Lecture 5

Security Requirements—Cont.

Dr. Lotfi ben Othmane

13 November 2015

Vulnerabilities in the News

- Disovered vulnerabilities in WordPress plugins

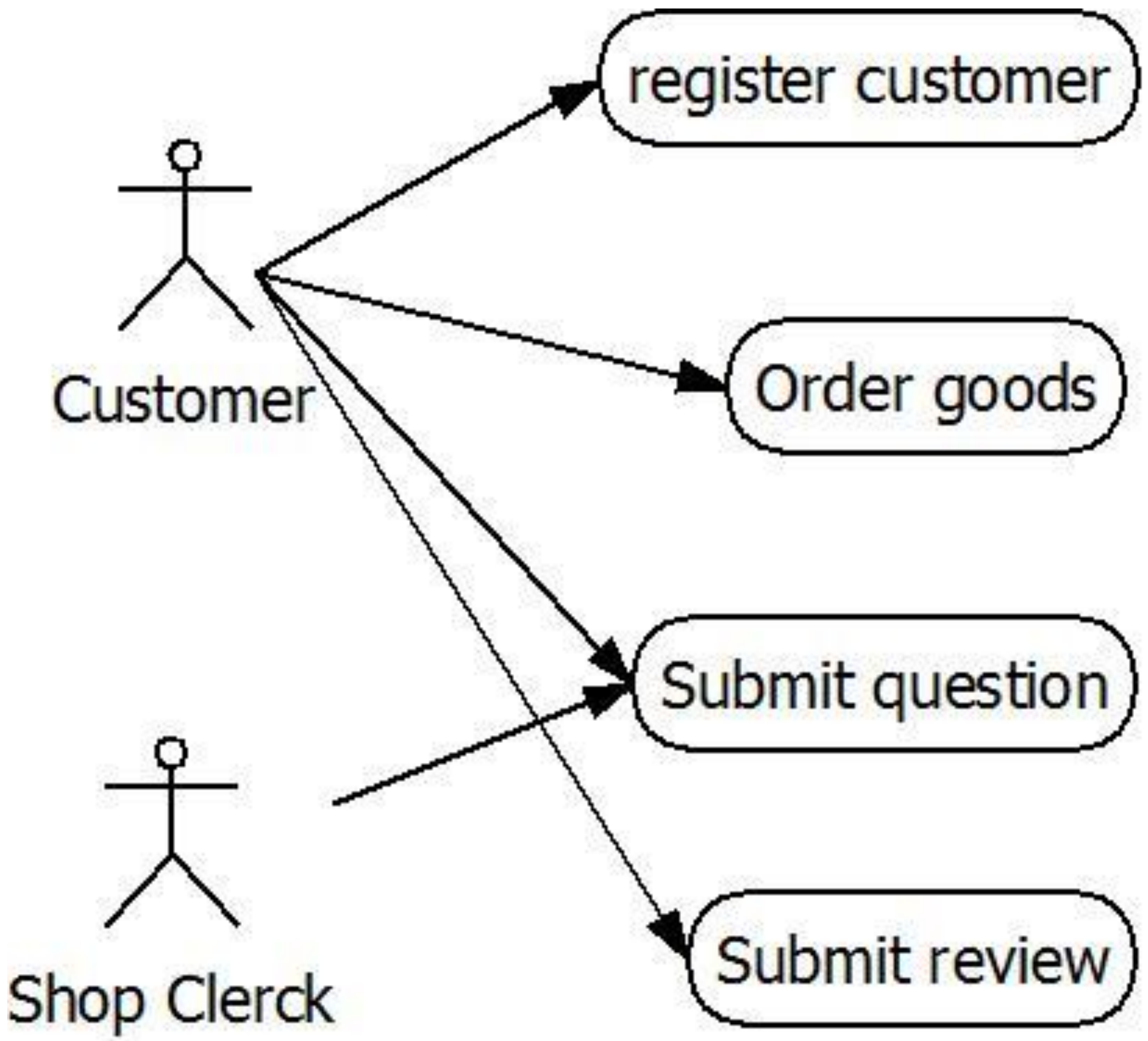
  by Mallikarjun Nuti

- SAMY worm

  by Omid Khodatars

Misuse Cases Security Requirements

- Use case is a sequence of actions, typically defining the interactions between a role and a system.
- Actor is a role played by a subject integrating with the system.



Misuse Cases Security Requirements

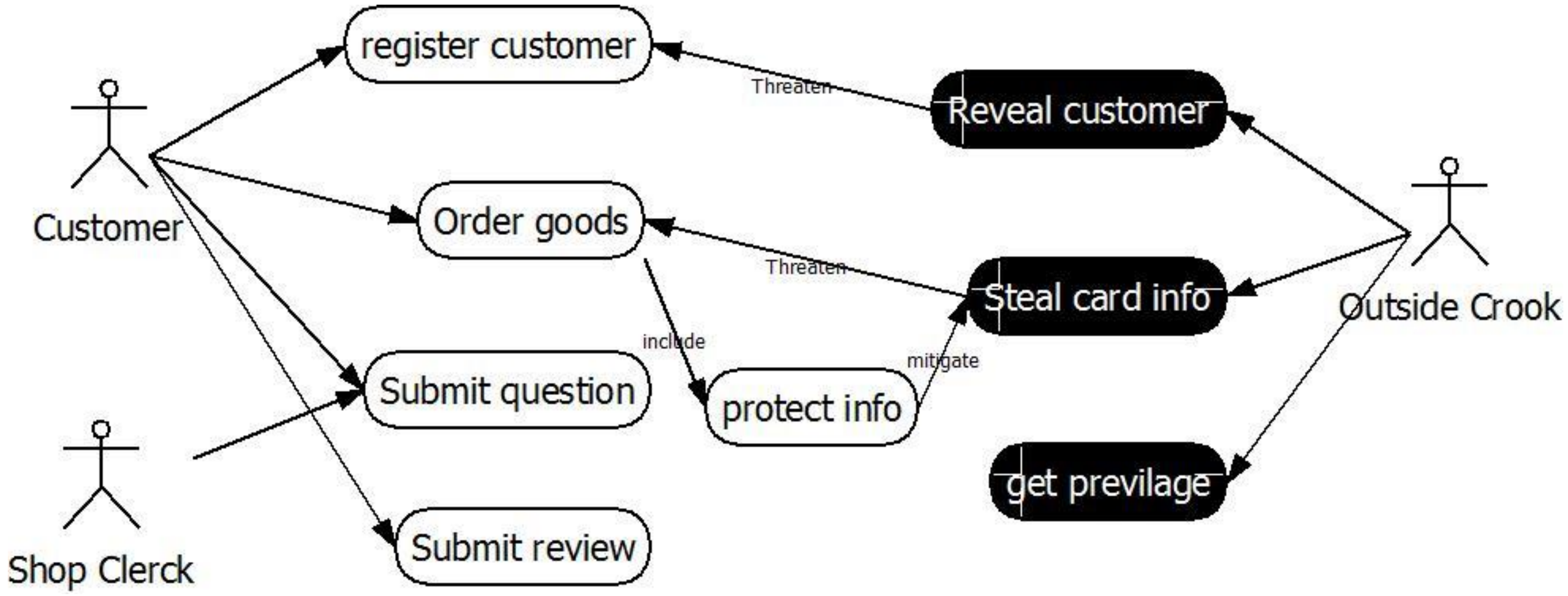
- Misuser is an actor that initiates misuse cases
- Misuse case is a sequence of actions that the system can perform causing harm to some stakeholders
- Misuse case threaten suse case implies the use case is exploited or hindered by the misuse case
- Use case mitigates misuse case implies the use case is a counter measure against the misuse case

SECURE SOFTWARE ENGINEERING GROUP

EC SPRIDE

Misuse Cases Security Requirements

Example



Misuse Cases Security Requirements

Process activities

1. Identify critical assets
2. Define security goals
3. Identify threats
4. Identify and analyze risks
5. Define security requirements (mitigation for the misuse)

   For the threats considering risk exposure and mitigation cost

Misuse Cases Security Requirements

Example

Misuse case name	Tamper with the database by Web query manipulation
Basic path	- The crook provides some values on the product search form and submit it - The system displays the products matching the query - The crook alters the URL query - The system sends error messages. The Crook gains information about the database - The crock alters the query to add, delete records as s/he whishes
Mitigation	- The system does not provide error information that could be used for misuse - The system validate the user input before querying the database

Incremental Security Requirements

- Security requirements have relationships
  - Authorization requirement requires authentication requirements
- Types include
  - Precondition/requires/constraints
  - Contradicts/conflicts
  - Similarity
  - Example for
  - Refines
- Incremental changes impacts security requirements
  - Add new dependencies
  - Replace or elaborate existing dependencies
  - Remove dependencies
- Security requirement dependencies shall be considered in incremental development

Recap

- Security requirements are constraints on the system. These constraints operationalize one or many security goals.
- Adequate requirements need to comply with the definition, incorporate assumptions about behavior, and could be satisfied.
- The goal-oriented security requirements approach operationalizes the security goals given the assets and functional requirements.
- In misuse case requirements approach, the requirements are the mitigations for the misuse cases.

Open Questions

- What are the main existing security risk elicitation methods? Why do we need many methods?
- How to trace the enforcement of security requirements?

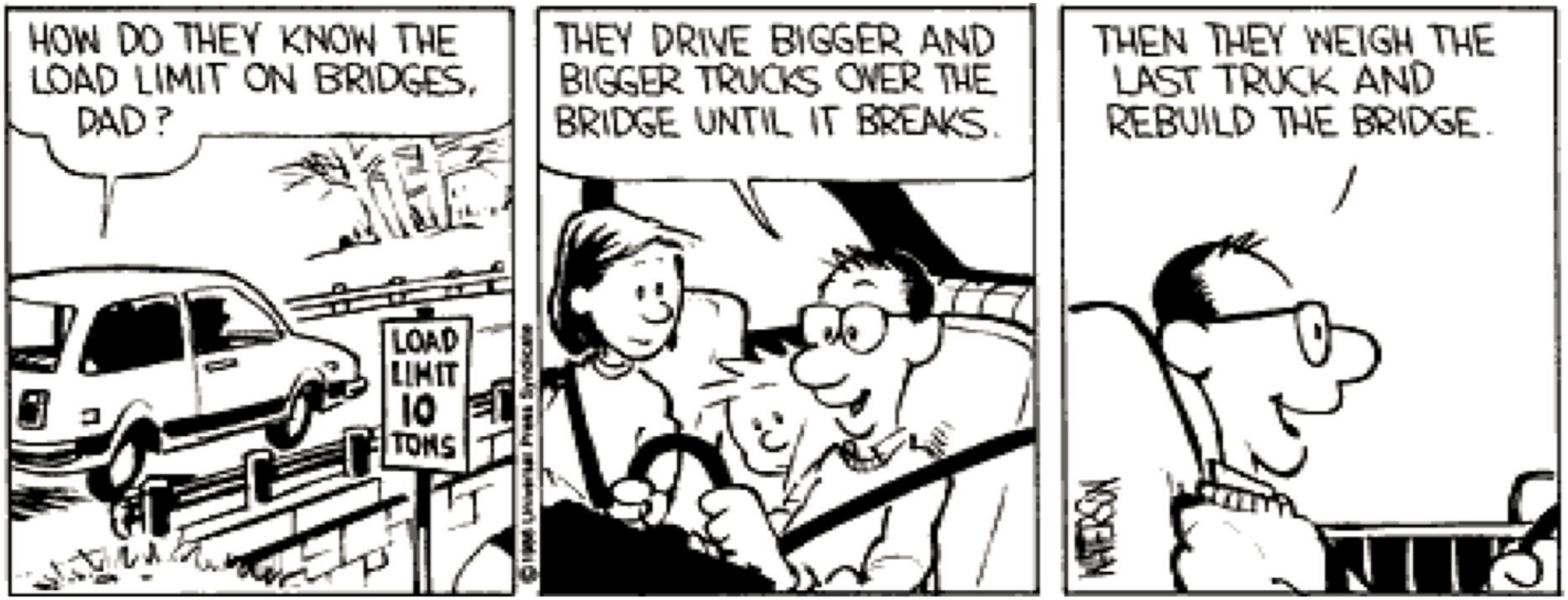
Lecture 5

Security Architecture

Philipp Holzinger
Dr. Lotfi ben Othmane

13 November 2015

What Is Software Architecture?


HOW DO THEY KNOW THE LOAD LIMIT ON BRIDGES, DAD?
LOAD LIMIT 10 TONS
THEY DRIVE BIGGER AND BIGGER TRUCKS OVER THE BRIDGE UNTIL IT BREAKS.
THEN THEY WEIGH THE LAST TRUCK AND REBUILD THE BRIDGE.
WATTERSON

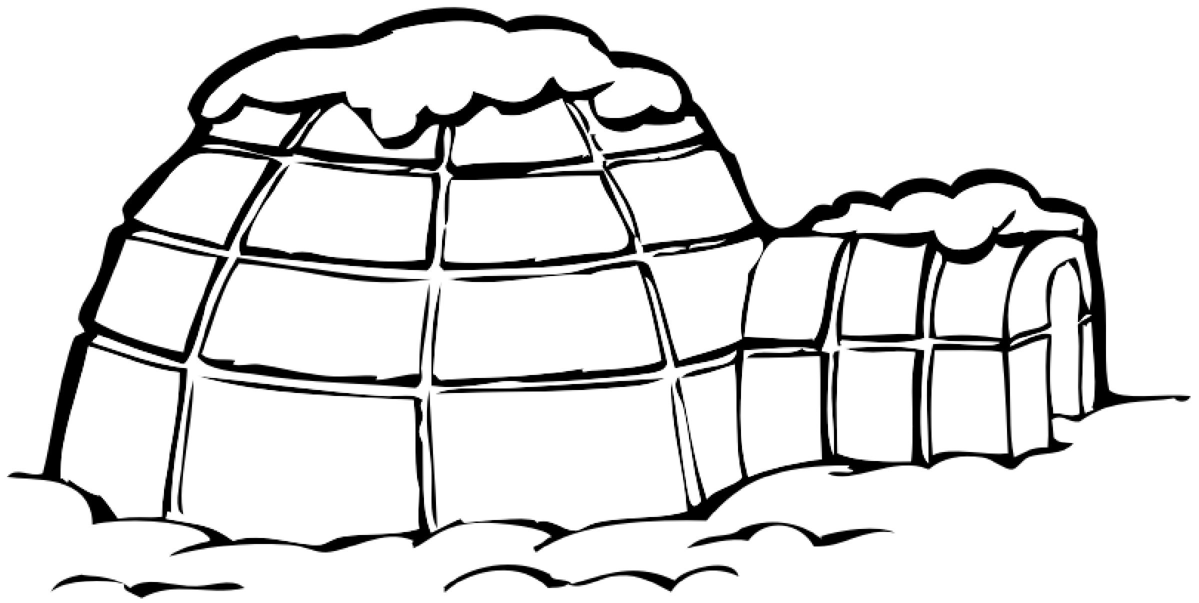
What is Software Architecture?

- “The organizational structure of a system or component.“
  IEEE Standard Glossary of Software Engineering Terminology

- “The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.”
  IEEE 1472

What is Software Architecture?

- Every software implements a software architecture
- Not all software architectures are documented



Source: https://openclipart.org/detail/1053/igloo

Software Architecture Documentation

Do we need it ?

Software Architecture Documentation

Do we need it ?

Yes, for…
…project planning
…risk management
…system evaluation
…maintanance

Software Complexity Is a Challenge

- We need views/perspectives/viewpoints
- Several approaches have been published
  - 4+1 View Model
  - IEEE 1471
  - …

4+1 View Model

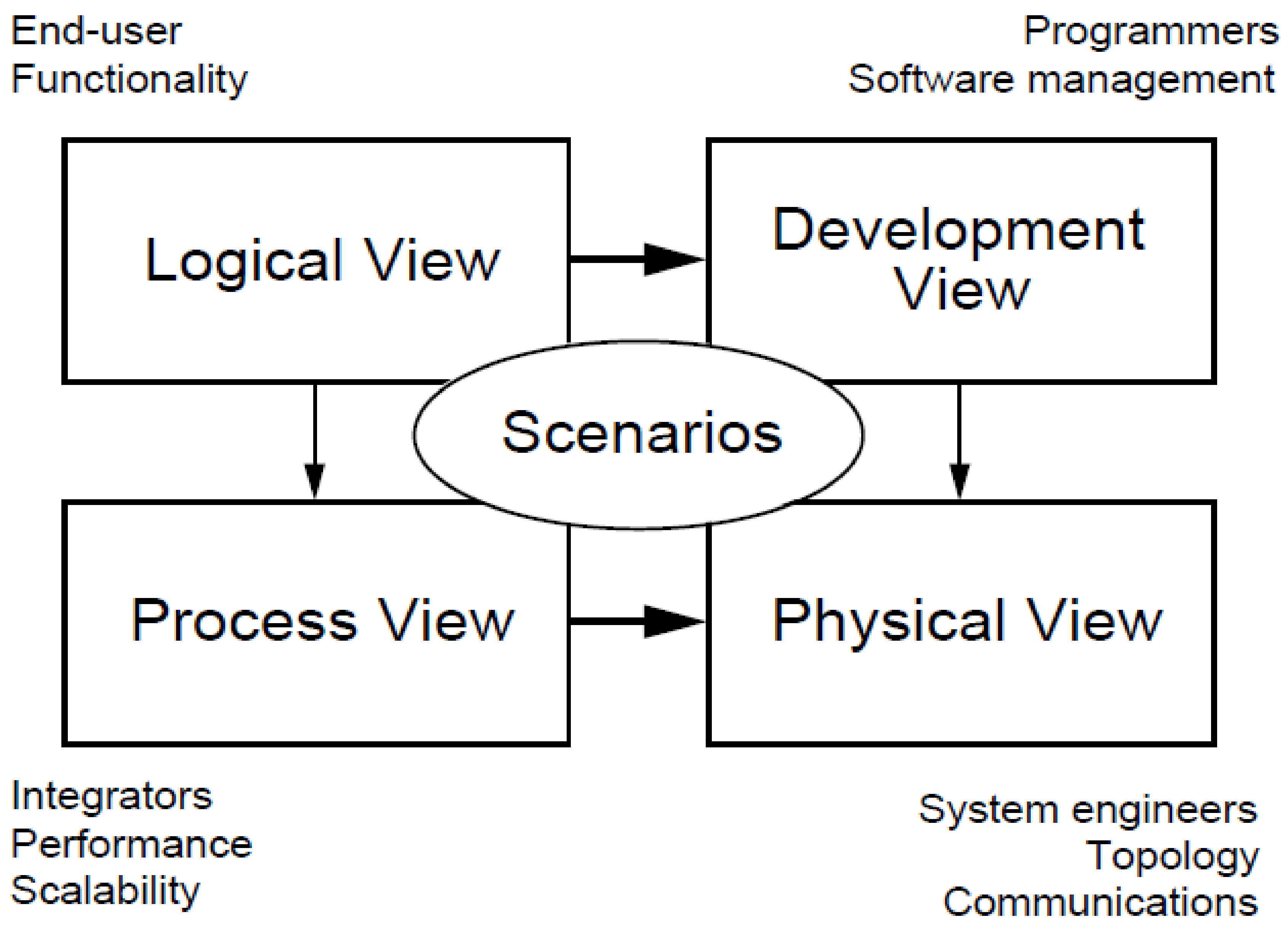
- „Architectural Blueprints – The ‚4+1' View Model of Software Architecture"
  Philippe Kruchten, IEEE Software 12 (6) Nov. 1995, pp. 42-50

- One of the most well-known approaches in software engineering

- Four views describe the system itself (solution), one view describes releveant use cases (problem)
  — Hence 4+1

- Each view addresses a set of stakeholders

4+1 View Model



Source: „Architectural Blueprints – The ‚4+1' View Model of Software Architecture" - Philippe Kruchten, IEEE Software 12 (6) Nov. 1995, pp. 42-50

4+1: Logical View

- Describes the features the system should provide to its users
- Decomposition of the system into objects or object classes
- Possible notations include
  - Class diagram
  - Sequence diagram

4+1: Development View

- Also known as implementation view
- Describes the organization of implementation artifacts (source files, make files, configurations, build scripts, etc.)
- Possible notations are
  - Package diagram
  - Component diagram

4+1: Process View

- Describes the behavior of the system at runtime
  - Processes, threads, and their communication
- Possible notation
  - Activity diagram

4+1: Physical View

- Also known as deployment view
- Describes how runnable entities of the system are to be deployed on physical machines
- Possible notation
  — Deployment diagram

4+1: Scenarios

- Also known as use case view
- It illustrates how architectural entities (objects, classes, processes, executable files) interact and interdepend to implement the relevant use cases
  - It connects the other four views and drives their design
- Facilitates evaluation of the architecture design

4+1 View Model in Security

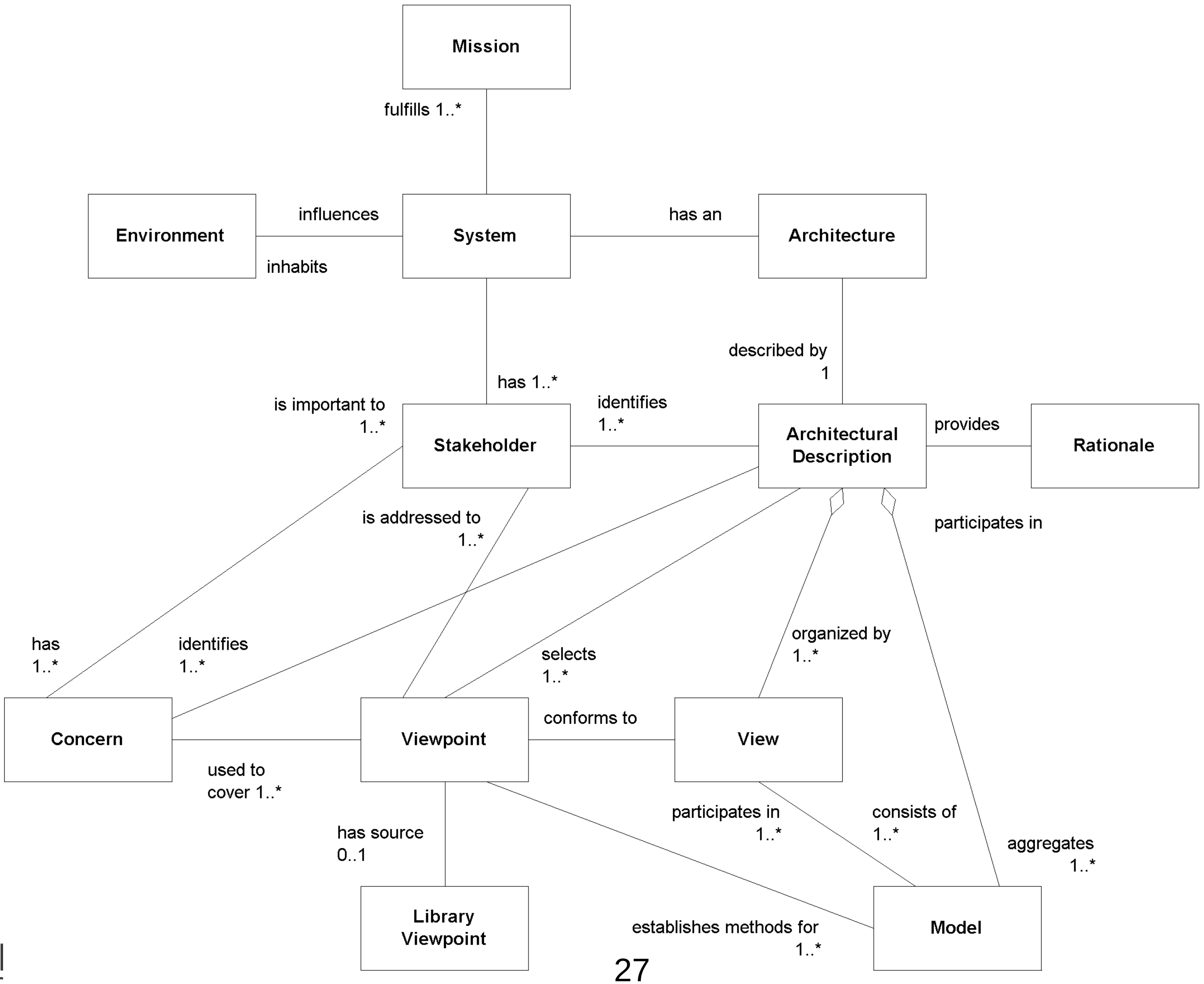
How well can we express security-relevant aspects of the system

?

IEEE 1471 - Overview

- IEEE-Std-1471-2000, „Recommended Practice for Architectural Description of Software-Intensive Systems“
- Describes how to document an architecture
- Defines a conceptual framework and vocabulary
- Describes a set of practices

IEEE 1471 – Conceptual Framework



IEEE 1471 – Conceptual Framework

- Every system implements an architecture
- Every architecture is described by one architectural description (AD)
- Every AD identifies
  - relevant stakeholders
  - architectural concerns
  - viewpoints
  - …

IEEE 1471 – Viewpoints

- A viewpoint comprises a view that addresses 1..* concerns of 1..* stakeholders
- A concern is an interest in the system that can be held by 1..* stakeholders
  - E.g., a security analyst may want to know how the system prevents XSS
- An AD is sufficiently complete, if it covers all concerns of all stakeholders

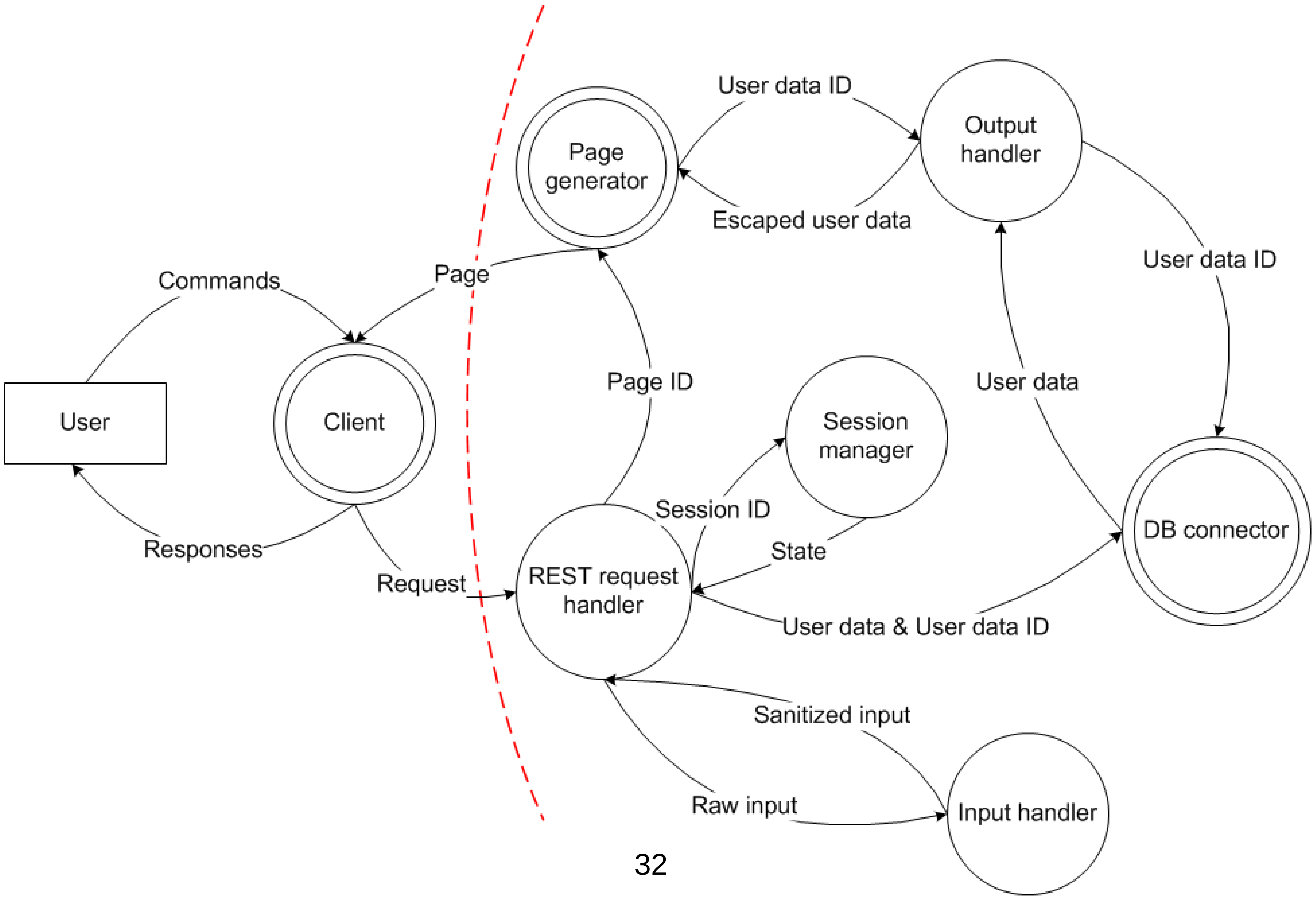
IEEE 1471 - View

- A view is a representation of the system
- It consists of 1..* models
  — UML, DFD, Misuse cases, etc.

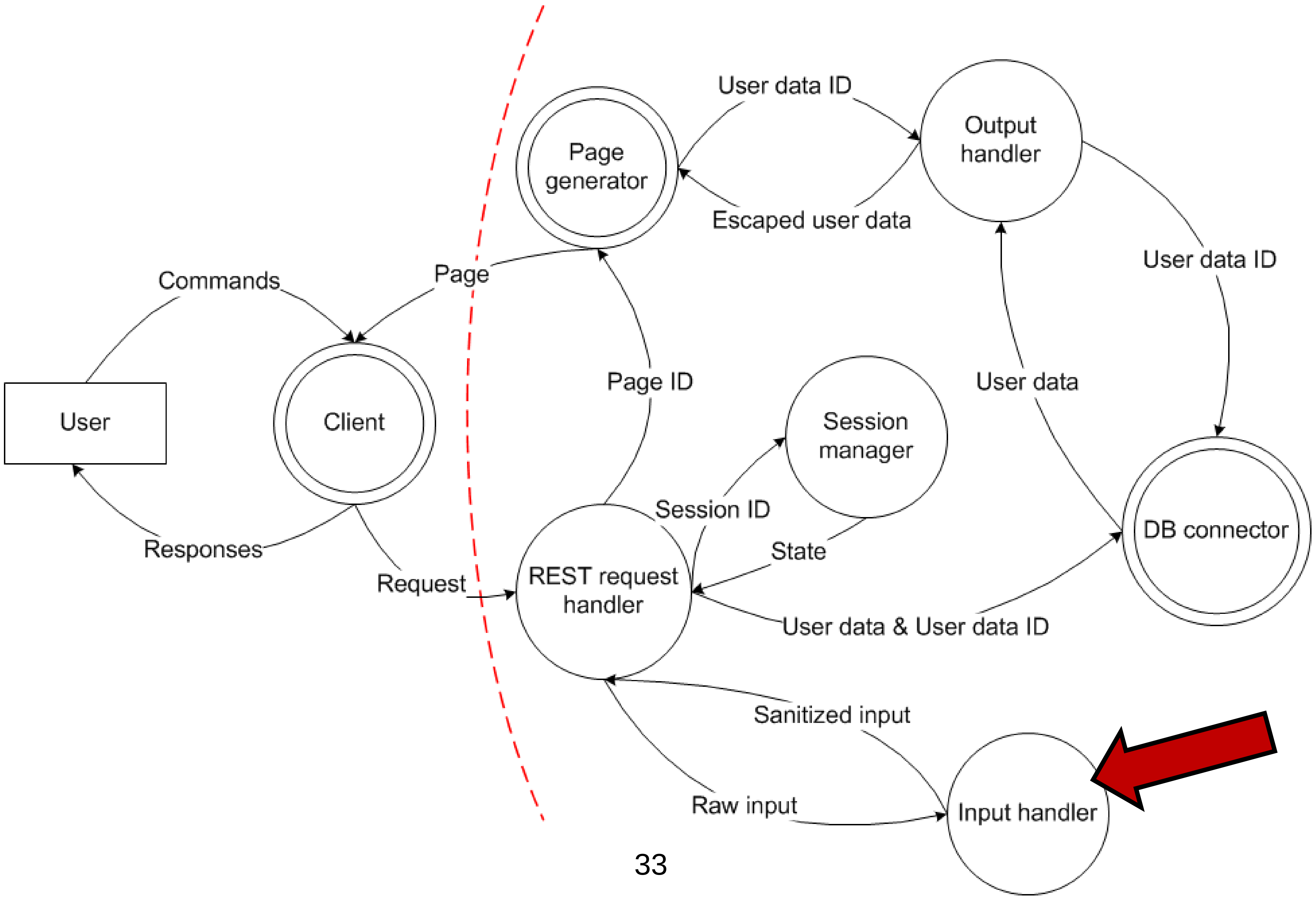
IEEE 1471 – Viewpoint Example

Viewpoint name	**Web security**
Stakeholder	Security analyst, developer
Concerns	How is XSS prevented?
Language/Modeling technique	Data flow diagram (DFD)
…	
Analysis technique	STRIDE
…	

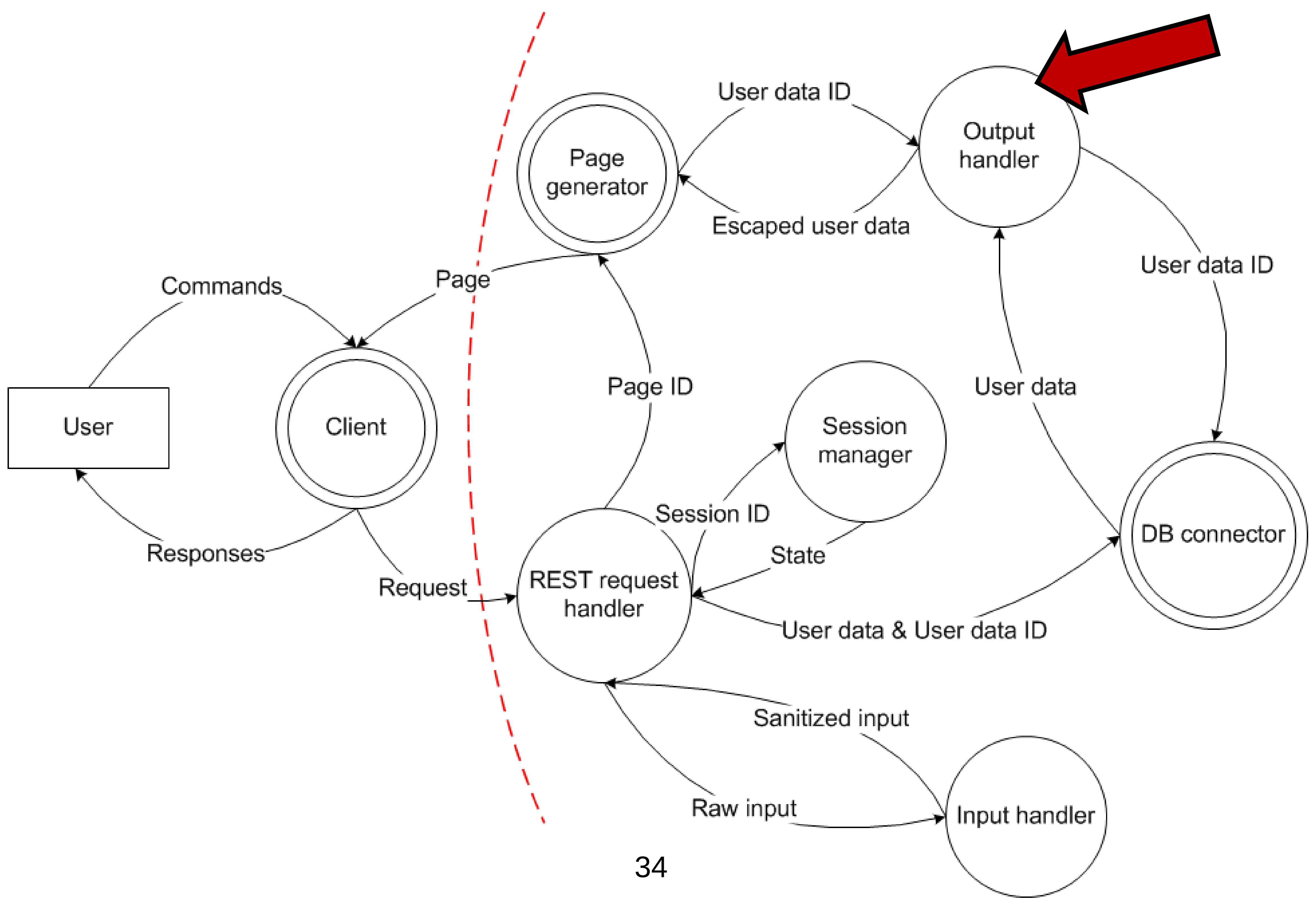
IEEE – View Example



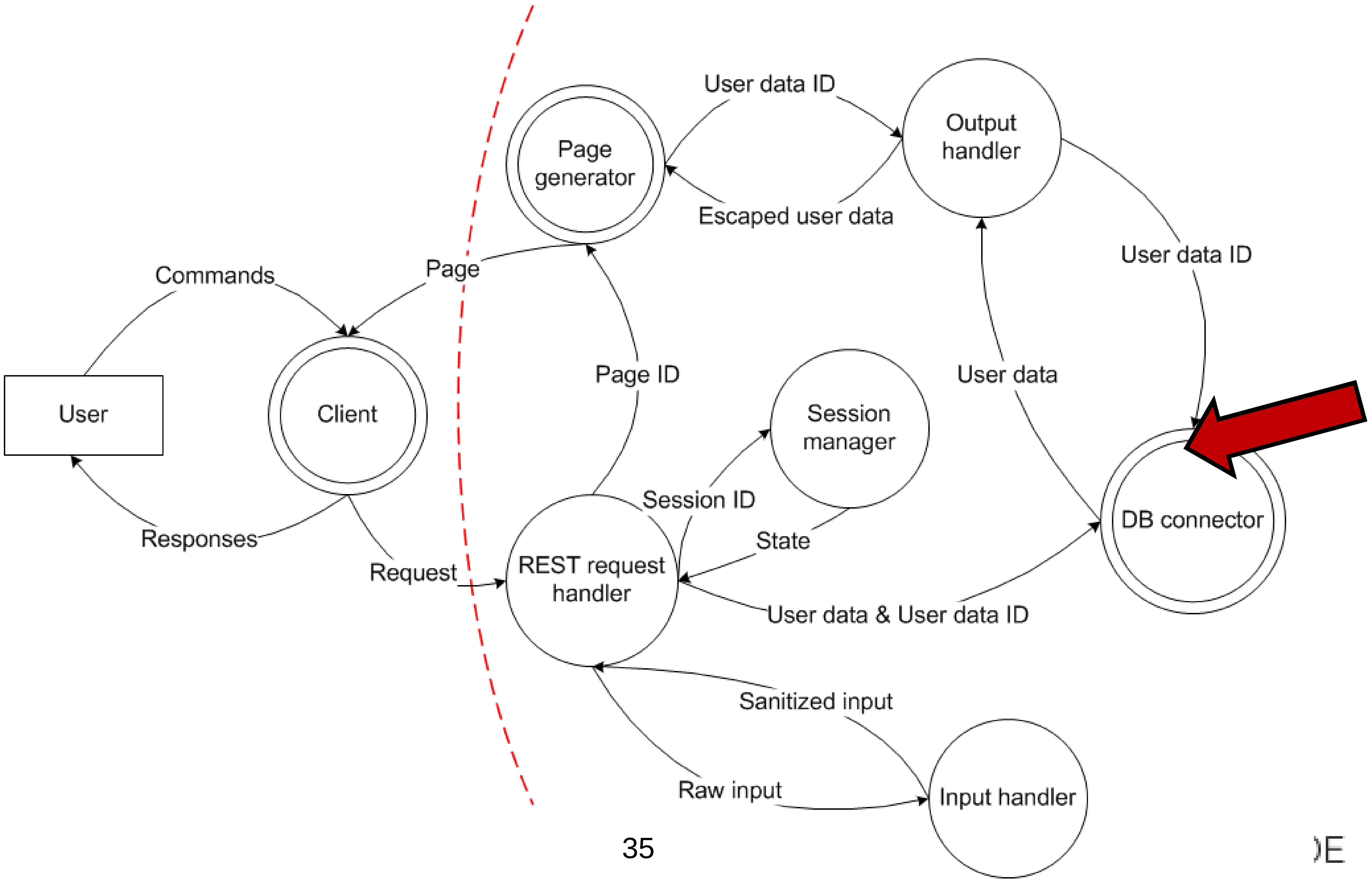
IEEE – View Example



IEEE – View Example



IEEE – View Example



Security Architecture Design Principles

- Defense in Depth
- Fail-safe defaults
- Least common mechanism
- Complete mediation
- Open Design
- Economy of mechanism
- Separation of privilege
- Psychological acceptability

Security Architecture Design Principles

Fail-safe defaults

- Decisions on permission rather than exclusion
- Protection scheme should identify conditions under which access is Permitted

Which one is correct?

```
DWORD dwRet =
IsAccessAllowed(...);
if (dwRet == ERROR) {
// Security check failed.
// Inform user that access is denied.
} else {
// Security check OK.
}
```

```
DWORD dwRet =
IsAccessAllowed(...);
if (dwRet == NO_ERROR) {
// Secure check OK.
// Perform task.
} else {
// Security check failed.
// Inform user that access
is denied.}
```

What is a Security Pattern?

A security pattern describes a particular recurring security problem that arises in specific contexts, and presents a well-proven generic solution for it.

The solution consists of a set of interacting roles that can be arranged into multiple concrete design structures, a s well as a process to create one particular such a structure.

SECURE SOFTWARE ENGINEERING GROUP

EC SPRIDE

Why Do We Need Security Patterns?

- Codify basic knowledge
- Share experience

Structure of Security Patterns

- Example
- Context
- Problem
- Solution
  - — Includes scope, e.g., #od users
- Consequences

Other information may be added such as implementation or lessons learned

Example 1- Password Design and Use

- Context - A password mechanism for authentication
- Problem - create, use, and manage password while they are accessible to owners and not to imposters
- Solution – Factors to consider in the design
  - Composition, length, and life time, etc.
  - Ownership, data entry, and authentication period, etc.
  - Distribution, storage, and transmission, etc.
- Consequences
  - Increase protection of passwords
  - Password guessing reduced

Example 2 - Single Access Point

- Example - Grant/Deny external access to a system after checking client rights
- Context - Provide external access to a system and ensure no misuse or damage by the client
- Problem – Multiple-part systems could be misused by complicated interactions
- Solution - Check access legitimacy based on given policy through a single access
- Consequences - simple implementation, no redundant authorization checks, cumbersome to use, single point of failure

Example 3 – Secure Channel

- Example – Transfer sensitive data between two parties through Internet
- Context – The system delivers functionalities and sensitive information to clients across the public internet
- Problem – How to ensure the protection of in-transit data through a public network is secure
- Solution – Create secure channels to obscure data in transit and ensure the client and sever exchange information to set a secure channel
- Consequences – Security is improved, scalability is potentially impacted, cost and maintenance overhead

Wrap up

- Every software implements an architecture
- Architectural documentation is needed to build, evaluate, and maintain complex software
- We cannot document the whole system in one representation to address all our needs
- Views and perspectives help stakeholders to focus on their individual concerns

Open Questions

- How to extract the security architecture from the code of the given software?
- How to verify the security architecture of a software given only the code?
- How to design a usable security description language?
- How to automate the application of security patterns?