INTEGRATING SECURITY AND USABILITY INTO THE REQUIREMENTS AND DESIGN PROCESS

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

Cecilia Mascolo
Department of
Computer Science
University College London
UK – London WC1E 6BT
c.mascolo@cs.ucl.ac.uk

M. Angela Sasse
Department of
Computer Science
University College London
UK – London WC1E 6BT
a.sasse@cs.ucl.ac.uk

Abstract: Security is a complex and important non-functional requirement of software systems. According to Ross Anderson, "Many systems fail because their designers protect the wrong things, or protect the right things in the wrong way" [Anderson, 2001]. Surveys [Department of Trade and Industry, 2004] also show that security incidents in industry are rising, which highlights the difficulty of designing good security. Some recent approaches have targeted security from the technological perspective, others from the human computer interaction angle, offering better user interfaces for improved usability of security mechanisms. However usability issues also extend beyond the user interface, and should be considered during system requirements and design. In this paper we describe AEGIS, a methodology for the development of secure and usable systems. AEGIS defines a development process and a UML meta-model of the definition and the reasoning over the system's assets. AEGIS has been applied to case studies in the area of Grid computing and we report on one of these.

1. Introduction

Developing a secure software system is a complex and time-consuming process that seeks to accommodate frequently competing factors, such as functionality, scalability, simplicity, time-to-market, etc. Software engineering research has recently focused on improving the modelling abilities in terms of non-functional requirements such as stability [Jazayeri, 2002], performance [Denaro et al, 2004], fault tolerance [Guerra et al, 2003] and security [Jürjens, 2003]. In this paper we will focus in particular on security issues.

Techniques to incorporate security issues in software design have already been developed [Jürjens, 2003, Schneier, 2003], however there is one important aspect of the design of complex secure systems which has always been neglected: current research in the field of Human Computer Interactions in Security (HCISec) illustrates that security mechanisms that do not work in practice are not effective [Adams et al, 1999, Ka-Ping, 2002, McDermott et al, 1999, Whitten et al, 1999]. Most of the research in HCISec focuses on providing better user interfaces (UIs) [Ka-Ping, 2002, Whitten et al, 1999], but it is clear that usability problems with secure systems are more than just UIs and need application of HCI factors and design methodology.

Secure systems do not exist in a vacuum; they exist for the purpose of providing people with services and as such cannot operate without the involvement of people. In security, the focus tends to be on people who want to abuse the system (attackers). This is to the detriment of the regular users, who play an important part in protecting it. Any secure system is a socio-technical system [Brostoff et al, 2001], and the requirements analysis and design process must take this into account.

Most countermeasures require the involvement of people at some level. Users can have vastly different levels of experience, knowledge and expertise. Designing a system that appropriately accommodates these differing levels of aptitude and training is vital if the countermeasures are to be dependable. Therefore the design and the development of a secure software system require the inclusion of yet another important requirement: usability.

This additional requirement introduces another layer of complexity in the development process.

To date, no attempt in this direction has been made. In [Ka-Ping, 2002], ten guidelines for usable security are recommended, and in [Brostoff et al, 2001] a security design approach based on a safety-critical methods is proposed, but neither of these actually provides practical assistance or guidance for developers. At best, they are given a means of analysing a system, not building it.

In [Flechais et al, 2003] we presented a novel method for building secure and usable software. In this paper we build and expand on that work and define the semantics of the steps of the development. We present the secure software development process AEGIS, which provides important tools for developing secure and usable systems. As part of AEGIS we define a UML meta-model identifying assets, the context of operation and supporting the modelling of security requirements. This clear semantics allows the developers and the users to formulate constraints and needs for the security aspects of the system in a simple but clear way, as shown in the case study reporting on our application of the work on Grid systems.

This paper is organised as follows: Section 2 will present an overview of the AEGIS process, while Section 3 will describe the UML meta-model for the AEGIS asset diagrams. In Section 4 we will present a case study in which AEGIS has been applied. Section 5 discusses some of the results and compares AEGIS to other work and Section 6 summarises the paper and indicates possible future directions for this work.

2. Overview of the AEGIS process

Appropriate and Effective Guidance for Information Security (AEGIS) is a software development process for secure and usable systems.

AEGIS is formulated to be a lightweight process that can fit into any software development process (for example in [Flechais et al, 2003], AEGIS was integrated into an incremental development process [Boehm, 1988]). The activity diagram in Figure 1 describes the core steps of AEGIS, which consist of identifying and securing the correct participants, getting them to model the system's assets in context using our semantics defined through the UML [Object Management Group, 2003b] meta object facility [Object Management Group, 2003a], assign a value on these assets, conduct a risk analysis and, finally, design the countermeasures that address the risks in a cost effective way. Usability needs are addressed thanks to the participation of users in the security design, together with active consideration being given to the user context during both security requirements modelling and countermeasure design.

In the next few sections we will give details of the different stages of the process.

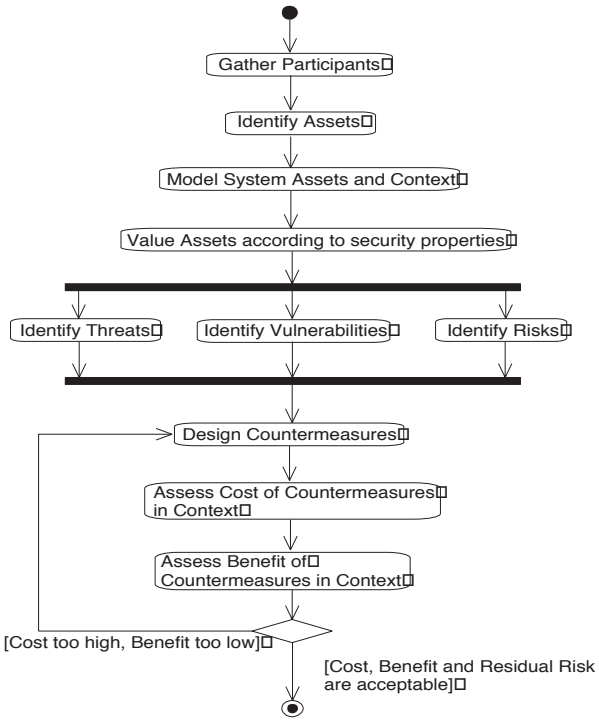

Figure 1. AEGIS activity diagram

2.1. Gather participants

AEGIS is designed as a participative design methodology [James, 1996, Mumford, 1983, Straub et al, 1998]. That is to say that different stakeholders in the system are actively involved in the process of eliciting security requirements and deciding on security countermeasures. This is because the system stakeholders have the most pertinent domain knowledge. Therefore any decision taken by these stakeholders should take

into account their different needs – and specifically the need for usability.

The first step is therefore to identify and secure the commitment of the stakeholders who will participate in that design. There are four main types of roles that can be differentiated (although an individual can play more than one role):

- Decision makers. They consist of project management, owners (customers commissioning the system), and anybody else given a decision-making role in the development of the system.
- Developers. They are the technical aspect of the design team, responsible for the capture and analysis of the system requirements down to the design and implementation. These include programmers, designers, security experts, interface designers, etc.
- Users. They are the people that the system should be designed to work with, and as such are a major source of system requirements.
- Facilitators. They are the people who run the AEGIS process, document the meetings and serve as mediators in general.

Despite being traditionally regarded simply as a technical problem, the design of security is instead a socio-technical issue [Adams et al, 1999] – i.e. designing and building security must involve both a technical and a social undertaking. Developers are the best equipped to handle the technical aspects of security; however the social aspects of security are generally the province of the owners and higher management, who have the authority to institute, encourage and enforce policies.

This is why it is essential to ensure the participation of these groups of people: the decision makers – who are better suited to dealing with the enforcement of the social requirements of security, the developers – who are necessary for the technical implementation of security, the users – who are the ultimate source of usability requirements of the system, and the facilitators – who ensure the smooth running of the design process.

An important additional aspect during this phase is to determine a single individual who will have leadership for the security of the project. The responsibility associated with this role is to document decision-making, citing the arguments and reasons for the decision, and to provide a final say in any disagreement that may occur during the process.

2.2. Identify and model assets in context

This step focuses on identifying the valuable parts of the system, and modelling them in the context in which they operate. Inspired by the HCI design methodology *Contextual Design* [Beyer et al, 1998], this contextual information is essential as a means of recreating the operating environment during the analysis and design phases, thereby ensuring that user needs are taken into account.

AEGIS defines three major categories of assets – operatives, hardware and data.

- Operative
  - User
  - Administrator
  - Developer
- Hardware
  - Network link
  - Computer
    - Processing node
    - Storage
    - …
- Data
  - Application
  - Information

Operatives identify the people interacting with the system, whether users, developers or administrators. These assets tend to be the most overlooked of all, because they are not generally perceived as being a part of the system, but a customer of the system.

Hardware assets are the physical entities in the system which need to be protected. From a security standpoint, an attacker who has physical access to the hardware is much more likely to succeed than one who does not. Identifying the presence and role of the physical assets in the system is therefore vital in the overall design of the security countermeasures.

Data assets are subdivided into applications and information. Applications refer to the software that runs on various hardware assets. These will generally correspond to the more traditional architecture for the system (which concerns itself with the software architecture).

- Security Measure
  - Operative
  - Hardware
  - Data

A final category is that of security measures, which consists of any of the previous assets. Security measures can take the form of operatives (e.g. guards, administrators checking system logs, or users having secure passwords), hardware (e.g. dedicated optical networks that are much more resistant to interception, dedicated encryption hardware or random number generators) or data (e.g. a security policy governing the backup of information, an encryption algorithm, a firewall application or an encryption key).

Once these assets have been identified, they can be modelled using the semantics defined in section 3.1. (see also Figure 5). Of critical importance in this phase is the recreation of the context in which the system operates. Context refers both to the physical and cultural environments that surround and affect the system. This step is crucial in providing the participants with the system knowledge necessary for designing a usable system later on.

The next step consists of assigning a value to these assets according to various security properties as described in the following section.

2.3. Value assets according to security

In order to elicit security requirements from the participants, it is necessary to first explain and agree on the meaning of security properties. The three most common security properties are defined as follows [Gollman, 1999]:

- Confidentiality: property of security that concerns unauthorised disclosure of information
- Integrity: property of security that concerns unauthorised modification of information
- Availability: property of security that concerns unauthorised withholding of information

Security requirements elicitation is achieved, for each of the assets, by having the participants judge the importance of the asset in terms of the security properties defined above

We recommend using a qualitative rating system based on natural language which gives flexibility in the rating of the assets, however it is equally possible to adopt quantitative rating systems. The qualitative approach allows participants to use their own words to define how important assets are, and by ranking the results hierarchically, a breakdown of the most important security properties for the assets can be identified.

Experience has shown that scenarios are very useful in making participants both understand what the security property means, but also, how important it is in relation to the asset. These various scenarios should be documented, possibly in the form of abuse cases [McDermott et al, 1999] – UML use cases of an attacker and the actions taken to conduct an attack.

For more information on the semantics of modelling the security requirements for the assets, see Section 3. The following step should consist of a risk analysis to identify threats, vulnerabilities and risks to the system.

2.4. Risk analysis

Risk analysis attempts to determine which threats and risks the system faces in order to feed into the design of security countermeasures that are appropriate to the threats and are cost-effective to the risks. Knowledge of existing and past threats and vulnerabilities is essential, as is the presence of expert security knowledge in order to interpret and adapt this information to the situation at hand.

This step is not about dictating the security needs of the system, it is about painting the picture of the threats, vulnerabilities and risks to the system in its current form. The designers, the developers and the decision makers should then

use this information to decide if, what, and how much security should be built into the design.

A risk analysis generally goes through a three-step process of:

- Identifying Threats – Threats are the potential sources of attacks to the system. Things that characterise threats include the attacker, their motive, their target, their resources and their risk-aversion.
- Identifying Vulnerabilities – Vulnerabilities are areas of the system that are amenable to exploitation. This is where security advisories, security scanners, good knowledge of the technologies being used and information about past attacks become important.
- Identifying Risks – Risk is the likelihood of an attack successfully exploiting one or a sequence of vulnerabilities in order to compromise an asset. This information is generally best acquired from security experts who have the knowledge and experience necessary to assess these risks.

2.5. Security design

This next phase is an iterative process of designing potential security measures and assessing their respective costs and benefits in the context in which they will be used. The aim of this is to reduce all the risks identified previously to an acceptable level, whilst ensuring the reliability of the system by providing usable mechanisms, education, incentives and disciplinary measures to motivate people in the system to behave in the expected manner.

The design of the security should be driven by the risks identified previously, with attention being paid to those which are deemed to be most important. During this design, the cost of the implementation, deployment, operation and maintenance of the resulting secure system should be assessed. For usability purposes, the user cost of applying the measures in the context of operation should also be assessed and factored into the decision making. These costs should be evaluated against the benefits of the security measures and their ability to mitigate risks.

In the next section we describe the UML meta-model that is used to give semantics to the assets definition specified by the participants.

3. Asset model semantics:

3.1. Asset model

The semantics for an asset model are described using the UML Meta Object Facility [Object Management Group, 2003a] as can be seen in Figure 2. The meta-model defines the semantics for models of assets which can then be built by the participants. The reasons for choosing UML for this kind of modelling are obvious: UML is a well-understood notation among developers, it is widely supported and easy to extend (through the meta object facility). The simplicity of the extension means that non-experts can also easily understand and use this as a starting point if provided with a basic introduction.

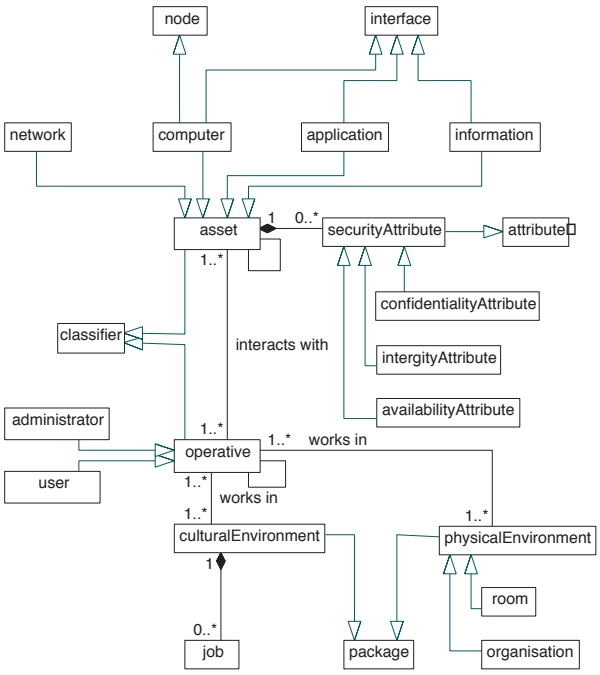

Figure 2. Asset model semantics

Four new objects are defined in the meta-model in Figure 2:

- `asset`
- `operative`

- physicalEnviroment
- culturalEnvironment

Although we have previously identified operatives as being assets, the AEGIS meta-model refines the semantics with a distinction between operative and asset. This is to accommodate the differences of interaction that operatives have with other assets and other operatives. Bearing in mind the similarity of an operative to a UML user, the same look was chosen to depict an operative, as seen in Figure 3.

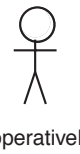

Figure 3. Diagram for operative

In addition to the assets and operatives, the physical and cultural context surrounding both the assets and operatives can also be depicted through the physicalEnvironment and culturalEnvironment components. Asset and operative both extend the UML MOF classifier and should therefore be modelled as such (see figure 5 for an example). physicalEnvironment and culturalEnvironment both extend the core UML package and should therefore be modelled as packages. These two components can thus contain assets and operatives and serve to represent the boundaries of both physical environments (such as rooms) and cultural environments (such as security culture). For example, a system administrator operative and a secretary operative sharing the same room should be apparent.

3.2. Security requirement modelling

In order to document the security requirements, Asset is a classifier (Figure 4) that contains securityAttributes, which have been defined as confidentialityAttribute, integrityAttribute and availabilityAttribute. These attributes should be used to record the value of each asset.

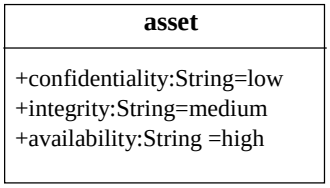

Figure 4. Diagram for asset

Additional attributes can also be defined, such as for example a non-repudiation attribute or dependability attribute, depending on the needs of the system. These securityAttributes extend the core attribute element of the UML MOF and can thus be depicted in a similar manner. Thus, an asset can be drawn as shown in Figure 4.

4. Case study:

AEGIS has been applied in a number of case studies involving Grid projects. These projects are developing the technology and expertise necessary to deploy large scale distributed networks for the purpose of providing access to very large sets of data (where normal distribution channels are inadequate), access to computational facilities (such as the spare CPU cycles on home computers, or specific supercomputers for instance) or a combination of both. Since the field of Grid computing is relatively recent, the security requirements and difficulties in building suitable countermeasures are not very well understood. Furthermore, the vast storage capability, processing power and bandwidth that makes Grids so useful also makes them prime targets for malicious attacks.

In order to test and validate the usefulness of AEGIS as well as its ease of use (for facilitators and developers), it has been taught to a group of graduate software engineering students who then applied the process to four different grid projects. We report here on one of the sessions, which involved a biological simulation grid project.

4.1. Learning AEGIS

The principles and processes of AEGIS were taught to a group of six students in a two-hour session. The basic principles of AEGIS were

explained through a series of slides, as well as a sample asset model. Once this introduction completed, the students were given a manual and access to two members of the biological simulation grid project.

The two members of the project were able to represent both a developer and a system user point of view and their participation was secured for two hours and thirty minutes (although the user unfortunately had to leave after one hour). The students were given the tasks of identifying the security needs of the project and conducting as much of a security analysis as possible within the timeframe.

Initial questions from the students were focussed on understanding what the project was about and how the basic architecture functioned. The grid project was described as providing a group of universities a means of centralising access to different *"simulations of molecules of biological importance"*.

It was quickly identified that the project was expected to provide a secure environment for these different universities to operate in. In addition to this, there were long-term plans to expand the system to private sector pharmaceutical companies. In light of this, the need to provide a secure environment was further reinforced by the fact that the private sector had very high confidentiality requirements, to the extent that *"it's really hard to convince them [pharmaceutical companies] to share their data with anybody – to even go outside of their own building"*.

Although academic use and provision of simulation data was free of confidentiality constraints, the private sector had very high requirements of confidentiality for their own simulations. A long-term aim of the project was therefore to provide their software to these private companies so they could federate their own databases in a compatible format and query the union of the private and public databases, but not allow queries from outside access to their own simulations.

The importance of the biological simulation data, also called trajectories, was further identified through the following questions *"would you place a high cost on producing the data? The manpower and equipment involved…"*, *"would the R&D of other pharmaceutical companies be interested [in this data]?"*. Both answers were positive and showed that producing the data was expensive and the simulations could be very valuable to third parties.

4.2. Modelling the system

At this point, the students tended to want to focus on the specific security needs of the confidentiality of the database of simulations. After a quick reminder that the analysis should start by identifying all the assets and various stakeholder operatives, the students started building an asset model later formalised as can be seen in Figure 5.

The grid project representatives initially had trouble understanding what was required of them *"what do you mean by asset?"*, although the students were quickly able to explain and lead them through an analysis. The modelling process consisted of one student drawing the asset model onto a whiteboard while the group of students as a whole asked detailed questions about the architecture that would inform the diagram.

For example, having identified that the project was geared to providing users with simulation data, the then users asked questions about how this data was served to the user, what kind of server it resided on, where the server was housed, and so on. This in turn led to the identification of a number of other assets, such as the application server which provided authentication, authorisation and accountability services. Apart from the basic user, operatives were identified by asking leading questions, such as *"who is in charge of maintaining the system"*, *"who has access to the server room"*, *or "who supplies the information in the database"*. These operatives were then modelled as shown in figure 5. The interactions between the operatives and the assets were identified throughout the process of building the model, such as the administrative task of maintaining

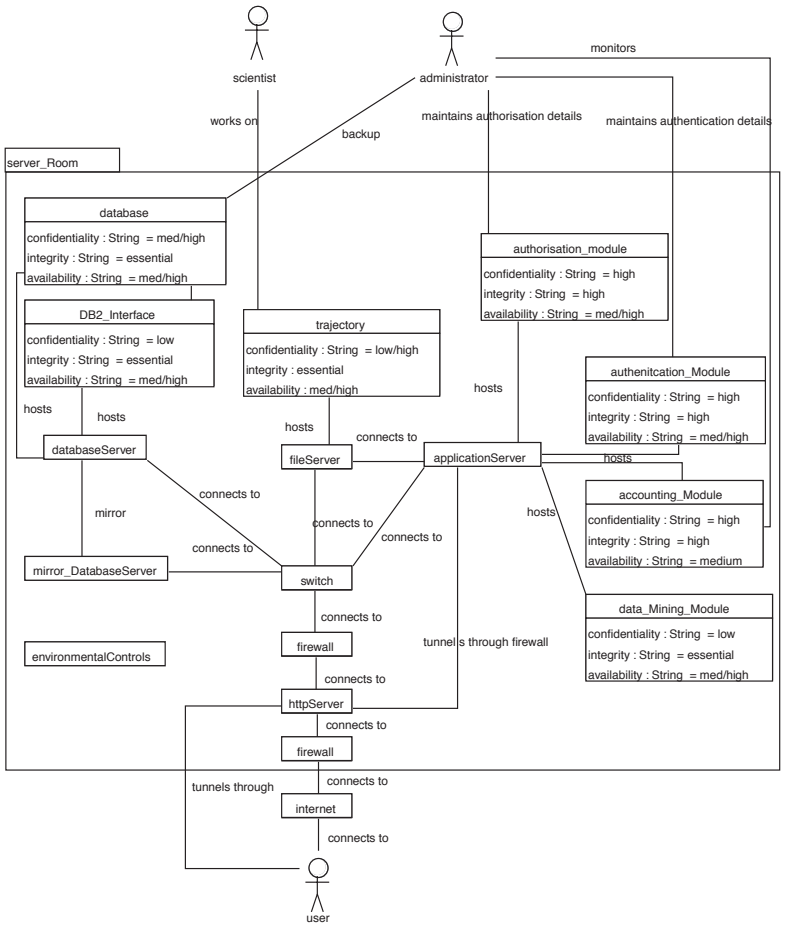

Figure 4. Case Study Asset Model

the authentication mechanism which became apparent when the students identified the existence of that asset.

One finding that corroborates other AEGIS case studies was that many of the administrative duties in the system, such as backup, patching, maintenance of the authentication mechanism (in this case based on SSL digital certificates, and a username and password combination for users who don't have certificates), and maintenance of the authorisation mechanism (role-based access control) were not initially apparent. Identifying these required detailed and probing questions, for example when the representatives mentioned that the system was backed up (*"who backs the system up? Is there a policy for when and what to backup?"*), or that digital certificates were used to authenticate users (*"How do users get a certificate?"*, *"Who do they apply to for access to the system?"*). What is interesting is that simply establishing that an administrator has to monitor, backup, and maintain the system – with little to no supervision or help – throws up a number of questions with regards to both the scalability of the system (can the tasks expected of the administrator be extended to cover one or two orders of magnitude more users?) and the effectiveness of the current system security (which in the absence of training, audit and documented policies is wholly dependent on the competence of the administrator – not on the technical countermeasures).

4.3. Identifying security requirements

Building on the discussion at the start of the analysis, students tried to get the representative to rate the confidentiality requirement of the trajectory files (simulation data). *"How important is it for you to be able to keep this secret?"* to which the answer was *"we have no need for confidentiality… At the moment."* When questioned further, *"from an academic user point of view, using your own words, how would you rate, how important would confidentiality be? Would it be low, unimportant, high, essential…"* The answer was that the requirement for confidentiality was low, however from the *pharmaceutical company's* point of view, the requirement for confidentiality was deemed to be medium to high in some cases. However since the system did not currently involve pharmaceutical companies, the current requirement was originally judged to be low.

From a requirements point of view, capturing this information is important. On the one hand, the system *as it is* does not require that particular type of security, on the other, the system *as*

envisioned in a future development may have a high requirement for this kind of security. Furthermore this also illustrates the need to identify and represent as many stakeholders in the system as possible to identify potentially conflicting viewpoints. As can be seen in Figure 5, the confidentiality requirement for trajectory was therefore rated as "low/high", which highlights this basic conflict.

Identifying the security requirements of other assets did not highlight any further conflicts, and it was quickly established that the integrity of the trajectory files was the most important security requirement of the system. This was because the whole purpose of the system was to provide accurate data. Thanks to the dependencies of the integrity of the trajectory files, the database was also judged to have an equal need for integrity. The availability of the data was not judged to be very important in the short-term, but in a future commercial application this would be more important. This was further justified by the fact that the project had already designed server mirrors in the architecture of the system.

Another series of assets that proved to be of interest were the authentication, authorisation and accounting modules. Although they were originally expected to resolve security issues in the system, the identification of high integrity and confidentiality requirements show that they also raise security issues. It is in this type of area that AEGIS shows its main difference from other security approaches, because it takes the point of view that every asset, including the security measures, has specific needs. These effectively highlight the need for good usability, training, incentives and enforcement for security measures that require the involvement of an operative. Without those, the requirements of the security measures may not be met.

5. Discussion and related work:

As shown in the case study, a number of issues were identified through AEGIS. First of these was that the security roles of operatives in a system are frequently overlooked, and technical security mechanisms are generally assumed to solve a security problem. By identifying security requirements on security mechanisms, these new security problems are highlighted. Modelling the tasks that operatives must perform in the system also helps to highlight some of these problems.

Although the case study shows that some confusion existed at the beginning of the process, the participants quickly adopted the method, and in a relatively short period of time new issues and requirements were identified. This also highlights the importance of the role of facilitator in the process of AEGIS where it is easy to get sidetracked on a particular area, whilst ignoring a multitude of other problems.

A final point concerns the resolution of conflicting requirements. Different stakeholders in the system will have different points of view about what is important to them. This is typical of any reasonably large engineering project and establishes the need for making decisions based on conflicting data. With regards to security, it is very important to understand the need for the cost-benefit analysis of any security decision. The differences between the short and long-term security needs in the system do not necessarily have to cause serious difficulties. It is cheaper to compromise on a short-term implementation than it is to compromise on the long-term design. Any security mechanisms that have been designed but not implemented will be cheaper to implement at a later date than in a system where it is necessary to overhaul its original design.

6. Summary and future work:

AEGIS has been presented as a development process that provides both usability and security. Through the definition of MOF-compliant semantics, we have described an asset model notation, capable of documenting security requirements. By modelling the context in which the system operates and the interactions of the operatives and the assets of the system, this notation also allows the documentation of usability needs. Finally, we have presented a case study in which AEGIS was taught and applied to a grid project. The case study highlighted that AEGIS is easy to learn, provides a clear means of documenting security

requirements and is useful in identifying the role and importance of operatives in the system.

Future work may include identifying issues concerning the resolution of conflicts in security requirements gathering, incorporating decision making support, improving tools support for AEGIS and also integrating AEGIS into Model Driven Architectures [Object Management Group, 2004].

7. References:

Adams, A. & Sasse, M. A. *Users Are Not The Enemy*. Communications of the ACM 1999. Vol. 42, No. 12 December

Anderson, R. *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2001. Wiley.

Beyer, H. & Holtzblatt, K. *Contextual Design : Defining Customer-Centered Systems*. 1998. Morgan Kaufmann Publishers, Inc.

Blakley, B., McDermott, E., & Geer, D. *Information Security is Information Risk Management.* New Security Paradigms Workshop 2001. pp 97-104.

Boehm, B. W. *A spiral model of software development and Enhancement*. IEEE Computer 1988. 21(5) , pp 61-72.

Brostoff, S. & Sasse, M. A. *Safe and Sound: a safety-critical approach to security design.* New Security Paradigms Workshop 2001.

Denaro, G., Polini, A., & Emmerich, W. *Performance Testing of Distributed Component Architectures*. S.Beydeda and V.Gruhn (eds), Building Quality into COTS Components - Testing and Debugging. 2004. Springer. http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/BeyadaGruhn/PerformanceTesting.pdf

Department of Trade and Industry. *Information Security Breaches Survey*. 2004. http://www.security-survey.gov.uk/

Flechais, I., Sasse, M. A., & Hailes, S. M. *Bringing Security Home: A process for developing secure and usable systems.* New Security Paradigms Workshop 2003.

Gollman, D. *Computer Security*. 1999. Wiley.

Guerra, P. A. d. C., Rubira, C., & de Lemos, R. *A Fault-Tolerant Software Architecture for Component-Based Systems*. Lecture Notes in Computer Science 2003. 2677 , pp 129-149. Springer.

James, H. L. *Managing Information Systems Security: a Soft Aproach*. Proceedings of the Information Systems Conference of New Zealand 1996. IEEE Society Press, Los Alamitos, CA.

Jazayeri, M. *On Architectural Stability and Evolution*. Reliable Software Technologies - Ada-Europe 2002, Vienna, Austria, June 17-21 2002. http://www.infosys.tuwien.ac.at/Staff/mj/papers/archstab.pdf

Jürjens, J. *UMLsec: Extending UML for Secure Systems Development*. LNCS 2003.

Ka-Ping, Y. *User Interaction Design for Secure Systems.* 2002. http://zesty.ca/sid

McDermott, J. & Fox, C. *Using Abuse Cases for Security Requirements Analysis*. Proceedings of the 15th Annual Computer Security Applications Conference 1999.

Mumford, E. *Designing Human Systems - The Ethics Method*. 1983. **http://www.enid.u-net.com/C1book1.htm**

Object Management Group. *Meta Object Facility (MOF) Specification.* Technical Report 2003a.

Object Management Group. *Unified Modeling Language version 1.5*. 2003b. http://www.omg.org/technology/documents/formal/uml.htm

Object Management Group. *Model Driven Architecture*. 2004. http://www.omg.org/mda/

Schneier, B. *Beyond Fear Thinking Sensibly about Security in an Uncertain World*. 2003. Copernicus Books.

Straub, D. W. & Welke, R. J. *Coping with Systems Risk:Security Planning Models for Managerial Decision-Making.* MIS Quarterly 1998. 22:4 , pp 441-469.

Whitten, A. & Tygar, J. D. *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0* . Proceedings of the 8th USENIX Security Symposium, August 1999, Washington 1999.