Agile Development of Secure Web Applications

Xiaocheng Ge
Dept. of Computer Science
York University
York, UK
xchge@cs.york.ac.uk

Richard F. Paige
Dept. of Computer Science
University of York
York, UK
paige@cs.york.ac.uk

Fiona A.C. Polack
Dept. of Computer Science
University of York
York, UK
fiona@cs.york.ac.uk

Howard Chivers
Dept. of Information Systems
Cranfield University
Swindon, UK
hrchivers@iee.org

Phillip J. Brooke
School of Computing
University of Teesside
Middlesbrough, UK
p.j.brooke@tees.ac.uk

ABSTRACT

A secure system is one that is protected against specific undesired outcomes. Delivering a secure system, and particularly a secure web application, is not easy. Integrating general-purpose information systems development methods with security development activities could be a useful means to surmount these difficulties [6].

Agile processes, such as Extreme Programming, are of increasing interest in software development. Most significantly for web applications, agile processes encourage and embrace requirements change, which is a desirable characteristic for web application development.

In this paper, we present an agile process to deliver secure web applications. The contribution of the research is not the development of a new method or process that addresses security concerns. Rather, we investigate general-purpose information system development methods (e.g., Feature-Driven Development (FDD)) and mature security methods, namely risk analysis, and integrate them to address the development of secure web applications. The key features of our approach are (1) a process capable of dealing with the key challenges of web applications development, namely decreasing life-cycle times and frequently changing requirements; and (2) an iterative approach to risk analysis that integrates security design throughout the development process.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design—*Methodologies*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Feature Driven Development, Security Risk Assessment, Web Applications

General Terms

Design, Security

ICWE'06, July 11-14, 2006, Palo Alto, California, USA.
ACM 1-59593-352-2/06/0007.

1. INTRODUCTION

Traditional development methods for building web applications are typically *heavyweight* approaches, including variants on the Rational Unified Process [15], or Structured Systems Analysis and Design [12]. These approaches are increasingly contrasted with, and in some cases supplanted by, so-called *agile processes*, defined in terms of a small set of development principles and practices, e.g., the use of small development teams, short development life cycles, and focusing on the production of software deliverables as opposed to documentation or models. Key characteristics of agile processes include: accepting that requirements will change during development; incremental development of requirements throughout the life cycle; and emphasis on people and their interactions instead of compliance to a specific set of process objectives. The evident compatibility between the domain of web applications and the characteristics of agile processes (as, e.g., expressed in the Agile Manifesto [1]) have been documented elsewhere [17].

Security is a key requirement of web applications. In this paper, a *secure system* is one that is protected against *specific* undesired outcomes. It is accepted in the security community that considering security early in the information systems development life cycle can result in less expensive and more comprehensive coverage of security requirements than by attempting to add mechanisms to an operation system. Security development methods have been studied for several decades. Early work was categoried by Baskerville [6]. The growing trend towards the use of agile techniques for building web applications means that it is essential that security engineering methods are integrated with agile processes. This is, in itself, a substantial challenge; some approaches are highlighted in [10, 18, 7].

At the heart of many security engineering methods is *risk assessment*. Risk-based reasoning and assessment provide decision criteria for assessing security requirements in their business and social context. Risk assessment is widely regarded as the only viable method of providing a cost-benefit justification for security controls, or alternatively, of judging which controls provide the most benefit [9].

The aim of this paper is to integrate agile processes and risk assessment for building web applications, a domain where both agility and risk analysis can and must play important roles.

Compared with other security engineering methods, such as SSADM-CRAMM [2], CRAMM [3] and SDLC [13], the approach offered in this paper has two key novelties.

1. It is an agile process. While we believe that both agile and heavyweight processes, e.g., model-driven development, have merits in different contexts, agile processes are particularly well suited for building web applications, particularly because they aim to deal with changing requirements and rapid delivery [17].

2. It is an engineering method which aims to deliver mechanisms to satisfy security requirements, as well as mechanisms for satisfying functional requirements, for web applications. Security is addressed at every stage of the development life cycle, and as such the approach presented is a thorough integration of security engineering techniques with mainstream development processes for building web applications.

2. BACKGROUND

2.1 Feature Driven Development

Feature-Driven Development (FDD) [19] is one of many agile processes. It is based on the Agile Manifesto, and provides concrete development practices that aim to implement the Manifesto's principles through the use of lightweight *modelling*. Like other agile processes such as Extreme Programming (XP) and Test-Driven Development (TDD), FDD focuses on short iterations that deliver tangible units of functionality. Unlike many other agile processes, FDD is a process that is model-driven; models (e.g., written in standardised languages such as UML) are created to help identify units of functionality, understand the context in which units are used, and to help derive code. As such, FDD implements the agile practice of "modelling with purpose".

FDD is based on the idea of discovering and implementing system *features*. A feature is a client-valued distinct unit of functionality. In many cases, features may correspond to system use cases; however, features may also consist of units of functionality that are not directly used or invoked by an external actor.

When contrasted with other agile processes, particularly XP, FDD has a straightforward process, which contains five phases (shown in Figure 1). The first three phases are performed at the beginning of the project and focus on planning. The last two phases are implemented, in practice, by a series of short iterations in which Design by Feature and Build by Feature are carried out. Briefly, these five phases are:

- Develop an Overall Model (DOM), an initial project-wide activity with domain and development members under the guidance of an experienced modeller in the role of Chief Architect.

- Build a Features List (BFL), an initial project-wide activity to identify all the features to support the requirements.

- Plan by Feature (PBF), an initial project-wide activity to produce the development plan.

- Design by Feature (DBF), a per-feature activity to produce the feature design package.

- Build by Feature (BBF), a per-feature activity to produce a completed client-value function (feature).

Typically, only a use case model and a class (or object) model are produced when carrying out FDD iterations.

2.2 Security Risk Analysis

In security terms, *risk* is a measure of the likelihood of a specific undesired outcome to a system asset. Risk assessment is the basis of most national standards for information security management [14, 21, 8], and is even a rational approach to broader security choices in society [20].

Risk assessment is not only the process of identifying risks and their characteristics, but also the determination of the exposure to risks. Such an activity typically involves members of the development team performing the following steps (details in [21]) in an iterative, incremental, timeboxed, and ongoing manner:

1. Understand the system contents:
   - System Boundary.
   - System Functions.
   - Assets at risk.
   - Business processes at risk.
   - Threats to these assets and business processes.
2. Analyze the identified risks:
   - List the potential vulnerabilities.
   - Analyze the vulnerabilities of the assets and business processes to these threats.
   - Estimate the risks probabilities of occurrence.
   - Estimate the potential impact of each risk to the success of the endeavor.
   - Estimate the importance and priority of each risk.
   - Categorize the risks.
3. Recommend specific actions and techniques to each significant risk:
4. Assign responsibilities and resources to perform risk avoidance and document the risks.

Risk assessment produces a report that describes the threats and vulnerabilities, measures the risk, and provides recommendations for control implementation.

2.3 Agile security

There is previous work on using agile processes to build secure systems. Beznosov presents a conceptual analysis of the suitability of Extreme Programming for building secure systems, and also introduced the notion of "good enough security" without defining it *a priori* [7]. The objective of the Planning Game, one of the XP Practices, is defined so as to plan small releases in short iterations while delivering 'good enough' security through tested functionality units. [7] also gives extended definitions of other practices such as testing, continuous integration, simple design and refactoring which are adapted to so-called 'Extreme Security Engineering'.

Chivers et al [10] proposes that Agile Security can be achieved by using an Incremental Security Architecture (ISA). Moreover, [10] states that "instead of following traditional techniques, [an Agile process] must have its own, agile, security practices". Aydal [4] presented an interesting attempt to use Refactoring and a modified version of the Planning Game to iteratively and incrementally retrofit security mechanisms to a complete system, within the context of an Extreme Programming development.

None of the work discussed above has focused strictly on the domain of web applications, though [4] explores refactoring patterns for producing Web services.

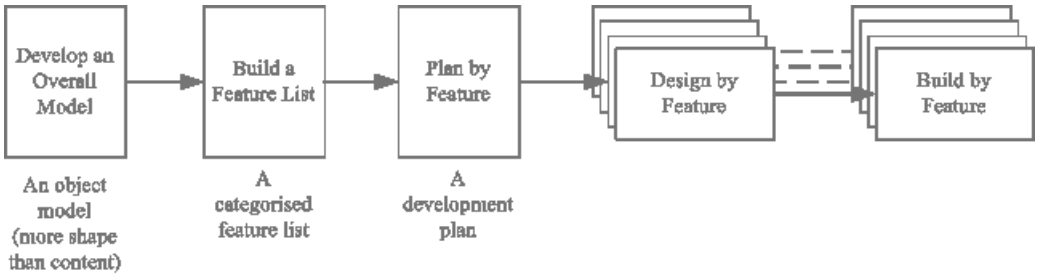

Figure 1: Brief Description of FDD Process

3. AGILE DEVELOPMENT OF SECURE WEB APPLICATIONS

In this section we present an agile process for building secure web applications. The process is agile because its development phases and activities satisfy the principles of the Agile Manifesto (especially principles focusing on simplicity in design and modelling for purpose), and also because the approach is an extension to an existing agile process, FDD.

The structure of the process is organised as shown in Figure 2. It is an extension of FDD. FDD activities are represented by rounded rectangles. Specific subtasks within FDD activities are represented by rectangles. This collection of subtasks now includes (a) specific, new subtasks for security analysis; and (b) tailored tasks for web applications (i.e., content design). As part of the activities, models are produced, particularly a use case model (which forms the basis of building a feature list and planning-by-feature), and a class/object model (forming the basis of design).

We now briefly describe each task and subtask in the overall process, and explain why extensions and modifications maintain agility of the whole process.

- **Requirements analysis**. This activity aims at determining the needs and expectations of relevant stakeholders, as is done in many proposed web application development methods, such as [5, 16, 11]. Our approach is no different from these, except in terms of completeness: we aim to produce a requirements domain model sufficiently detailed to feed in to the next steps, i.e. particularly to determine a list of features for the overall system. In this sense, the analysis activity is founded on the agile principle of "modelling for a purpose".

- **Security Policy Decision**. A security policy is the set of rules, principles, and practices that determine how security is implemented. Ideally, the development team should create and update a document describing the system security policy integral with the rest of the development process. The security policy will help developers think out and plan the application design specifically with security in mind. Each time the requirements change, someone on the development team should be responsible for reviewing the policy with executive team members and making necessary changes in the document. This makes it easier to effect changes in codes that honor and protect the security policy's requirements.

- **Use case analysis**. The purpose of this activity is to make a categorised list of Features offered by the system, as well as a development plan based on this list. The domain model (DOM) constructed earlier describes aspects of the business, irrespective of whether all of those aspects are delivered by the system. The use case analysis is used to detail this model and tranform it into a system model which has a clear scope of the system when contrasted with the business model. Here, the system model is an abstract use case model, which means any details of implementation including detailed design of use cases, implementation platform, and physical deployment are not considered. This in effect implements the agile practice of modelling with purpose: only those modelling elements needed to help determine the list of features and a plan of attack for later stages are used.

- **Content Design**. This is the major design activity which considers the implementation of each feature. It consists of *structure* and *functional* design. Structure design specifies and organises the contents related to the feature; and functional design specifies user operations of the feature. In terms of modelling, content design starts with a detailed use case of a particular feature. The developers must refine this model, e.g., using UML class or collaboration diagrams. These models omit details, e.g., how to implement and deploy the overall system. Then the developers need to choose the platform of implementation, such as J2EE or .NET. Platform specific details must then be merged with the Content Design to obtain a concrete model, which can be more easily mapped to source codes or database schema.

- **Security Risk Analysis**. This is one of the key features of our approach. ¿From the view of a project manager, security risk analysis is an iterative, incremental, ongoing process. In a particular iteration of our approach, the activities of Content Design and implementation focus on one feature, but risk analysis is also performed for any existing and integrated applications because new vulnerabilities may be introduced when a new feature is built and coupled with other applications. This is particularly critical for web applications. The result of risk analysis may modify the Content Design. A Feature *including* security controls is thereafter built.

  Risk analysis is also based on a model. The underlying risk model is that *attackers* seek to harm *assets* by obtaining access via system *vulnerabilities*. The only harms that matter are specific unwanted outcomes or concerns, with associated *impacts*. The set of potential attack paths between attackers and the assets of concern are *threats* that are considered during risk analysis. There are a number of possible responses to a threat, including simply deciding that the risk can be accepted, but during the early lifecycle of a system the designer is concerned with establishing a security policy.

- **Implementation**. Finally, in this task, we transform the concrete model of the Feature into a release, which is a exe-

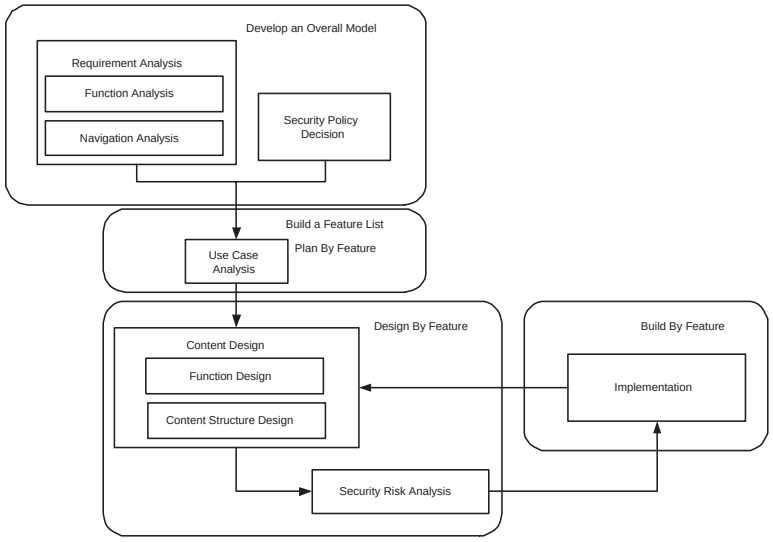

Figure 2: Secure Web Application Development Process

cutable system that satisfies our current understanding of any security requirements.

The critical point to note is that risk analysis is iterative and incremental, but must still take into account existing features (e.g., features developed in previous iterations). This is no different from typical agile verification and validation practices, which must assess both newly added code (e.g., by running new unit tests) and its interactions with existing code (e.g., by re-running all existing unit tests).

We now illustrate the use of the approach in a short case study.

4. CASE STUDY

We use an illustrative case study of a simple web application for an estate agency. The initial requirements are as follows.

> An estate agency needs a system to offer the possibility of researching properties online. The user starts to browse the web site of agency; and may input some property criteria (eg. location, price range, etc.). The system processes the query and display the list of properties that match the criteria.
>
> Furthermore, the system provides a link to each property on the list, and allows users to submit an offer, or to review their previous offers. To access the offer functions, a user must log into the system (i.e. provide username/password and sign in).

4.1 Overview Model

The system use cases can almost be documented directly. The use case model (figure 3) presents the main functions of the system and associates them with roles. The internet user searches properties, submits an offer, and reads offers. The use cases for *submit an offer* and *read offers* << *includes* >> the *sign in* use case.

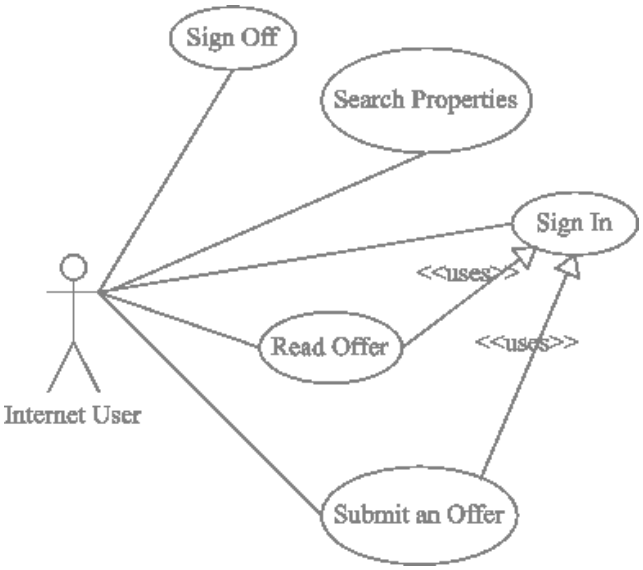

Figure 3: Use Case Model

4.2 Security Policy

A security policy always has regard to the organisation that owns a system; different organisation may have different security policies for similar systems. In general, a security policy may include the following:

1. Overview
    - Importance of Security
    - Security Goals
    - Security Principles
    - Security Policy Scope
2. Security Policy
    - Identification
    - Authentication
    - Authorization
    - Integrity
    - Intrusion Detection
    - Nonrepudiation
    - Privacy
    - Security Auditing
    - Physical Protection
    - System Maintenance Security
3. Responsibilities

For our illustrative system, we focus on the following security goals and policies.

- Security Goals
    - Restricted functions and information shall only be available to identified, authenticated, and authorized individuals.
    - The integrity of a property's data, whether stored or communicated, shall be secured.
    - The details of each client are confidential data and shall remain private.
- Security Policies
    - Authentication. The application shall verify the identity of all users before allowing them to perform their associated tasks.
    - Authorization. Each user shall be granted only access sufficient to perform the tasks for which they have been explicitly authorized.

4.3 Iterative Development Plan

The four component use cases give rise to the four features (F1 to F4) of a development plan. The development iterates over the four features. F1, *search properties* has no dependent features. F2, *sign in / out*, is second, because the remaining features include it. The remaining features are F3 *Submit an offer*; F4 *Read Offers*. To illustrate our approach, the first two iterations are described.

4.3.1 Iteration I: Search Properties

Figure 4 shows the **conceptual model** of the Search Properties feature. A user enters a set of criteria, that may match the attributes of some properties. The result of matching criteria to property attributes is a list of results. The model uses a class stereotype, $<< data >>$; this represents a class with no methods, an abstract model of a relation in a database.

Taking the conceptual model and the details of implementation media (which here include J2EE and a relational database architecture) we derive a more concrete **development model**, which shows

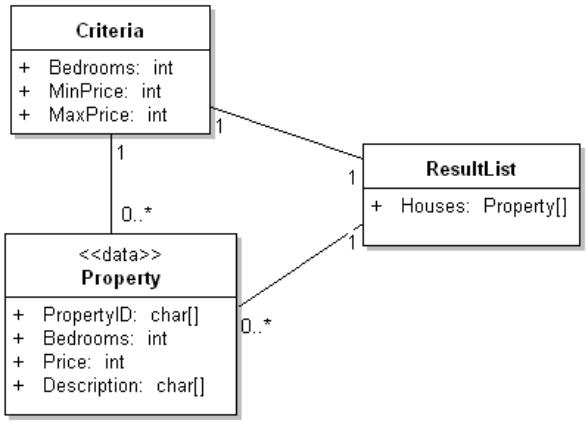

Figure 4: Conceptual model of Feature Search Properties

the objects and collaborations of a user's interaction with the system. In figure 5, the actions of a user *Internet User* enable one of the featured web services — here, the *search* operation — which is represented using the stereotype, web.service. The operation generates an object for the entered *criteria*, and invokes a database session (stereotype db.service). The database session encapsulates the querying of the properties database; the operation generates a result list object. The web service, *Display* takes the result list object and displays the contents to the user.

Risk Analysis

From figure 5, we observe that, at this stage, there is one data asset, *property*. The security policy requires that the system protect integrity, but the design allows access to this asset by any internet user browsing the estate agency system. From the diagram, we can see that only the *search* object can invoke a database session, necessary for access to the asset. We propose to use database privieges to restrict access to the *property* data, and block the attack path.

4.3.2 Iteration II: Sign In

The second iteration starts by considering the next feature in isolation. Figure 6 shows the **conceptual model** for the *sign in* feature. The concrete **development model** (figure 7) again shows the Internet User enacting the appropriate web service, which invokes a database session. The database session queries the *client* data, and the *sign in* service generates a *security token* to record the outcome of the database session.

Risk Analysis

The risk analysis for the second iteration needs to consider the whole of the system developed so far. This is expressed diagrammatically in figure 8. In addition to the data assets, *property* and *client*, according to the risk analysis the session object, *security token* is vulnerable to attack by a user.

As an example, we highlight analysis of one attack path, that from *search* to *client*. The web service, *search*, invokes a database transaction, which could be used to access and attack *client* (for instance, using an SQL injection[1]). Figure 8 shows that there are two segments to the attack path — between *search* and *database* and between *database* and *client*. Here, we elect to use database access controls to block irregular access to the *client* asset.

[1]SQL injection cannot generally defended by simple access control because of the use of a connection pool in practice.

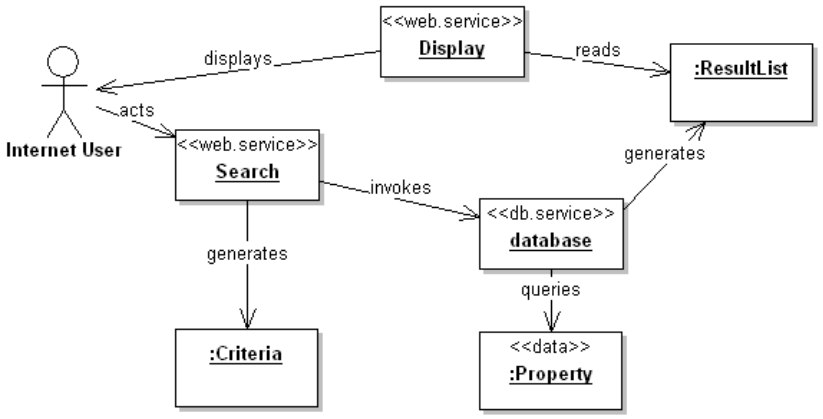

Figure 5: Concrete Development Model of *Search Properties* Feature

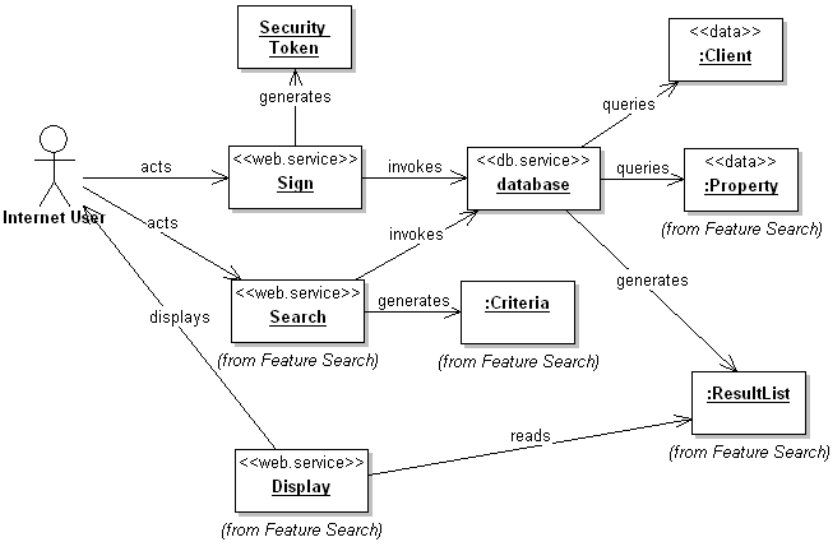

Figure 8: Risk Analysis in the Iteration of Feature Sign In

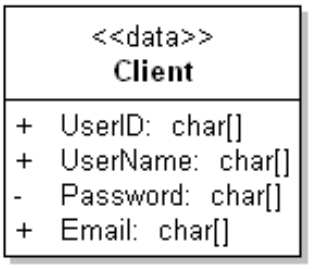

Figure 6: Conceptual model for the *Sign In* Feature

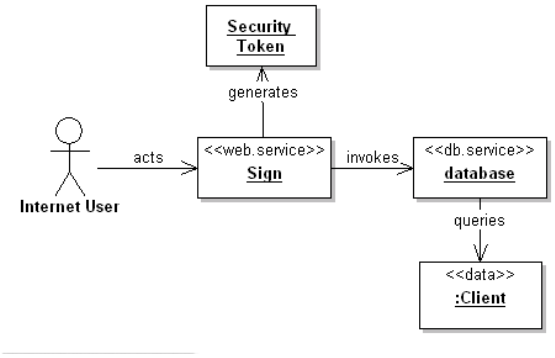

Figure 7: Concrete Development Model of Feature Sign In

4.4 Security Evolution and Futher Discussion

The case study demonstrates how the agile approach proceeds by simple design-build iterations. At each iteration, there is a risk analysis of the whole system, an analysis in principle that becomes increasingly complicated. However, we observe the controls implemented in early iterations may also mitigate attacks that potentially arise in later iterations. The notion of Incremental Security Architecture [10] is also a mechanism that can help make the analysis simpler, via triggering of refactorings to make explicit how security mechanisms in the system help to satisfy security requirements (or mitigate risks or vulnerabilities). For example, when we add the feature to allow reading of offer information, the *read* object again invokes a database session, opening another attack path on the *client* data. However, the access controls put in place in iteration II pre-empt use of this attack path. Our approach serves to increase confidence that each new iteration leaves the system secure against identified risks, without jeopardising existing security mechanisms.

In the case study, our approach is only applied to application-specific security. To get a higher level of security, we need to consider generic issues such as physical and business security. That is not difficult in theory, simply requiring a broader analysis of security issues at each iteration. During that analysis, the developers would examine not only the system design, but also the environment of the running system. Preventative mechanisms might include implementation of a firewall, protection of the web and database servers, training for agency personnel etc.

5. CONCLUSIONS

Web applications such as our example estate agency system commonly evolve from existing database systems. Heavyweight development and security methods such as SDLC [13] are not amenable to use with evolving systems. Many developments omit security consideration from the development process, attempting to add on security mechanisms in an ad hoc way after implementation.

Our agile approach provides a lightweight development environment, with incremental risk analysis and provision of appropriate security mechanisms at each increment. At each increment, we determine what assets have been added, and what new attack paths might have been opened. The analysis of the attack potential takes account of existing asset protection. By re-assessing risks at each increment, our approach seeks to avoid the addition of mechanisms that inhibit or invalidate existing security mechanisms. As we perform the risk analysis, each iteration and increment recommends a list of security controls. These controls, when implemented, will affect different layers of a typical web application (e.g., the browser interface, the application logic, the network infrastructure, supporting database). Controls that affect platform-specific layers seem to be more likely to have a strong influence on successive iterations. Of course, this is all based on the assumption of a clear, thorough business analysis and a clear description of the overall security policy (or policies). If, during an iteration, a policy is changed, developers are likely to need to analyse a substantial part of the existing system to ensure that it is still secure against the modified policy.

The development process of our experimental system shows where the risk analysis locates itself in an agile development process and how its result affect the later release of the system. What is particularly interesting about the process is that it appears to be generic, i.e., amenable to use for building any type of system, including web applications, in a context where risk analysis is important.

We are continuing to develop the full estate agent system, and also intend to apply the agile process for building a secure Grid

which involves overlapping (and potentially inconsistent) security policies that must be reconciled.

Acknowledgments. Research supported by the EPSRC, GR/S64226/01.

6. REFERENCES

[1] Agile Manifesto. http://agilemanifesto.org.

[2] SSADM-CRAMM subject guide for SSADM version 3 and CRAMM version 2. Technical report, Central Computer and Telecommunications Agency, IT Security and Privacy Group., 1991.

[3] CRAMM. Technical Report http://www.cramm.com, Insight Consulting Limited, 2003.

[4] E. Aydal. Extreme programming and refactoring for building secure web-based applications and web services. MSc in Software Engineering Thesis, Computer Science Department, University of York, 2005.

[5] L. Baresi, F. Garzotto, and P. Paolini. Extending UML for modeling web applications. In *Proceeding of 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3*, Maui, Hawaii, USA, January 2001. IEEE.

[6] R. Baskerville. Information systems security design methods: Implications for information systems development. *ACM Computing Surveys*, 25(4):375–414, 1993.

[7] K. Beznosov. eXtreme Security Engineering. In *Proceeding of First ACM BizSec Workshop*, Fairfax VA, USA, October 2003.

[8] CERT Coordination Centre. Operationally critical threat, asset, and vulnerability evaluation (OCTAVE). Technical Report http://www.cert.org/octave/, Software Engineering Institute, CERT Coordination Centre, 2003.

[9] H. Chivers. Security and systems engineering. Technical Report YCS378, Department of Computer Science, University of York, June 1994.

[10] H. Chivers, R. Paige, and X. Ge. Agile security using an incremental security architecture. In *Proceeding of the Sixth International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2005)*, Spring-Verlag LNCS 3556, pages 57–65, Sheffield, UK, 2005.

[11] F. Garzotto, P. Paolini, and D. Schwabe. HDM — model-based approach to hypertext application design. *ACM Trans. Inf. Syst.*, 11(1):1–26, 1993.

[12] M. Goodland and C. Slater. *SSADM Version 4: A Practical Approach*. McGRAW-HILL Book Company Europe, 1995.

[13] T. Grance, J. Hash, and M. Stevens. Security considerations in the information system development life cycle. Technical report, National Institute of Standards and Technology (NIST), Special Publication 800-64, October 2003. (revision 1 released June 2004).

[14] B. S. Institution. Information security mangement part 2: Specification for information security management systems. Technical report, BS 7799-2:1999, 1999.

[15] P. Kruchten. *The Rational Unified Process: an Introduction*. Addison-Wesley, 2003.

[16] G. R. Lifia, H. Schmid, and F. Lyardet. Engineering business processes in web applications: Modeling and navigation issues. In *Proceeding of 3ird International Workshop on Web-Oriented Software Technologies, IWWOST'03*, July 2003.

[17] A. McDonald and R. Welland. Agile web engineering (AWE) process. Technical report, Department of Computer Science, University of Glasgow, UK, December 2001.

[18] R. Paige, J. Cakic, X. Ge, and H. Chivers. Towards agile reengineering of dependable grid applications. In *Proceeding of 17th International Conference of Software and System Engineering and Their Applications (ICSSEA)*, CNAM, Paris, November 2004.

[19] S. R. Palmer and J. M. Felsing. *A Practical Guide to Feature-Driven Development*. Prentice Hall, 2002.

[20] B. Schenier. *Beyond Fear: Thinking Sensibly About Security in an Uncertain World*. Copernicus Books, 2003.

[21] G. Stoneburner, A. Goguen, and A. Feringa. Risk management guide for information technology systems. Technical report, National Institute of Standards and Technology (NIST), Special Publication 800-30, July 2002.