Modelling and conception of software in aspect to security

UHH

Seminar: Software Architecture
Universität Hamburg
Shahin Imanverdiyev & Senad Ličina

ABSTRACT

put Abstract here. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec nec efficitur enim. Mauris leo metus, fermentum dapibus lobortis eu, gravida at libero. Vestibulum eu sem eu sem finibus convallis nec at lectus. Donec finibus dui eget eros bibendum, sit amet pharetra odio rutrum. Etiam sodales suscipit mattis. Sed molestie sagittis rutrum. Vivamus non luctus velit, a venenatis felis. In non sem lacinia, vehicula nisi eget, elementum nibh. Curabitur faucibus est leo, vel placerat lacus dapibus eu. In ut enim a nisl facilisis viverra ut nec tellus.

1. INTRODUCTION

Modern society and economy increasingly depend on digital systems. As attacks against such systems can have devastating results, it is very important to secure such systems in a proper way. Nowadays the Internet is a heavily used and very important communication medium. An increasing number of devices are connected to the Internet, which is why it is important to transmit and store sensitive data securely.

The correct development of secure software is difficult. There have been numerous successful attacks abusing vulnerabilities of software systems in the past and it can be expected that people will try to detect and abuse such flaws in the future as well.

The traditional approach for security assurance has been "penetrate and patch", where security is assured by attempting to break into a running system and exploiting well-known vulnerabilities. Penetrate and patch happens too late in the development process and vulnerabilities will be available and possibly exploited until they are recognized fixed. This is why it is important to take security aspects into account in an early stage of the system development.

2. REQUIREMENTS

In this section some concepts are explained which are required to understand this paper. First the key concepts of security are discussed, ...

2.1 Security Principles

The CIA Triad is a well-known security model which is based on the three key principles of security: **confidentiality**, **integrity** and **availability**. It is used to identify problem areas in the security of software systems and requires that its principles are preserved for the system resources.

Confidentiality ensures that only authorized users are able to read private or confidential information. Access control mechanisms can be used to gain better confidentiality.

Integrity (in the field of computer security) prevents information from modification or deletion by unauthorized users and ensures that undesired changes can be undone.

Availability refers to the prevention of unauthorized denial of access to information or resources.

These key principles are often described as incomplete and thus extended by further principles like **authenticity**, **accountablility** or **non-repudiation**.

Authenticity is the assurance that information and resources are valid. The validation that all involved parties are truly who they claim to be is important for authenticity.

Accountablility refers to the property that every security crucial action can be traced back to the responsible

Non-repudiation assures that the existence of specific actions can not be denied.

2.2 Unified Modeling Language

> "The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system."
> – [BJR99, Chapter 1]

The Unified Modeling Language (UML) is the prevalent language for the specification of object orientated software systems. It describes architecture and behavior of software in a graphical manner using various diagram types.

Use case diagrams model a systems functionality as percieved by users (or actors). Use case diagrams consist of use cases and actors, they can be used to represent interactions between system and user. An example for a use case diagram is given with figure 1. The actors (customer, online banking & banking system) are depicted by stick-figures, the use cases (transfer funds, deposit funds & withdraw cash) are represented as ovals.
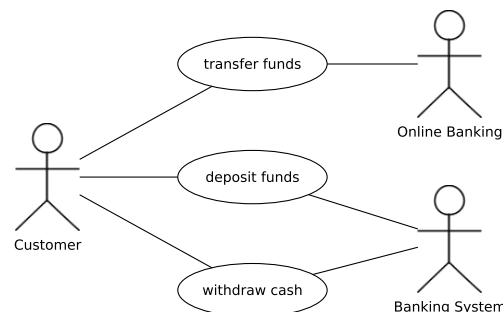

Fig. 1. Example of an use case diagram representing possible interactions between a customer and banking interfaces.

Class diagrams model static concepts within the implementation of an application. Classes are drawn as rectangles consisting of three sections for its name, its attributes and its operations. Relationships among classes are drawn as paths between them, whereby different arrow types have different meanings. Figure 2 shows a class diagram representing different implementations of the *AccountInterface* and an implementation for the *CustomerInterface*. The classes *CheckingAccount* and *SavingsAccount* are inheriting from the class *Account* which is implementing the *AccountInterface*.
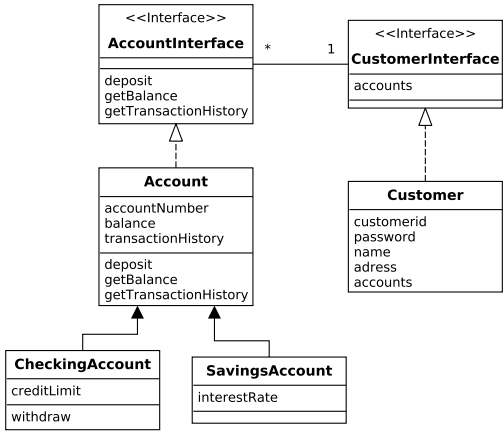

Fig. 2. A class diagram representing example classes for customer and accounts.

Statechart diagrams are visual representations of state machines. State machines model the sequences of states which an entity can go through. States are connected by transitions, which are allowed to have conditions. The initial state is depicted by a black dot, the final state as a black dot in a circle. Figure 3 shows the states an account object can go through.
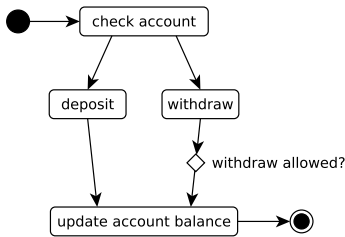

Fig. 3. A statechart diagram depicting the simplified processes cash withdrawal and deposition on an account.

Sequence diagrams shows object interactions by arranging messages in time sequence. The vertical lines are lifelines which are representing classifier roles. Messages are drawn as arrows between lifelines. An example sequence diagram for the scenatio where a customer is withdrawing money at an ATM is given with figure 4. Figure 4 shows an sequence diagram for the example scenario of money withdrawal at an ATM.
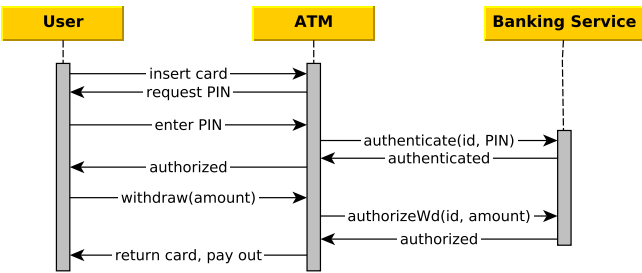

Fig. 4. Money withdrawal at an ATM as a sequence diagram.

3. MODELING SECURE SOFTWARE WITH UML

Citation goes like this... [SHRB11] footnotes [1] are possible aswell. Reference to Section 2. blabla wichtig intentionally left blank.

4. CONCLUSION

Typical Conclusion.

5. REFERENCES

[BJR99] Grady Booch, Ivar Jacobson, James Rumbaugh: The Unified Modeling Language Reference Manual. Addison-Wesley, Reading (US-MA) 1999.

[JS11] Jan Jürjens, Holger Schmidt: UMLsec 4 UML2 - Adopting UMLsec to Support UML2. Technische Universität Dortmund, Dortmund, Germany 2011.

[Jür06] Jan Jürjens: Secure Systems Development with UML. Springer-Verlag, Berlin, Germany 2006.

[SHRB11] Matthias Straka, Stefan Hauswiesner, Matthias Rüther, Horst Bischof: Skeletal Graph Based Human Pose Estimation in Real-Time. Graz University of Technology, Austria 2011.

---

[1]this is a footnote.