CIS 418/518 – Secure Software Engineering
Requirements Engineering for Secure Software

Jagadeesh Nandigam

School of Computing
Grand Valley State University
nandigaj@gvsu.edu

Outline

Use Cases

- A *use case* defines a goal-oriented set of interactions between external actors and the system under consideration.
- *Actors* are parties outside the system that interact with the system.
  - Role played by someone or something
  - External to the system
  - Interacts with the system
  - Tries to achieve a goal
- Use cases capture who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals.
- Use cases are suitable for capturing *functional requirements*.

Use Cases

- Two types of actors based on how an actor interacts with the system.
  - A *primary actor* is one that uses the system to achieve a goal.
  - A *secondary or supporting actor* is one that the system needs assistance from in completing a goal (i.e., a use case).
- In UML, a use case is represented by an oval and an actor is represented as a stick person.

Use Case Specification

- Use cases are textual descriptions/specifications of behavioral requirements to achieve a goal.
- We write use cases, we don't draw them.
- A use case specification includes:
  - Name
  - Identifier
  - Description
  - Actors (primary as well as secondary)
  - Triggers
  - Preconditions
  - Primary flow
  - Alternate flows
  - Postconditons (minimal and success guarantees)
  - Nonfunctional requirements

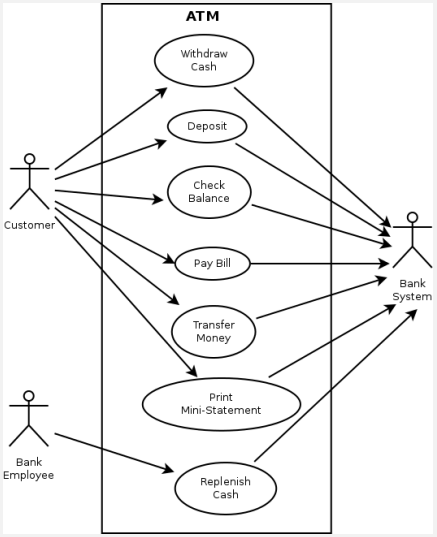
Use Case Specification

- A scenario represents a single path through the use case.
- A use case represents one or more scenarios.
  - A scenario for the main/primary flow through the use case.
  - Alternate scenarios for each possible variation of flow through the use case.
  - Alternate scenarios are triggered by options, error/exception conditions, security breaches, etc.
  - Scenarios may be depicted using UML sequence diagrams.

Use Case Diagrams

- A *use case diagram* is a visual representation of the relationships between actors and use cases.
- Arrows and lines are drawn between actors and use cases and between use cases to show their relationships.
- A use case diagram shows which actors interact with which use cases. Beyond this, it contains very litte information of use to developers.
- The details of a use case is in it's textual specification.

Use Case Diagrams



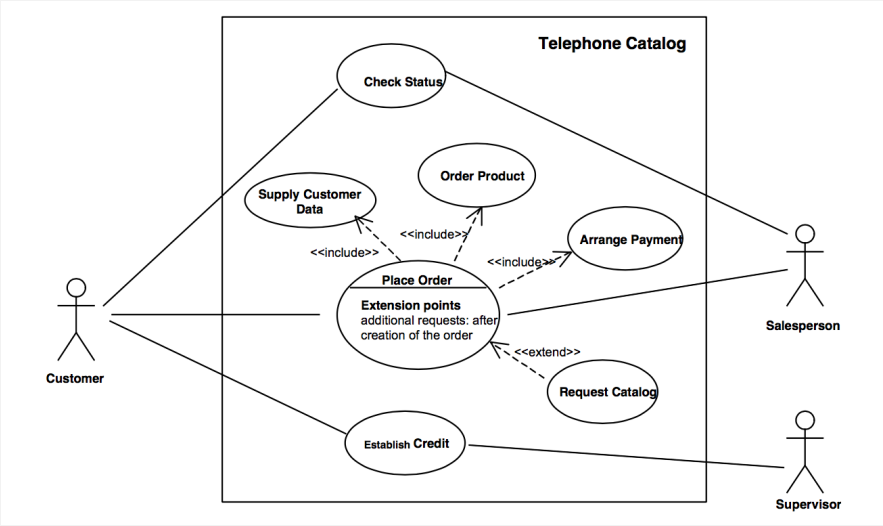
Structuring Use Cases

- Three relationships in a use case diagram:
  - *include*
  - *extend*
  - *generalization*
- The *include* relationship between two use cases indicates that the sequence of behavior described in the included (or sub) use case is inserted in the sequence of the base (including) use case.
- Included use cases are used to extract common parts of the behavior of two or more use cases.

Structuring Use Cases

- The *extend* relationship provides a way of capturing a variant (optional and supplementary) to a use case.
- The extend relationship specifies the condition (via extension points in the base use case) that must be satisfied if the extending use case is to execute.
- A *generalization* relationship is used when you find two or more use cases that have commonalities in behavior, structure, and purpose. When this happens, you can describe the shared parts in a new, often abstract, parent use case, that is then specialized by child use cases.

Structuring Use Cases

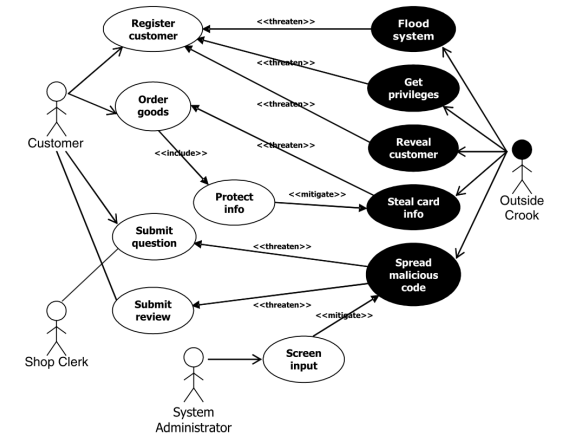


Misuse Cases

- Misuse cases specify behavior **not wanted** in the proposed system for the purpose of identifying and eliciting **security requirements**.
- *Misuse Case* – A sequence of actions, including variants, that a system can perform, interacting with misusers of the system and causing harm to some stakeholder if the sequence is allowed to complete.
- *Misuser* – An actor that initiates misuse cases, either intentionally or inadvertently.
- Specific relationships between use and misuse cases:
  - *threaten*
  - *mitigate*

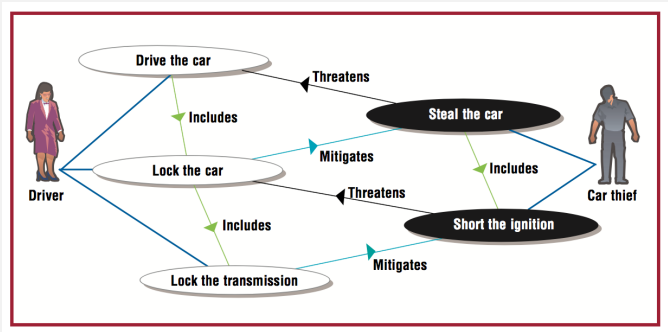
Misuse Cases

- A regular use case represents a general functional requirement.
- A misuse case represents a **threat** to a regular use case.
- Security use cases represent **security requirements** that **mitigate** the security threats.



Misuse Cases

- Both use and misuse cases can include subsidiary cases of their own kind.
- Misuse cases threaten use cases with failure, and appropriate (security) use cases can mitigate known misuse.



Misuse Case Specification

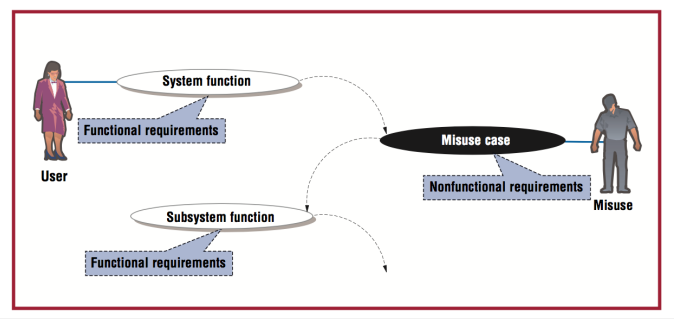
- Two ways of expressing misuse cases textually:
  - Lightweight misuse case description
  - Extensive misuse case description
- The lightweight approach embeds the description of misuse within a regular use case template using a **Threats** field or column.
- Lightweight descriptions are appropriate
  - When brainstorming, early in development, to get an overview of the threats faced by the system.
  - When specifying misuse cases believed to be less critical for overall security.
- Extensive misuse description makes use of a customized use case template with a list of fields tailored for describing misuse behavior.
- Misuse involving intricate action sequences and alternate paths calls for extensive descriptions.

Attack Patterns in Misuse Case Generation

- An attack pattern is a blueprint for an exploit.
- Attack patterns are descriptions of common methods for exploiting software.
- An attack pattern description contains sufficient detail about how a specific type (such as buffer overflow, SQL injection) of attack is executed along with recommended methods of mitigating that attack.
- Attack patterns are extremely useful in generating valid abuse and misuse cases.
- Using a published catalog of attack patterns, select those attack patterns relevant to your system.
- Build misuse and abuse cases around the attack patterns selected for your system.

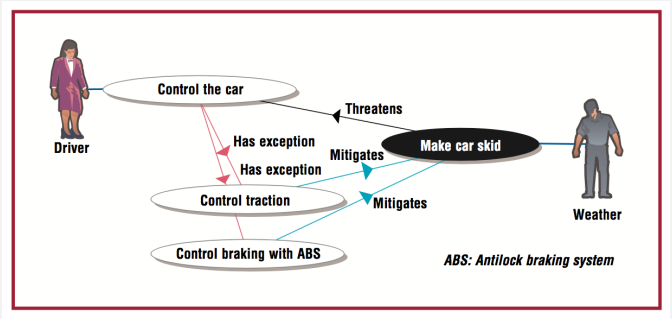
Misuse Cases and Nonfunctional Requirements

- Misuse cases can be viewed as defining nonfunctional requirements that help elicit funtional requirements for a system.



Misuse Cases and Nonfunctional Requirements

- Misuse cases can be used to define many types of nonfunctional or quality requirements such as reliability, maintainability, portability, safety, and so on.
- Eliciting safety requirements for a car through use and misuse cases



Use & Misuse Cases and Types of Requirements

- Applicability of use and misuse cases for eliciting different types of requirements

Applicability of use and misuse cases for eliciting different types of requirements

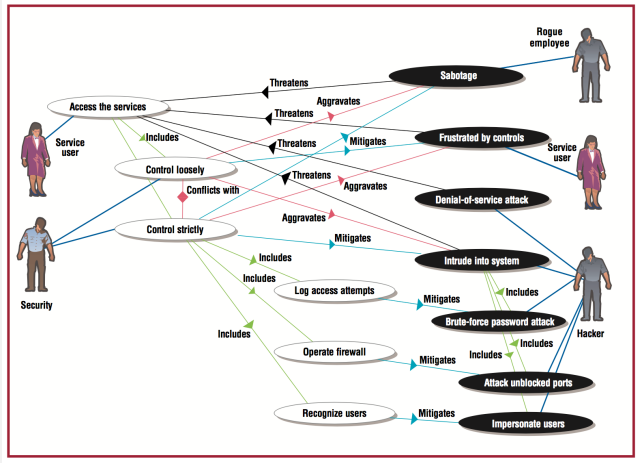
Elicitation through	Use case	Misuse case
Functional requirement	Important mechanism	Useful, but indirect
Nonfunctional requirement	Possible	Important mechanism

Generating Test Cases from Misuse Cases

- Good testing goes beyond happy-day scenarios to explore boundary and error condtions, exceptions, and inadvertent and intentional misuse and abuse.
- Thinking out negative scenarios is an essential skill for a test engineer.
- Misuse cases can be used to generate negative scenarios to ensure better system testing and to improve the quality of the delivered system.

Design Trade-Offs with Misuse Cases

- Misuse cases play a role during system design to help you consider and evaluate design alternatives and trade-offs.
- These design alternatives and trade-offs can be illustrated by adding **aggravates** and **conflicts with** relationships between cases.



Strengths and Weaknesses of Misuse Cases

- Strengths
  - Early focus on security by identifying security threats to define security requirements.
  - Use/misuse case diagrams link regular use cases to both threats (misuse cases) and potential countermeasures (security use cases) and helps in the prioritization of requirements. The real cost of implementing a use case includes the protection needed to mitigate all serious threats to it from misuses.
  - The links between use cases and misuse cases can support the tracing of security requirements to the threats that motivated them.

Strengths and Weaknesses of Misuse Cases

- Weaknesses
  - May lead to analysis paralysis when faced with potentially large number of threats.
  - Not equally suitable for all kinds of threats
    - Misuse is not always an identifiable sequence of actions
    - Misuser is not always identifiable
    - Misuse does not always exploit an identifiable sequence of actions

References

- Guttorm Sindre and Andreas L. Opdahl, "Eliciting security requirements with misuse cases", Requirements Engineering, Volume 10, Issue 1, January 2005.
- Ian Alexander, "Misuse Cases: Use Cases with Hostile Intent", IEEE Software, Volume 20, Issue 1, Jan/Feb 2003.
- Paco Hope, Gary McGraw, and Annie I. Anton, "Misuse and Abuse Cases: Getting Past the Positive", IEEE Security & Privacy, Volume 2, Issue 3, May/June 2004.