genuine requirement. The Russian mafia should not be able to break the cryptography, but it is okay if the Russian government can." [Subject A]

In building secure systems, both functional and non-functional requirements play a critical role in all phases.

"The functional aspects are something like the core aspects of the product … Non-functional are performance and scalability... we try to give functional requirements more importance because that's what is seen […and] marketed. But non-functional requirements are worked in with that because what's the point in releasing a product if it doesn't scale beyond 100 users, or … doesn't perform. So it goes down [to] all phases. Whether it is architecture [or] design, you combine those two things at all points of time and work towards a cohesive architecture." [Subject B]

"I do not feel there is that big a difference between non-functional and functional. It is just a requirement and somehow you have got to accommodate it." [Subject C]

Security requirements are defined relative to specific goals capturing known vulnerabilities. These goals must be accounted for in designing the system structure. Given that security is embedded in the system structure, it cannot be altered easily.

"You can decide in advance that the system has to impose some specified work factor on the adversary and have that as a design goal. [Once] you have that as a design goal, you have to hit that mark or do better. You can't really continuously tune your level of security." [Subject A]

"We have a great deal of flexibility to adjust and replace mechanisms. [For example] we can add stronger cryptography on the wire protocols. What it's much less easy to do is to change the basic structure of the system in a way that has an impact on security. Sometimes we end up having to do that, and it's a lot of work." [Subject A]

Security goals must be designed with a farseeing vision; the lack of that will lead to failures.

"A specific example of this … there was a cellular phone protocol that was in the process of being standardized …. This protocol depends for its security on the assumption that bad guys can't put up a tower … that's [definitely] not a good

[1] We believe that the disagreement is due to the necessary reification from non-functional requirements to functional structures.

assumption …. That protocol does not exist in that form anymore as it turns out it's not that hard to put up something that looks to a cell phone handset as if it were a tower." [Subject A]

Even though security is an integral part of the system, we must be able to address the issues of modularity and externalizability. Security needs to be configurable (to achieve different security levels) and its implementation must be replaceable depending on the context without breaking the system.

"[It is ideal that] in the production environment with full security, various layers of security can be turned off. If you turn pieces off, the system still functions. [Also,] you can layer more and more security if you want." [Subject C]

"The example I can give is J2EE architecture … the first release of J2EE did not cover much of security. It was totally enclosed within the architecture, meaning it was not open … so [every vendor] did security in their own way because it was not specified by the standard … we realized it was no good. So rather than implementing something ad hoc for the moment, we said … it would be nice to externalize the security so that anybody can plug into it." [Subject B]

Subject *B* pointed out that the aforementioned decision helped their company in two ways: (i) they were able to integrate with several other products and (ii) when the standards came out they only had to replace their API with the standard API, unlike the other vendors who were struggling to dissociate security from their application server.

"If you design with some pretty standard rules up front, it makes things a lot easier moving on." [Subject C]

Security requirements often restrict the choices of other requirements. There is an obvious tradeoff between security and performance because extra operations are required in more secure systems.

"Generally speaking, the more security you need, the more isolation you need to have. As the system, gets more and more secure, the interface narrows …, and there are fewer ways to talk to it. … [It] tends to be the case that security trades off against performance. … As you harden the interfaces of the components and isolate it more and more, you make it more difficult to cross the boundary between the non-secure portion of the system and the part of the system that enforces security." [Subject A]

In short, we have seen how architects establish, prioritize and architect security requirements in practice.

3 DISCUSSION

We first summarize our findings, give our conclusions, relate them to current work, and finally suggest issues for further research.

Here is a summary of the key findings of our case study.

- Key aspects
  - immature and unstable in general
  - well-known solutions available for some aspects
  - composition is difficult
- Characteristics of architects
  - ability to bridge the gap between business and technology
  - breadth in knowledge
  - possession of soft skills
  - attacker's mentality and appreciation of challenges
- Establishing security requirements
  - threat models are effective for discovering and eliciting security requirements
  - ontology-based approaches are not appropriate
  - discovering the shape of the problem is critical
  - difficulty in assessing the criticality and feasibility of security requirements can lead to poorly managed expectations of security
  - security goals must be defined with respect to known vulnerabilities
  - the physical and intellectual aspects of security both play a critical role in designing a highly secure system
- Prioritizing security requirements
  - once the level of security is determined, it takes precedence over other requirements
  - non-technical issues can trump security
- Architecting security requirements
  - other requirements are critical in building secure systems
  - system goals must incorporate the intended level of security and the structure must account for these goals
  - security is an integral part of the system and is not easily alterable
  - farseeing vision is required for anticipating potential attacks
  - modularity and externalizability help to achieve high configurability in security

From the data collected, we observe that security is a critical domain that requires highly specialized treatment. Building secure systems and managing security requirements effectively depends on established software engineering principles and practices. Well-engineered systems provide the foundation for achieving security goals. Security is no different from many other domain specific areas, such as telephony or performance tuning, which exhibit unique characteristics yet still rely heavily on software engineering techniques, methods and principles.

Software systems today are expected to provide a high level of security. To achieve this, security must be included in the design goals right from the beginning. It is an integral part of the system and is not tunable or

imposable after the fact. Software engineers must be aware of the unique aspects of the requirements in this domain and use appropriate methods. The evidence in our case study supports this position.

The literature also has supporting evidence for our position. In his keynote speech, Wolf pointed out that *"Security engineering is a technical field dependant upon methods, tools, and models for requirement analysis, design analysis and implementation analysis"* and concluded that security engineering really is just good software engineering [4]. The software engineering research community is starting to take notice of the security domain and its unique domain properties. As a result, new techniques, methods and technologies are emerging. One noticeable contribution is the anti-goal models introduced by van Lamsweerde et al. in capturing malicious obstacles: "*In the context of security engineering, standard obstacle analysis appears too limited for handling malicious obstacles"* [5,6]. Nevertheless, as Wolf pointed out that, "*Software threat analysis is a young art"* and existing models do not adequately support the analysis needed by security specialists [4]. There is much work to be done in the security domain.

In conclusion, the authors believe that the convergence between the security domain and software engineering is inevitable. Having said that, we think security specialists should employ established software engineering principles and practices to their advantage, and software engineers must recognize the unique aspects of the security domain and continue to provide and to apply appropriate methods to attain a higher level of software security. Thus, we claim that current software engineering literature is congruent with the findings of our study of managing security requirements in practice.

Several issues have surfaced in our case study which require further research.

- How should architects be involved in requirements elicitation and negotiation systematically?
- How can we identify frequently occurring problem shapes and requirement forms that are adequate for secure systems?
- What are the specific modeling tools/methods needed for capturing security requirements?
- What are the evaluation techniques required to assess security levels in architecture?
- Are there any conflicts within security requirements and how are they resolved?

In addition to our general study of requirements and architecture, we will continue our research on these threads. In the cases where more evidence is required, we will either follow up with the current subjects or conduct new interviews.

ACKNOWLEDGEMENTS

We thank all of our anonymous interviewees for their participation and contribution. This research was supported in part by NSF CISE Grant CCR-0306613 and IBM CAS Fellowship.

4 REFERENCES

[1] R. Anderson. *"Security Engineering -- A Guide to Building Dependable Distributed Systems"*. John Wiley & Sons, Inc. 2001.

[2] M. Brandozzi and D.E. Perry. "Transforming Goal Oriented Requirements Specifications into Architectural Prescriptions". In *Proc. of workshop on Software Requirements to Architectures (STRAW '01)*, ICSE 2001.

[3] M. Brandozzi and D.E. Perry. "From Goal-Oriented Requirements to Architectural Prescriptions: The Preskiptor Process". In *Proc. of workshop on Software Requirements to Architectures (STRAW '03)*, ICSE 2003.

[4] A. L. Wolf. "Is Security Engineering Really Just Good Software Engineering?" In *Proc. of the Foundations of Software Engineering*, Keynote speech. 2004.

[5] A. van Lamsweerde. "Elaborating Security Requirements by Construction of Intentional Anti-Models". In *Proc. ICSE'04: 26th International Conference on Software Engineering*, 2004.

[6] A. van Lamsweerde, A., Brohez, S., De Landtsheer, R., & Janssens, D. "From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering". In *Proc. Requirements for High Assurance Systems Workshop* (RHAS'03), 2003.

[7] W. Liu, C. L. Chen, V.Lakshminarayanan and D. E. Perry. "A Design for Evidence-based Software Architecture Research". Submitted for publication, 2005.