Simplifying development of secure software – Aspects and Agile methods

Licentiate Thesis

By

Gustav Boström

Department of Computer and System Sciences
Stockholm University and Royal Institute of Technology
Forum 100, SE-16440 Kista, Sweden

gusbo@kth.se

Abstract
Reducing the complexity of building secure software systems is an important goal as increased complexity can lead to more security flaws. This thesis aims at helping to reduce this complexity by investigating new programming techniques and software development methods for implementing secure software. We provide case studies on the use and effects of applying Aspect-oriented software development to Confidentiality, Access Control and Quality of Service implementation. We also investigate how eXtreme Programming can be used for simplifying the secure software development process by comparing it to the security engineering standards Common Criteria and the Systems Security Engineering Capability Maturity Model. We also explore the relationship between Aspect-oriented programming and Agile software development methods, such as eXtreme Programming.
Acknowledgements

I would like to thank my supervisor, Professor Paul Johanneson, for good support in discussing the direction of this research. I’m also very grateful for all the interesting conservations around security I’ve had with Professor Louise Yngström. This work also owes very much to my colleagues, Jaana Väyrynen, Martin Henkel and Mariné Bodén. Without our fruitful cooperation some of the articles would perhaps not even have been written and would certainly not have been of the same quality. Furthermore I would like to thank my lovely wife Mina for encouraging me in my work and for putting up with many late working nights. Finally I must thank my wonderful son, Filip, for putting his excellent sleeping abilities into use just when daddy needed to work.
# Table of Contents

Abstract .......................................................................................................................... 1
Acknowledgements ........................................................................................................ 3
Table of Contents ........................................................................................................... 5
Introduction .................................................................................................................... 7
Research Approach ........................................................................................................ 8
Aspect-oriented Software Development for security implementation ....................... 9
Secure software development methods and Agile software development ............... 10
Contributions of this thesis .......................................................................................... 11
Related work .................................................................................................................. 13
Conclusions and further work ...................................................................................... 14
Included papers ............................................................................................................. 15
References ..................................................................................................................... 16
Supplements .................................................................................................................. 19
**Introduction**

Building secure information systems is a difficult task. The computer industry has countless examples of systems being built with major security flaws [HoglundExploitingSoftware]. Even though several methods and technologies exist to address the problem, computer security still remains much of an art. One reason for this situation is that information systems today are becoming more and more complex. The Windows XP operating system for example now weighs in at 40 million lines of code. It is also a fact that large software systems lead to more bugs [HoglundExploitingSoftware]. The complexity is however not only due to size, but also the structure of the software. Programmers often have to face not only the complexity of their own business domain, such as banking, but they also at the same time have to deal with concerns such as access control, integrity, authentication, non-repudiation, confidentiality and distribution. Even though modularization techniques such as object-oriented software engineering and component based software has tried to alleviate this problem, security concerns have proven difficult to modularize due to their pervasive nature. Security logic cannot be easily located to a single module with today’s existing technologies [Laddad]. It has been shown this software complexity leads to flaws and security failures. [HoglundExploitingSoftware][ViegaSoC].

On top of these technical issues, developers also have to spend an often unproportionate amount of time on software engineering bureaucracy not always well adapted to the needs of their project. For example, one group used the Common Criteria to develop a secure electronic payment system [PalME]. This rather small project took 18 developers 3 months of time due to the heavy documentation burden. Instead of going with such complex heavyweight methodologies, developers often instead choose not to use any methodology at all.

Consequently it follows that addressing the complexity of secure software development is of great importance. Thus the goal we address in this thesis is to try to reduce this complexity issues by two means: By investigating a new programming technique, Aspect-oriented programming [AOP], and by exploring how secure software engineering process can be simplified. Thus we are in fact addressing two different kinds of complexity: Complexity in Program design and implementation and development process complexity.

The research areas covered in this thesis are thus: **Security engineering**, in particular Access Control and Confidentiality implementation, **Aspect-oriented programming**, and **Software development methodology**.

The thesis is organized as six self-contained papers. In the first paper we present a case study on how confidentiality can be implemented with better modularity using Aspect-oriented programming for applying encryption in a database. In the second paper, which is largely based on the first, we investigate how software engineering properties are affected by the use of Aspect-oriented programming for encryption implementation. The third paper investigates the use of Aspect-oriented programming (AOP) for access control implementation in a web-based application and how this leads to better modularity. In the fourth paper we study how eXtreme Programming [XP] can be used with security engineering in mind in order to simplify the development process. The fifth paper
investigates the relationship between Aspect-oriented programming and agile software development and how Aspect-oriented programming is a useful tool for mitigating some potential problems with agile software development. In the sixth paper we provide an example of how Aspect-oriented programming can be useful for inserting Quality of Service instrumentation into an existing legacy application.

To provide a background to the included papers the thesis starts with a short general overview of the need for Aspect-oriented programming to address security implementation complexity and also a section on why Agile software development can reduce the complexity of Secure software development from a process perspective. This is followed by section on the specific contributions of this work and a related works section. The thesis ends with conclusions and suggestions for further work.

**Research Approach**

The research approach taken in this thesis is dependent on the kind of complexity that is addressed. In trying to find ways to address the programmatic complexity of security implementation we have chosen to build prototypes implementing the concerns of confidentiality and access control. The prototypes act as case studies from which insights can be gained. In essence one could describe the methodology to be "proof by demonstration", e.g.: by building something one has proven that it can be done. This reasoning is however not enough since it is also important to analyze whether the new approach is better than existing approaches. The tool used to analyze the prototypes has largely been to reason about the software engineering properties of the implementations. Especially the concepts of Modularity and Maintainability have been discussed. To gain additional input the developers of the applications that have been used in the case studies have also been interviewed to get their view on the usefulness of an AOP-approach. The reason for choosing this qualitative approach is that at this moment it is still not clear how a good AOP security implementation should look like. Consequently it is necessary that the research at this stage should be of an exploratory nature. The drawback is that more work needs to be done for validating the approach further. The conclusions of this thesis can therefore be seen as mostly indicative. Nevertheless this is still useful since it greatly increases the level of knowledge available by providing a starting point. A different approach for analyzing the complexity of security implementations could have been to use software metrics, such as module dependency metrics, as an indicator of complexity [ZhaoMetrics]. In the authors opinion it is however difficult to apply these in this case. It is doubtful if metrics alone can provide a good indicator of complexity. Furthermore AOP metrics are very immature and has not yet been validated sufficiently. In the case of AOP it is also unclear if the existing proven metrics are actually applicable since AOP is quite different from existing programming paradigms.

To address the problem of process complexity in secure software development the method has been to analyze to what extent the new agile development method XP can be used to support security engineering. In this case we have chosen to base our analysis on two frameworks use in security engineering, the Systems Security Engineering Capability Maturity Model[SSE-CMM] and the Common Criteria[CC]. The idea is that the standards embed knowledge of the necessary activities required from a software development process in order to produce secure systems. This can be seen as a theoretical approach. To further increase the validity of these statements more empirical data should
be gathered. In this case however this is very difficult since few practitioners use agile methods for secure software today. Another approach to tackle complexity in the software development process could have been to investigate using software engineering tools to support the developers. This is clearly a possible approach, but we feel that before adding tool support it is necessary to first reduce complexity of the underlying process. As the example of Business Process Reengineering shows, one must first reengineer the process before automating. Another argument against tool support is that it is not always available to all developers. This can reduce the impact of such a solution.

Two of the papers in the thesis also deal with the connection between AOP and agile software development. The approach here has largely been to logically reason about how the software engineering properties of AOP can increase the technical agility of programming. That is to explore how AOP can be used for decreasing the costs of software evolution. The reasoning in this case has been based on a realistic constructed scenario. Another interesting approach to pursue could have been to investigate how AOP affects the design process of agile software development. Until recently however the number documented methods for Aspect oriented design have been few. Therefore this approach would have been difficult since it would have been necessary to also look deeply into that area.

**Aspect-oriented Software Development for security implementation**

Almost all of a company’s sensitive data is nowadays stored in its software systems. Authentication and Authorization mechanisms are necessary to protect this data from unauthorized access. This is referred to as **Access Control**. Often this needs to be done according to the roles that the employees have in the organization. Different groups of employees should be allowed only to see the subset of the information that they are entitled to. The customers of the company also need to have access to other subsets of the information. Several techniques exist that can provide the Authentication and Authorizations services necessary to fulfill these constraints. To handle the Authorization, Role-based Access Control model, RBAC, has become increasingly popular [RBAC]. Unfortunately despite of the existence of these techniques it is still very difficult to construct applications with confidentiality requirements [Viega]. This is due to the fact that even though Authentication and Authorization services do not need to be written by software developers, these services still need to be used and interact with domain objects of the application. This is not so difficult in small systems where this interaction can easily be dealt with manually in an unstructured way. However, as the system size increases the complexity of this task is also raised. This is due to the fact that Confidentiality is a so-called **Crosscutting concern** [AOP]. That is, it cuts across the entire application, across the different domain objects, the database, all the way out to the graphical user-interface. This leads to poor modularity, which in turn makes the application more complex and thus more difficult maintain and evolve. Object-orientation has been proposed as a technology that should make it easier to build modularized applications. However, Object-oriented Programming falls short when it comes to dealing with crosscutting concerns [Laddad]. Although techniques such as Patterns and Meta-object Protocols, can help with this in some instances, the resulting solutions are often still very complex. Object-orientation is good at modularizing for example the Authentication service in itself, that is the server part of authentication, but it fails to
modularize the important interaction with the Domain Objects, that is the client part [Laddad]. The failure to modularize confidentiality has the effect that developers cannot focus on one concern at the time. E g: When developing a form for showing salaries of employees in a Human Resources application the developers need to think not only about the business logic, but also the user interface logic, the authentication logic and the authorization logic. This leads to poor Separation of Concern since the business logic gets tangled with other concerns [Dijkstra]. This problem is particularly pronounced in the User-interface since the User-interface needs to be different for users that belong to different roles as this affects what they are allowed to see. This functionality is called Information Hiding or Information Filtering. Information Hiding is also a cross-cutting concern since it can be present in all parts of the user-interface.

Often it is also a requirement that sensitive data should be protected from unauthorized access outside of the application. That is, it needs to be protected at rest on secondary storage on disk. Furthermore data might need to be protected in transport over the network. These two concerns further add to the burden of the application developer since he needs to also think about Encryption in order to achieve confidentiality.

Unfortunately little security research has been performed in order to alleviate these problems in confidentiality implementation. Recent developments in the area of Advanced Separation of Concern in Object-oriented systems do however show great promise. Aspect-oriented Programming is a technique that has been proposed as an evolution of object-oriented programming and has as specific aim to handle cross-cutting concerns such as authorization in a simpler and more modular way [AOP]. Several security researchers have also proposed that AOP is an interesting technology for security implementation [DeWinHowAOP][DeWinAOPSoC][ViegaSoC]. DeWin have for example shown how Authentication and Authorization can be implemented in an object-oriented application [AOPPracticalExample]. However to the authors’ knowledge nothing is published on the subject of Information Hiding. Since Information Hiding is also a cross-cutting concern it can be assumed that AOP can prove to be beneficial also in this area. Encryption should also benefit from this approach. These developments are clearly valuable to the industry since they promise simpler development of secure applications.

**Secure software development methods and Agile software development**

In order to produce a secure software system it is necessary to work according to a process that takes security issues into account [Viega]. Unfortunately many of today’s existing software development methods neglect this. Security is therefore often seen as an activity separate to software development. There exist some attempts to include security engineering activities into the software development process as described by for example [RedwineNoopur]. However the processes are mostly examples of heavyweight approaches. A problem with this is that adopting such an approach can increase the complexity of developing software instead of providing help. It is the author’s opinion that the heavyweight nature of these processes can scare developers away from secure software development practices since use of the existing processes is believed to be too cumbersome to be practical. This is especially true for smaller teams who often use a more ad-hoc method of software development. Another disadvantage of most current methods for secure software development is that they are mostly based on a linear,
waterfall style where requirements are frozen before development begins [Lee][CC]. This is unfortunate as it is generally agreed today that iterative development is to prefer [Larman].

As a counter-reaction to heavyweight development methodologies Agile software development processes have become increasingly popular [Larman]. These methods take a less formal approach to software development, de-emphasizing documentation and emphasizing small teams, iterative development and quick customer feedback. These methods motto can be summarized in the Agile Manifesto[AgileManifesto]:

“… We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

Although these statements are primarily aimed at software development in general, we believe that these conclusions can be useful also for secure software development. However, it is also important to include security engineering activities when applying Agile methods. A generally agreed example of security engineering activities is hard to find, but the SSE-CMM [SSE-CMM] provides this overview:

1. Identify the organizational security risks
2. Define the security needs to counter identified risks
3. Transform the security needs into activities
4. Establish confidence and trustworthiness in correctness and effectiveness in a system
5. Determine that operational impacts due to residual security vulnerabilities in a system or its operation are tolerable (acceptable risks)
6. Integrate the efforts of all engineering disciplines and specialties into a combined understanding of the trustworthiness of a system

In our work we focus on how these security engineering activities affect the software development process. In particular we are examining how the agile method eXtreme Programming can be adapted to support these activities. The Common Criteria [CC] security standard, although not a process on its own, also provide guidance as to what activities a software development method should include in order to support the development of secure software. In our work we analyze XP in the light of CC and SSE-CMM to see to what extent XP can be used for secure software development.

**Contributions of this thesis**

The main goal of this thesis is to address the problem of complexity in secure software development. A first step is taken in the first paper: “Database encryption as an Aspect”.
The contribution here is to show how encryption can be applied in an existing application using aspects. The results were promising. It was possible to add encryption of database contents without changing the existing source code. The result is that the encryption logic can be clearly modularized. This leads to a better software structure which should lead to a system that is easier to understand and maintain than a system where encryption logic is tangled with business logic. However, the paper also revealed problems that can arise when encryption logic interacts with business logic when performing certain forms of database queries. The end result is that even though aspects are useful for implementing encryption, it is probably also necessary to refactor an existing application first, for example by introducing Object-relational mapping techniques that allow for better and cleaner joinpoints for the encryption aspect (See [AOP] for a definition of joinpoints.).

The second paper: “A case study on estimating the software engineering properties of implementing Database Encryption as an aspect”, is largely based on the first and elaborates on the effects on software engineering properties that the introduction of Aspect-oriented programming has. The conclusions are that although comprehensibility is improved due to increased modularity and less code tangling, AOP can also have negative effects since existing code can stop to work due to the injected logic of the encryption aspect. Tool support and testing is of high importance to mitigate this problem.

The third paper, “Aspects in the User interface – the case of Access Control”, presents another case study of applying AOP to the area of access control in an existing web-based application. The access control concern was heavily tangled in the existing code with reduced modularity and maintainability as a result. AOP can relieve this problem, but the problem of feature interaction, in this case with user-interface features, needs to be dealt with. This case study also points to the need of broader tool support for AOP than what is provided today. Access control concerns also need to be implemented on a database level and consequently AOP tools should support aspects also on the database level.

The fourth paper, “Security Engineering and eXtreme Programming, An impossible Marriage?”, addresses the issue of complexity in secure software development from a method perspective. Instead of attacking the problem from a programming viewpoint it looks at how Security Engineering relates to eXtreme Programming from a process standpoint. The reasoning is that the market today puts time pressures on software development activities and that cumbersome development processes, such as proposed by the PalME project [Palme] are not adequate. There is a need to use simpler, more agile methods also for secure software development. In order to properly address security engineering issues the eXtreme Programming process has to be adapted and activities added to ensure security assurance and proper definition of security requirements.

The fifth paper, “Aspects in the Agile Toolbox”, explores the relationship between Agile software development and AOP. It suggests that AOP is an important technique for Agile Software development since it facilitates adding concerns, such as security, late in the development cycle, although the example in this paper is a Quality of Service implementation. The sixth paper, “Moving from Internal to External Services using Aspects”, is largely based on the fifth and explores this issue further with a specific focus on Service Oriented Computing.
Related work

Using Aspect-oriented programming for security implementation is a relatively new field. Among the first publications are DeWin et al. with “On the importance of the separation-of-concerns principle in secure software engineering”. They emphasize the importance of the principle of separation of concern and how Aspect-oriented programming can help in achieving it. [DeWinAOPSoC]. DeWin et al. also provide more concrete examples on how to apply AOP for access control in [AOPPracticalExample] and [DeWinHowAOP]. Our work extends this work by providing case studies also for encryption and context-dependent access control which is not covered in their work. Laddad also has extensive examples of this in his book “AspectJ in Action” [Laddad]. These examples are however constructed examples and not based on real software systems. Devanbu and Stubblebine also point to the importance of AOP as an interesting avenue for future implementations of access control and authentication [Devanbu]. Bodkin provides an interesting overview of applications for AOP in Java application security [BodkinEntSecAspects]. He also has the most interesting description of how to implement access control concerns in the user interface. His approach however, is slightly different than ours in that it tries to filter elements from the HTML stream. We believe that refactoring the application to a more object-oriented structure is a cleaner approach. He also mentions encryption as a potential application of AOP. He however does not provide an implemented example of this. Laney et al. investigate how AOP can be used to adapt a system to fulfill integrity constraints through digital signatures [LaneyEvolution]. This is interesting work and shows that integrity can also be an interesting candidate for AOP implementation. Huang et al. has a proposal for a generic AspectJ aspect library for the security concerns, authorization, authentication, and auditing [HuangGenSecFramework]. The examples they provide however are quite short and again not based on real systems. Work that is not directly related to AOP, but nevertheless interesting for simplifying access control is [BridgingLangModel] and the work on secure UML [UMLSec][SecureUML]. This work is interesting since it could provide a way of specifying security constraints on a model level and not only on the code level. From these references it is however still unclear how an actual implementation of such constraints should be done. The authors of UML Sec have also started to investigate how to combine this approach with AOP [JurjensFoxModTrans]. Their approach however seems to stay at the modeling level.

In the area of methods for security engineering a significant body of research exists. Standards such as the Common Criteria (CC) [CC] and Secure Systems Engineering Capability Maturity Model (SSE-CMM) [SSE-CMM] are quite mature. There are also several examples on how these standards can be integrated into a development process. For example [EvertssonYngström][Lee]. The standards around security engineering are however biased towards development of large-scale systems and have an implicit assumption that the waterfall model of software development will be followed. A few researchers have however recently begun investigating how agile software development can be used in conjunction with security engineering. Some of the more interesting examples are [XSEBeznozov] and [BezKruchtenAssurance]. There are also some experience reports on using XP for high integrity applications [AmeyStaticVerXP]. Ron Morsicato and Mary Poppendieck also write about using XP for safety-critical applications [XPSafety]. Siponen et al. also provide some guidance on how to include
security engineering in agile processes [SecurityInAgile]. Although not explicitly related to agile software development, AEGIS is a method for lightweight iterative secure software development [AEGIS]. None of these authors however have taken our approach of comparing XP against SSE-CMM and CC.

Quality of Service implementation using Aspects in a CORBA-environment is discussed by Becker and Geihs in [QoSAspectMiddleware], but they provide no concrete example. Duzan et al. provide a working implementation using language extensions called Quality Objects [BBNDuzan]. Their implementation is however also somewhat CORBA-centric although it can also run on RMI. Verheecke et al. describe a Web Services specific example using AOP through the JaSCo platform [CibranWSML]. In this paper they however only briefly mention using AOP for monitoring and provide no concrete example. Arsanjani et al. also argue for using AOP to implement Quality of Service in [WSPromisesArsanjani], but they show no implementation.

**Conclusions and further work**

As a result of our studies we have come to the following conclusions:

- There are indications that AOP can be a useful technology for implementation of the security features Access Control and Confidentiality since it improves modularity.
- Quality of Service instrumentation is another area in which an AOP implementation is advantageous through easing the evolution towards instrumented Web Services.
- Agile software development methods, in particular XP, need and can be adapted to conform to security engineering practices as defined by the SSE-CMM and the Common Criteria.
- Agile software development and AOP are a good combination that reinforces the strengths of each other since AOP provides a new tool for reducing the pain of the continuous refactoring that is so emphasized in agile methods.

Although AOP is useful as an implementation technology for information security the case studies also revealed that feature interactions with other system concerns such as user interface logic and search functionality can be difficult to resolve. Restructuring of the code will often be necessary. It is therefore important to continue the work of validating an AOP approach for security implementation by working with more complex examples. It is necessary to provide guidelines for how the code should be structured for an easier implementation to be possible. In both the access control case and the confidentiality case it is also clear that the security aspects affect not only the programming language code, but it can also be necessary to weave security aspects into the database queries. This multi-level weaving is an interesting area to be studied. How should aspects on the database level be handled? How can both the database aspects and the programming language aspects be kept in synch? An interesting approach here could be to look at aspect orientation at a more abstract level, for example in modeling. Of particular interest could be to investigate how the access control constraints could be specified on a model level and then automatically or semi-automatically be transformed into an aspect implementation both in the user-interface, the business logic and in the
database. Current AOSD tools do not provide support for such multi-level weaving. However, the Concern Manipulation Environment [CME] appears to be a good basis on which to build. When it comes to Quality of Service implementation the approach has not yet been validated in a working system. The TrustCoM EU-project provides an opportunity for such validation [TrustCoM]. It also interesting to investigate other approaches for weaving in the Quality of Service aspect, for example by using the so called SOAP handlers available in Web Service implementations.

When it comes to simplifying the software development methods for secure software development we have only begun to investigate the area. The approach seems promising, but it is necessary to validate the approach. In particular we aim to provide a concrete example of how XP can be adapted with security engineering activities. Especially we will try to see how the security requirements activities can be integrated into the process. Integrating security assurance activities is also a necessary continuation of the work. This work should also be complemented with validating the methods in security related projects. The TrustCoM-project again provides several opportunities in the area of electronic contracts.

Another interesting possibility for further investigation is to do a deeper study on the relationships between AOP and Agile methods. In particular, could it be useful to apply aspect-oriented modeling techniques when dealing with security requirements in XP? Aspects could also be used as an assurance tool to make sure that the implementations follow the secure coding guidelines. The combination of test-driven development and aspects is also good team. However it would be interesting to provide guidelines for how to best use the techniques together.

Included papers

The following papers are included in the supplements to this thesis (See the section after the references.):

2. Gustav Boström, A case study on estimating the software engineering properties of implementing Database Encryption as an aspect, AOSD 2005, Workshop SPLAT
3. Gustav Boström, Aspects in the User interface – the case of Access Control, To be published
5. Gustav Boström, Jaana Wäyrynen, Martin Henkel, Aspects in the Agile Toolbox, AOSD 2005, Workshop SPLAT
6. Martin Henkel, Gustav Boström, Jaana Wäyrynen, Moving from Internal to External Services using Aspects, Proceedings of Interoperability of Enterprise Software and Applications 2005
References


[CibranWSML] Cibrán, M. A., Verheecke, B. and Jonckers, V., Modularizing Client-Side Web Service Management Aspects, Published in the proceedings of the 2nd Nordic Conference on Web Services (NCWS’03), Växjö (Sweden), November 2003


Supplements