Requirement Engineering

Summary of the previous lecture

- Project management
- Responsibilities/tasks of a Project manager
  - Planning
  - Risk management
  - People management
  - Reporting
  - Proposal writing
- Traditional vs. web project management

Outline

- **Introduction to RE**
- **RE basics**
- **Requirements specification**
- **RE process**
- **RE specifics in web engineering**

1. Requirement Engineering

- **Requirements Engineering: the principles, methods, & tools for drawing, describing, validating, and managing project goals and needs**
- **Given the complexity of Web apps, RE is a critical initial stage activity, but often poorly executed**

1. Requirement Engineering…

- **Costs:**
  - **Inadequate software architectures**
  - **"Unforeseen" problems**
    - **budget overruns**
    - **production delays**
    - **"that's not what I asked for"**
  - **Low user acceptance**

1. Requirement Engineering…

- **Why requirement engineering:**
  - **requirements don't define themselves (Bell & Thayer, 1976)**
  - **removal of mistakes post hoc is up to 200 times more costly (Boehm, 1981)**
  - **iterative collection and refinement is the most important function of a software engineer (Brooks, 1987)**

1. Requirement Engineering…

- **Why requirement engineering:**
  - **A study based on 340 companies in Austria, more than two thirds consider the SRS as the major problem in development process (1995)**
  - **A study on Web applications, 16% systems fully meet their requirement while 53% deployed systems do not (Cutter Consortium, 2000)**

1. Requirement Engineering…

- **Why requirement engineering:**
  - **A study among 8000 projects, 30% of projects fail before completion & almost half do not meet customer requirements (Standish group, 1994)**
    - **Unclear objectives, unrealistic schedules & expectations, poor user participation**

2. RE basics

- **Identify and involve the stakeholders**
  - **those that directly influence the requirements**
  - **customers, users, developers**
- **What are their expectations?**
  - **may be misaligned or in conflict**
  - **may be too narrowly focused or unrealistic**

2. RE basics…

- What is requirement?
- The descriptions of what the system should do
  - services that it provides and the constraints on its operation
- IEEE 601.12 definition of requirement:
  - 1) Solves a user's problem
  - 2) Must be met or possessed by the system to satisfy a formal agreement
  - 3) Documented representation of conditions in 1 and 2

2. RE basics…

- **Requirements types**
- **Functional requirements:**
  - **statement of services**
  - **how system reacts to input**
  - **how system behaves in particular situation**
- **Non-functional requirements:**
  - **constraints on services (timing, quality etc.)**
  - **applies as a whole**

2. RE basics…

- **Requirements are collected iteratively and change**
- **Keys to requirement definition:**
  - **Negotiation**
  - **Scenario-based discovery**
  - **Clear definition of context and constraints**

3. Requirements specifications

- process of writing down the user and system requirements in a requirements document
- User requirements (for users)
  - should be understand able to users
  - avoid notations, use simple tables, forms etc.
- System requirements (for Software engineers)
  - starting point for the system design
  - how system provides the services

3. Requirements specifications…

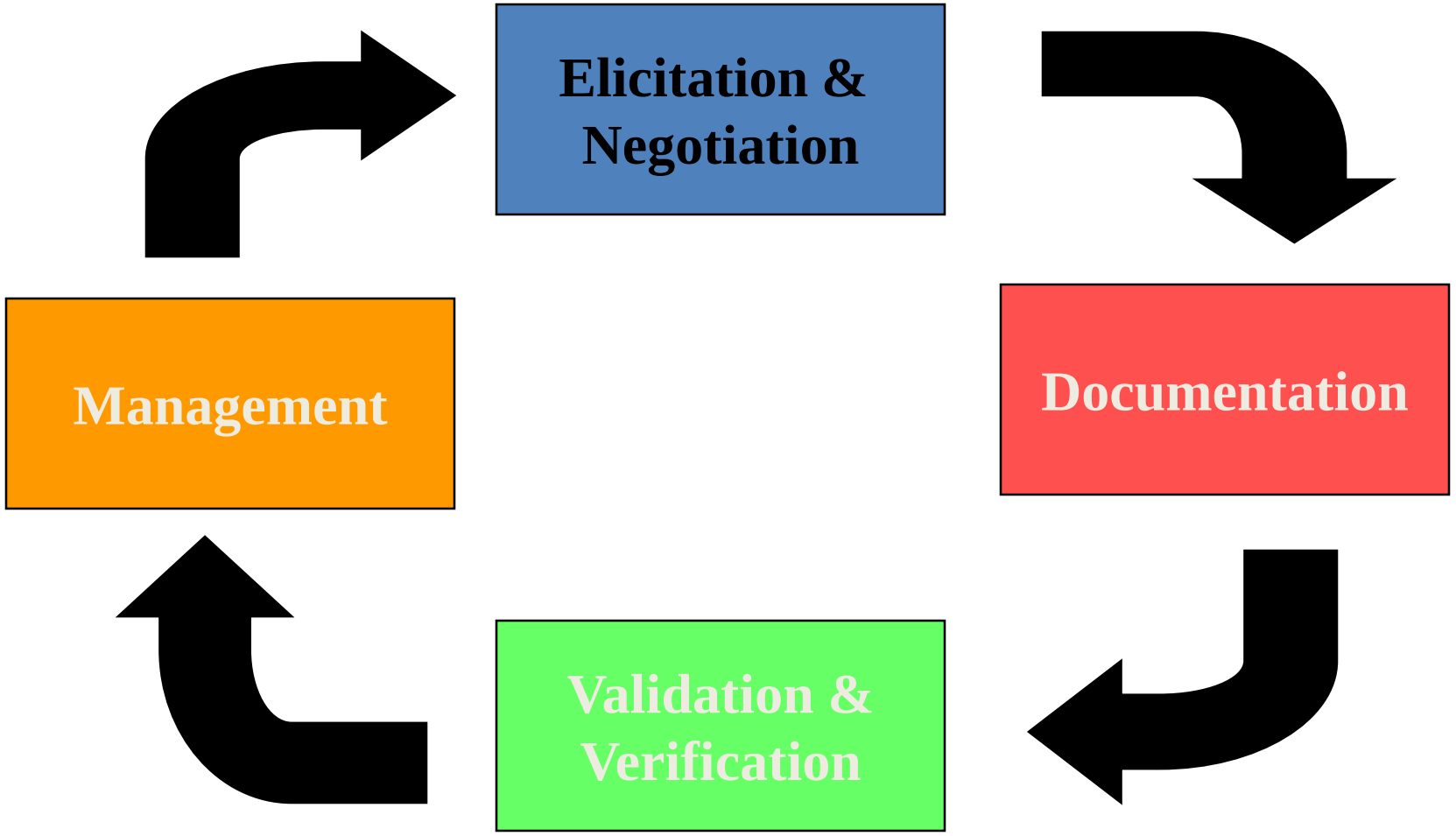
- **Natural language specification:**
- **Stories or itemized requirements**
  - **create a standard format**
  - **distinguish between mandatory and desirable requirements**
  - **don’t use the technical words**
  - **associate rationale with each requirement**

3. Requirements specifications…

- **Structured specification:**
- **Includes**
  - **description**
  - **inputs/outputs**
  - **description of the action**
  - **pre condition**
  - **post condition**

15

4. RE process



16

4. RE process…

- **Elicitation and negotiation:**
- **RE engineer involve the stakeholder to define**
  - **application domain**
  - **services**
  - **constraints**
- **Steps:**
  - **requirement discovery**
    - **Interviewing, scenarios, questionnaires, use-cases etc.**
  - **classification and organization**
  - **prioritization and negotiation**

4. RE process…

- **Documentation:**
  - **requirements are documented after consensus**
- **Requirement verification and validation:**
  - **validated: doing right things?**
  - **verification: doing things right?**

18

4. RE process…

- **Requirements management:**
- **understanding and controlling changes**
  - **problem analysis and change specification**
  - **change analysis and costing**
  - **change implementation**

19

5. RE specifics in web engineering

- **Distinguishing characteristics:**
- **Multidisciplinary:**
  - **experts from different disciplines i.e. media experts, content experts, usability experts etc.**
  - **challenging to achieve consensus**
- **Unavailability of stakeholders:**
  - **many stakeholders such as users are unknown during RE process**
  - **need to find suitable representatives**

5. RE specifics in web engineering

- **Distinguishing characteristics:**
- **Rapidly changing requirements & constraints:**
  - **environment is highly dynamic**
  - **harder to stabilize requirements**
- **Unpredictable operational environment:**
  - **impossible to control the operation environment**
  - **affects the quality requirements**
    - **change of bandwidth can change response time**

5. RE specifics in web engineering

- **Distinguishing characteristics:**
- **Legacy Systems:**
  - **constrained by existing system**
  - **existing components drive the possibilities**
- **Quality aspects:**
  - **are decisive i.e. performance, security, availability**
  - **harder to get exact specification**

5. RE specifics in web engineering

- **Distinguishing characteristics:**
- **User interface:**
  - **key success-critical aspect**
  - **should be aware of IKIWISI**
- **Quality of content:**
  - **accuracy, objectivity, credibility, relevance, actuality, completeness, or clarity**

5.1 RE principles for web engineering

- **Understanding the system context**
  - **web apps are always a component of a larger entity**
  - **why do we need the system?**
  - **how will people use it?**
- **Involving the stakeholders**
  - **get all groups involved**
  - **balance – one group's gain should not come at the expense of another**
  - **repeat the process of identifying, understanding and negotiating**

5.1 RE principles for web engineering…

- Iteratively define requirements
  - requirements need to be consistent with other system aspects (UI, content, test cases)
  - start with key requirements at a high level; basis for:
    - feasible architectures
    - key system use cases
    - initial plans for the project

5.1 RE principles for web engineering…

- **Risk Orientation**
  - **risk management is at the heart of the analysis process**
  - **what are the greatest risks?**
    - **integration issues / legacy systems**
    - **expected vs. actual system quality**
  - **how to mitigate risks?**
    - **prototyping (avoid IKIWISI)**
    - **show changes to customer iteratively**
    - **integrate existing systems sooner than later**

SUMMARY

- **Introduction to RE**
- **RE basics**
- **Requirements specification**
- **RE process**
- **RE specifics in web engineering**

References

- **Chapter 2, Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). Web Engineering, Hoboken, NJ: Wiley & Sons**
- **Chapter 4, Sommerville, Software Engineering, ISBN-10: 0-13-703515-2 , PEARSON**

Modeling web applications

Summary of the previous lecture

- **Introduction to RE**
- **RE basics**
- **Requirements specification**
- **RE process**
- **RE specifics in web engineering**

Outline

- **System modeling**
- **Modeling requirements**

1. System modeling

- **Process of developing abstract models of a system**
- **Representing system using graphical notations**
  - **UML**

1. System modeling

- each model presents a different view or perspective of the system
  - External perspective: system context and environment
  - Interaction perspective: how system interact with environment
  - Structural perspective: how system is organized
  - Behavioral perspective: dynamic behavior of the system

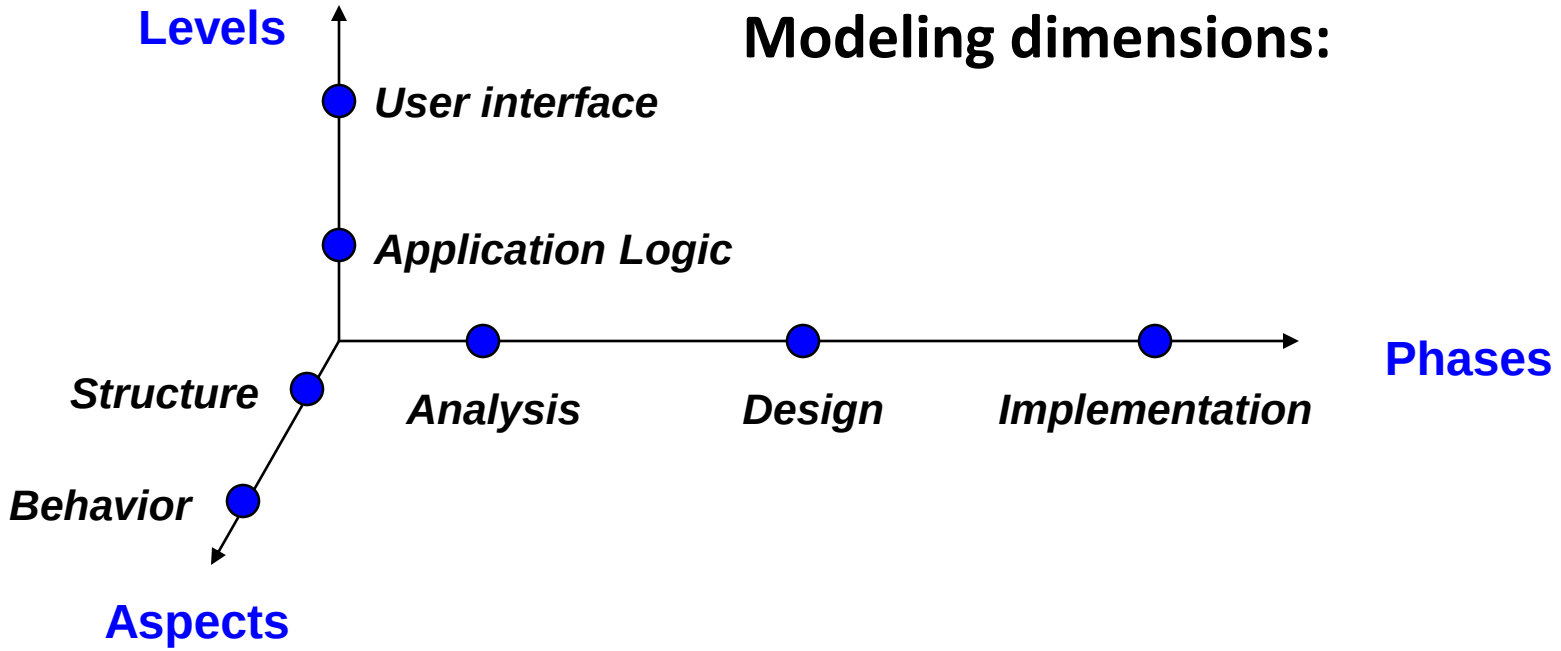
1. System modeling…

- Models are used during
  - RE phase to derive system requirements
    - use-case diagram, activity diagram
  - design phase to describe the system to engineers
    - class diagrams, sequence diagrams etc.
  - after implementation
    - to document system's structure and operation

1. System modeling…

- **Why system modeling?**
  - **reduce complexity**
  - **document design decisions**
  - **facilitate communication among team members**

1. System modeling…

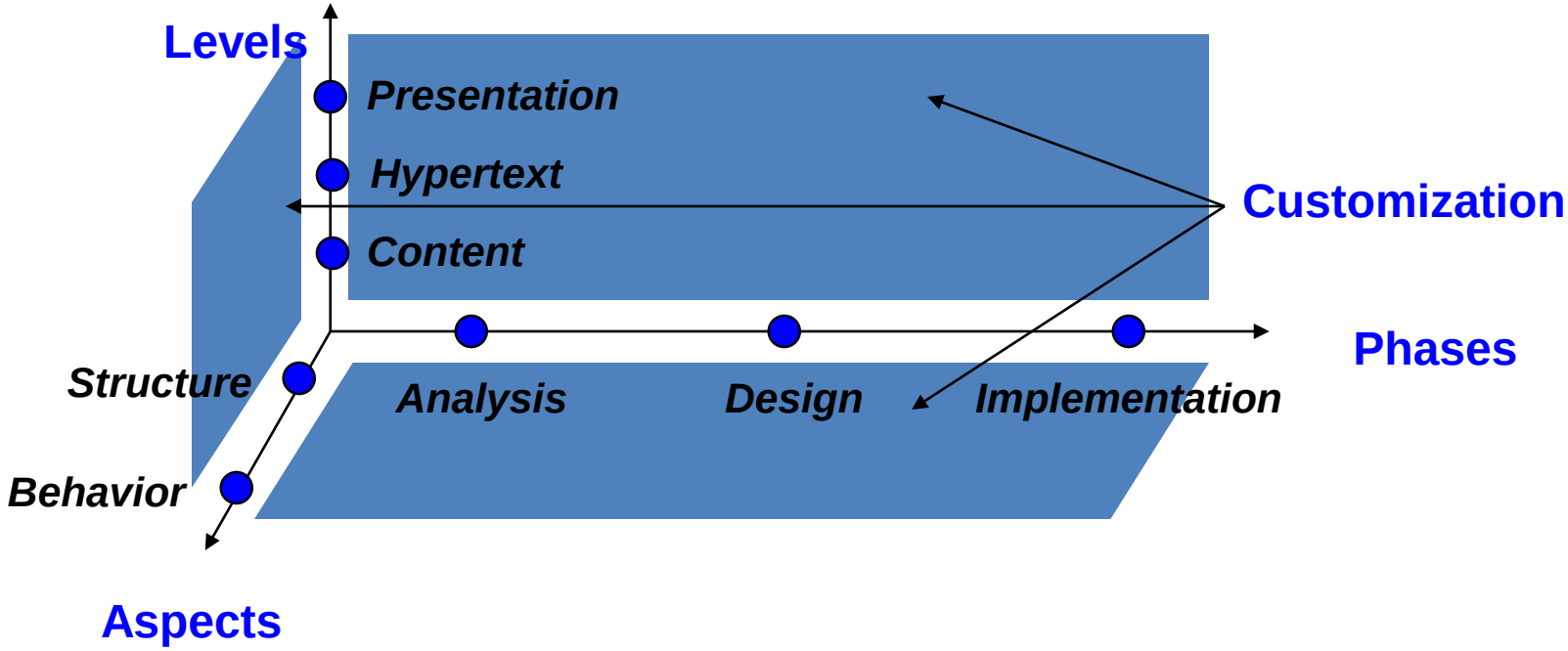


- Levels – the "how" & "what" of an application
- Aspects – objects, attributes, and relationships; function & processes
- Phases – Development cycle

1. System modeling…

- **"The Unified Modeling Language is a visual language for specifying and documenting the artifacts of systems"**
  - **Structural – Class diagrams**
  - **Behavioral – Use Case diagrams, State machine diagrams**

1. System modeling…



- Levels – Information, node/link structure, UI & page layout separate.
- Aspects – Same as Software Applications
- Phases – Approach depends upon type of application
- Customization – Context information (user's preferences, bandwidth restriction, device characteristic etc.) and allow to adopt web application accordingly
- Influence other three dimensions

1. System modeling…

- **Requirement modeling**
  - **use-case diagram**
  - **activity diagram**
- **Content modeling**
  - **class diagram**
- **Navigational modeling**
  - **to model nodes and navigational structure among them**
- **Presentation modeling**
  - **model user interface, page-layout**

1. System modeling…

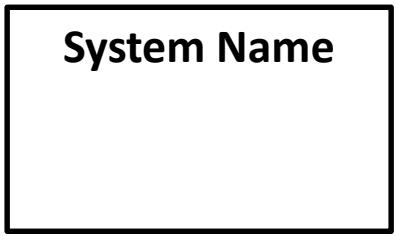
- **For Web-centric modeling, UML is used with some extensions from UWE (UML-based web engineering)**
- **http://uwe.pst.ifi.lmu.de/**

2. Modeling requirements

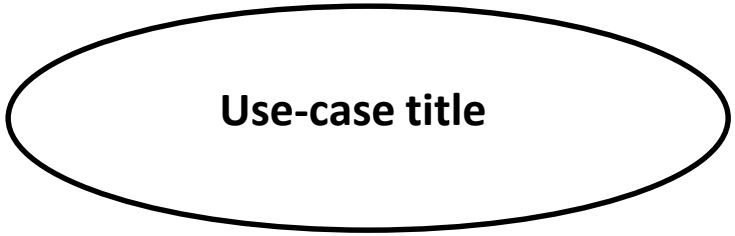
- **Use-case Diagram:** **The goal of the diagram is to provide a high-level explanation of the relationship between the system and the outside world (set goals)**
- **Activity diagram:** **a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency**

2.1 Use-case diagram

- **Components:**
- **The system**

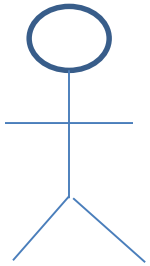


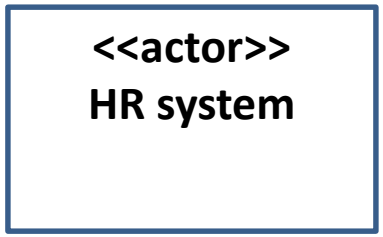
- **The use case task referred to as the use case that represents a feature needed in a software system**



2.1 Use-case diagram

- **Components:**
- **The actor(s) who trigger the use case to activate**

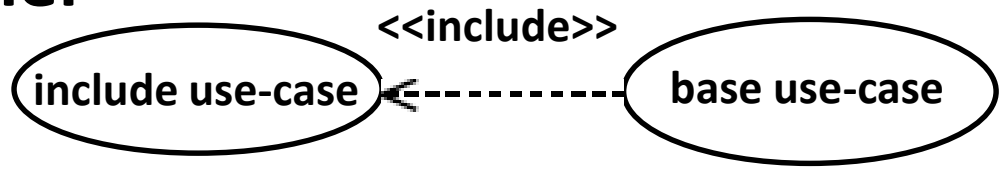




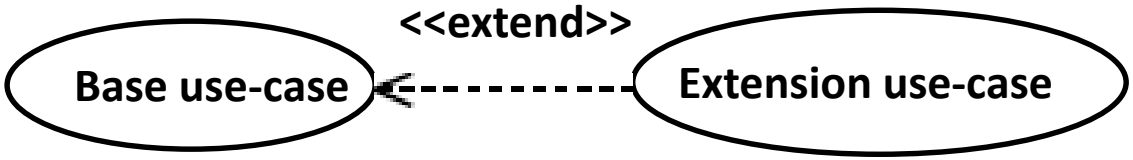
- **The communication line to show how the actors communicate with the use case**

2.1 Use-case diagram…

- **The include relationship represents the inclusion of the functionality of one use case within another**

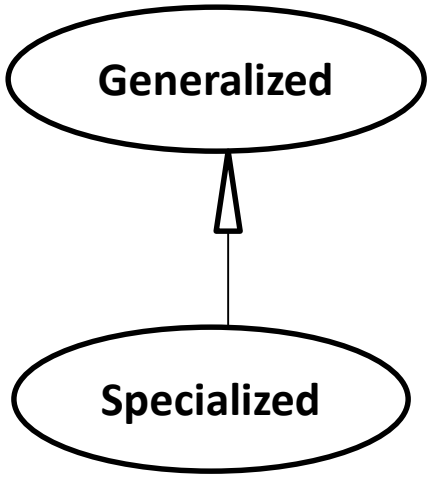


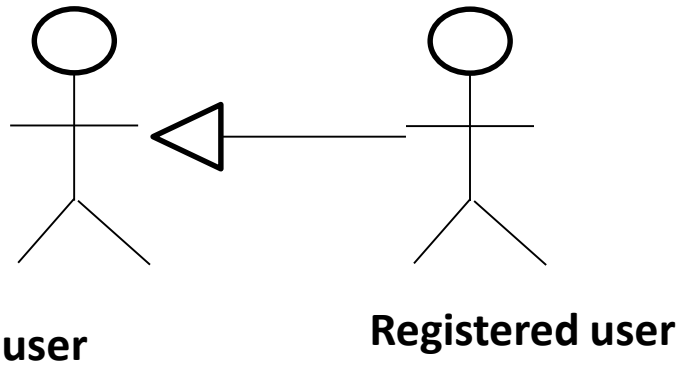
- **The extend relationship represents the extension of the use case to include optional functionality**



2.1 Use-case diagram…

- **A use-case-generalization is a relationship from a child use case to a parent use case, specifying how a child can specialize all behavior and characteristics described for the parent**





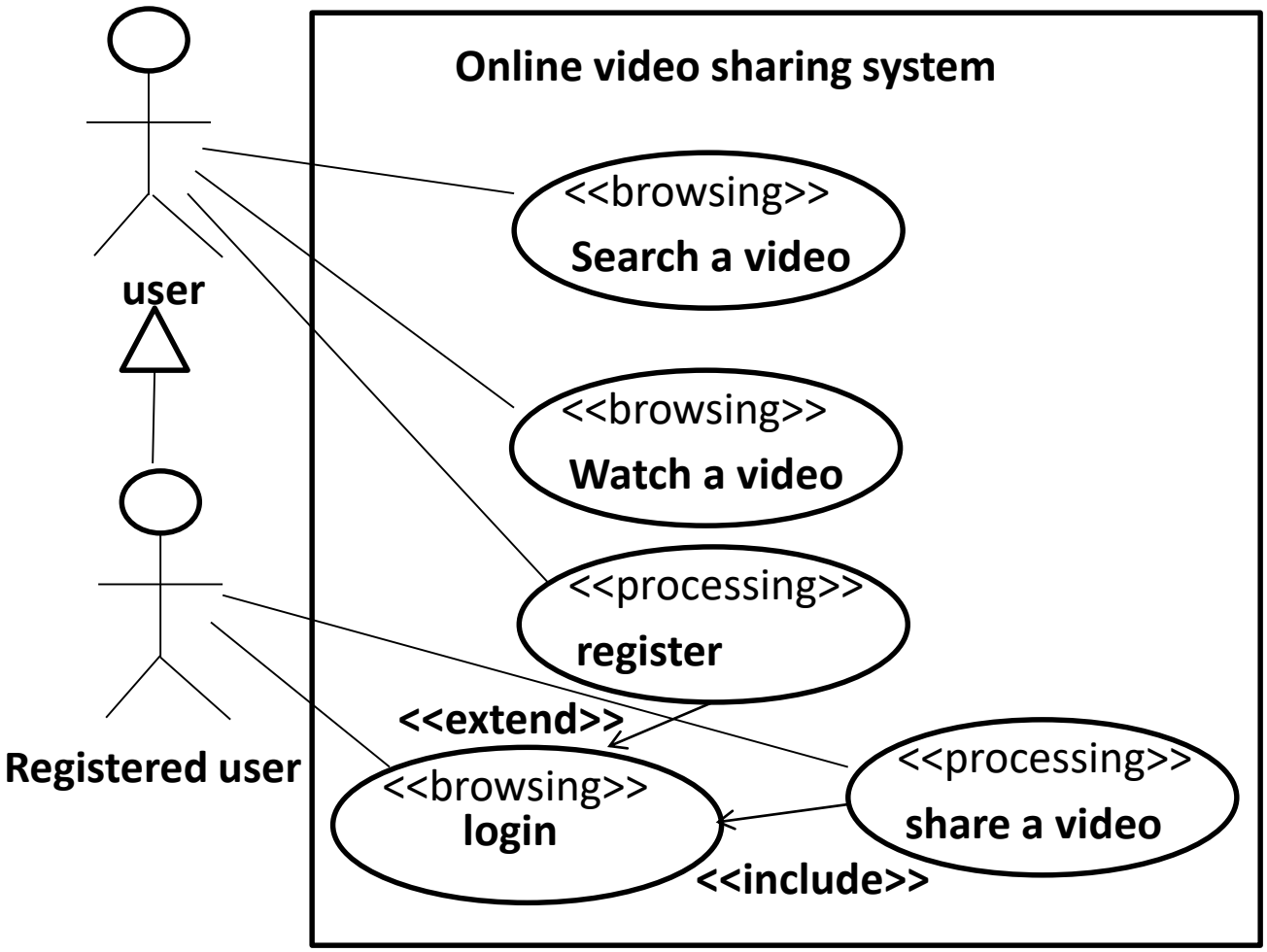
2.1 Use-case diagram…

- **Web specific requirements:**
- **Need to distinguish between functional and navigational use-cases**
  - **UWE provides <<browsing>> to represent a navigational use-case while <<processing>> to represent a functional use-case**

2.1 Use-case diagram…

- **Consider an online video sharing system:**
  - **Users can search and view the videos**
  - **A user must be a register user to share videos**

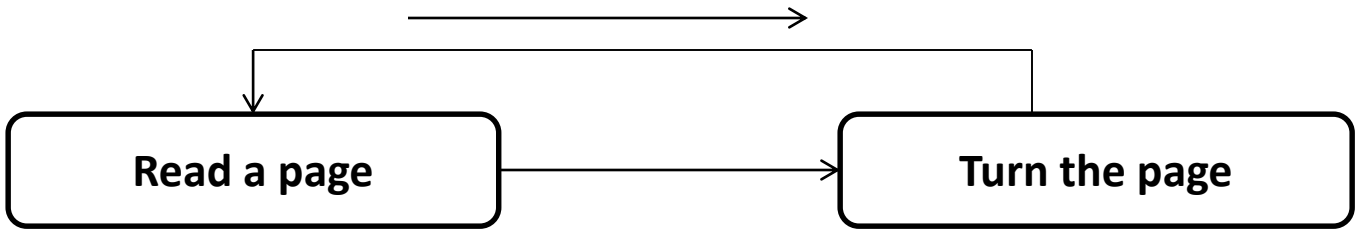
2.1 Use-case diagram…



2. The activity diagram

- **Elements of an activity diagram:**
- **An activity is a step in a process where some work is getting done**

- **The transition takes place because the activity is completed**



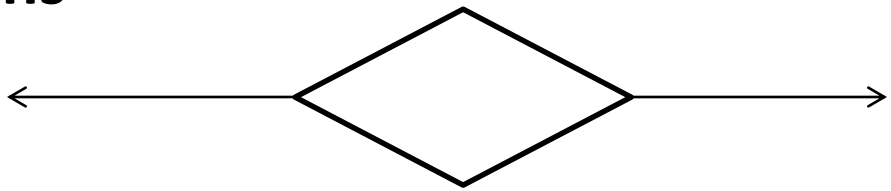
2. The activity diagram

- **Elements of an activity diagram:**

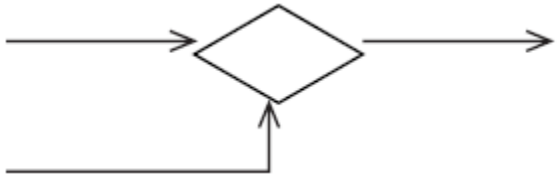
- **A guard condition can be assigned to a transition to restrict use of the transition**

2. The activity diagram…

- **Decisions**

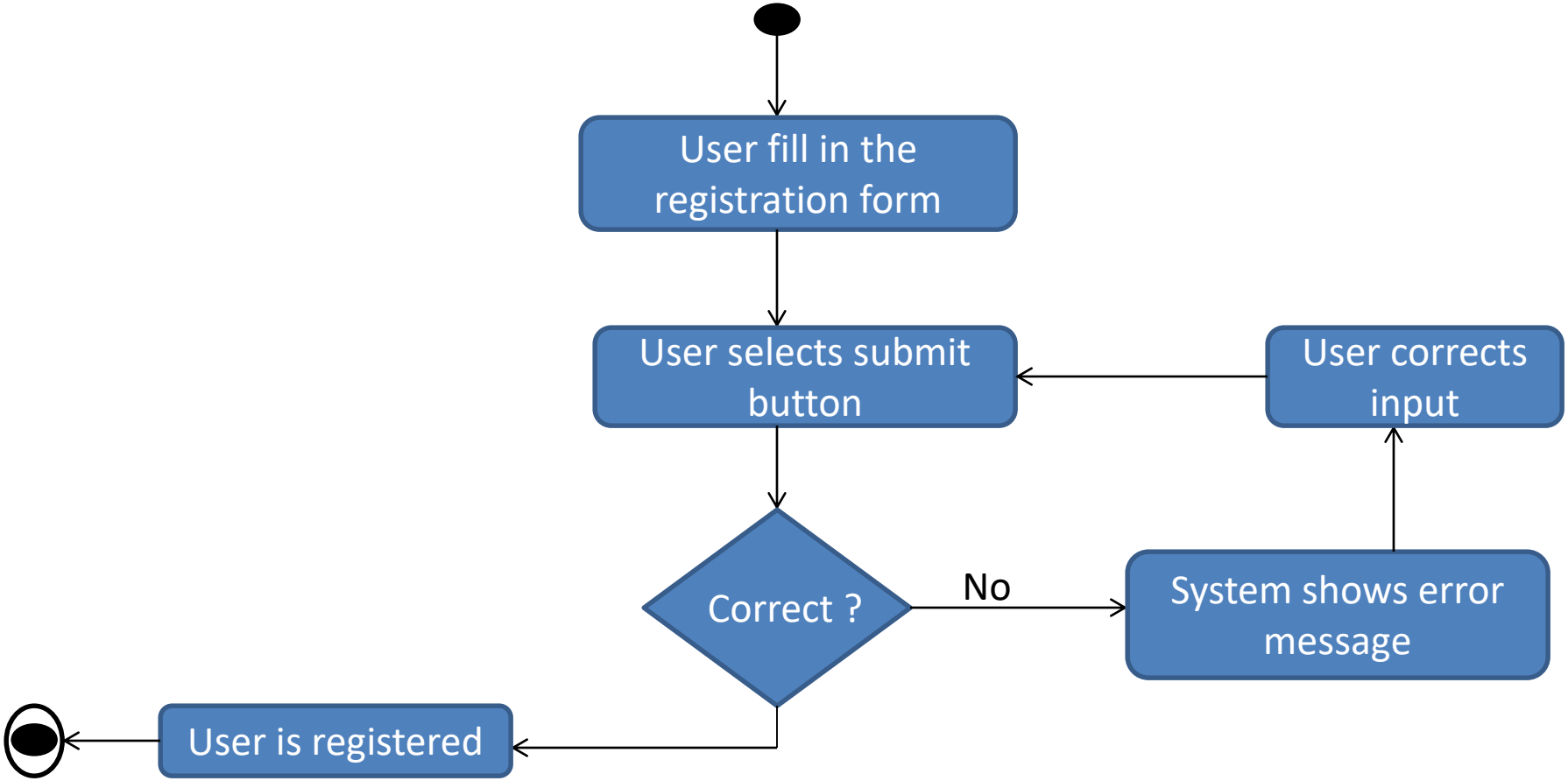


- **Merge point**

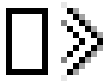


- **Start and end**

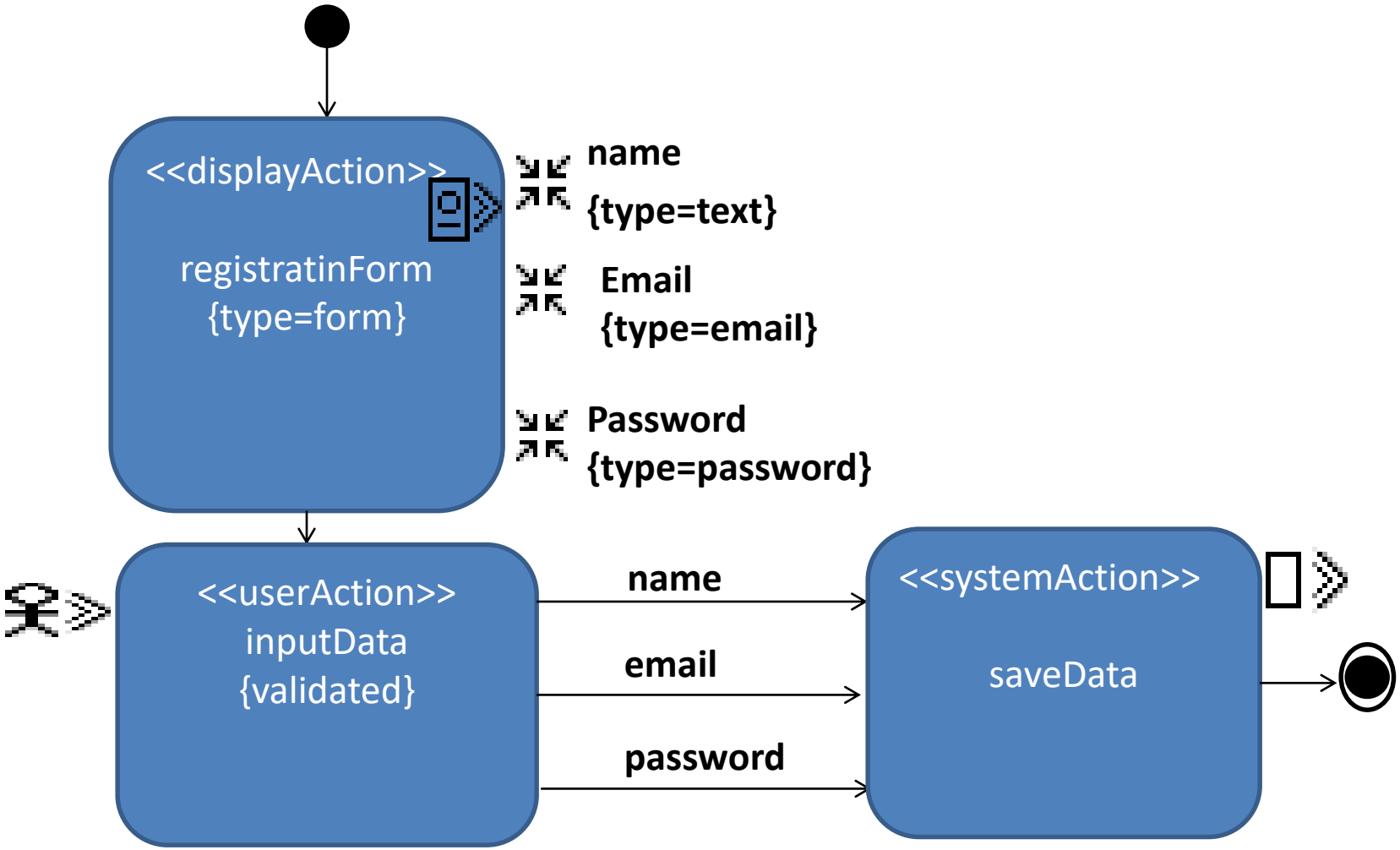
2. The activity diagram…



2. The activity diagram…

- **UWE activity diagram elements:**
- **userAction**  : user's action or response
- **systemAction**  : system's action
- **displayAction**  : display action
- **navigationAction**  : navigation
- **displayPin**  : output
- **interactionPin**  : input

2. The activity diagram…



Summary

- **System modeling**
- **Modeling Requirement**
  - **use-case diagram**
  - **activity diagram**

References

- **Chapter 3, Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). Web Engineering, Hoboken, NJ: Wiley & Sons**
- **Chapter 5, Sommerville, Software Engineering, ISBN-10: 0-13-703515-2 , PEARSON**

Modeling web applications

Summary of the previous lecture

- **System modeling**
- **Requirement Modeling**
  - **use-case diagram**
  - **activity diagram**

Outline

- Requirement modeling
  - use-case diagram
  - activity diagram
- Content modeling
- Navigation modeling
- Presentation modeling

1. Content modeling

- The information provided by a web application is one of the most important factors for the success of that application
- Content modeling aims at modeling the information requirements of a web application
  - diagraming the structural and behavioral aspects of the information
  - ignores the navigational information

1. Content modeling

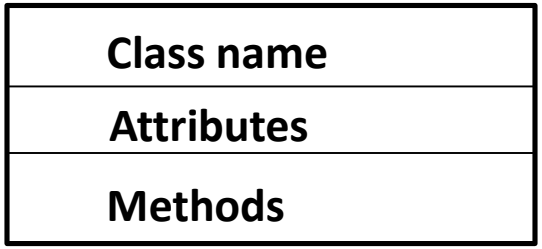
- **Key models**
  - **Class diagram: to model the structural aspects of information**
  - **State machine diagram: to model behavioral aspects of information**

1.1 Class diagram

- **Class diagram describes the structure of a system by**
  - **system's classes**
  - **class attributes**
  - **operations (methods)**
  - **relationship among objects**

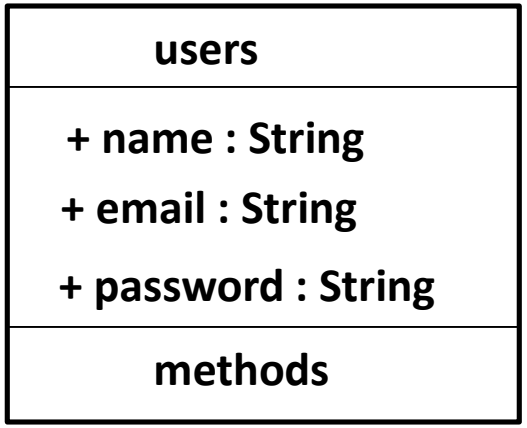
1.1 Class diagram…

- **Elements of a class diagram:**
- **class:**
  - **class is represented by a rectangle with three compartments**
    - **name**
    - **attributes**
    - **methods**



1.1 Class diagram…

- **Elements of a class diagram:**
- **Adding attributes:**
  - **an attribute describes a piece of information that an object owns**
    - **specified by name**
    - **kind (data type)**
    - **visibility (+, - , #)**
    - **default value**
    - **visibility name : type= default value**
      - **+ name : string = 'ali' {maximum 25 characters}**



1.1 Class diagram…

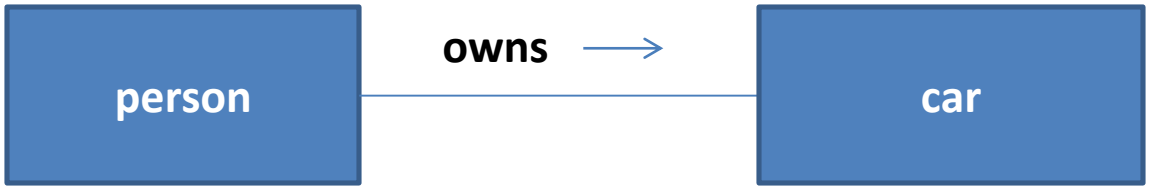
- **Elements of a class diagram:**
- **Adding methods (functions):**
  - **behaviors (things objects can do or can be done with them)**
    - **name**
    - **arguments**
    - **visibility (+, - , #)**
    - **return value**

users
attributes
- register(name:string, email:string,password:string):bool - login(email:string, password:string):bool

    - **visibility name (argument_name:type): return_value**
      - **+ userLogin(email:string, password:string):null**

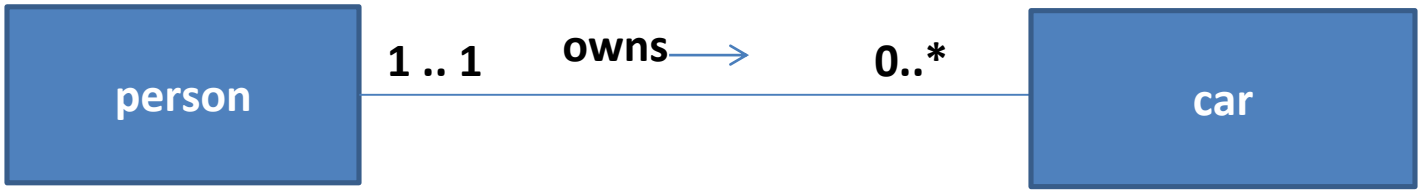
1.1 Class diagram…

- **Elements of a class diagram:**
- **Association**
  - **relationship between classes**
    - **name of relationship**
    - **direction of relationship**



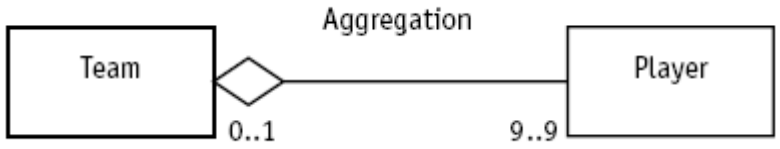
1.1 Class diagram…

- **Elements of a class diagram:**
- **Association multiplicity**
  - **How many objects participating in the relation**



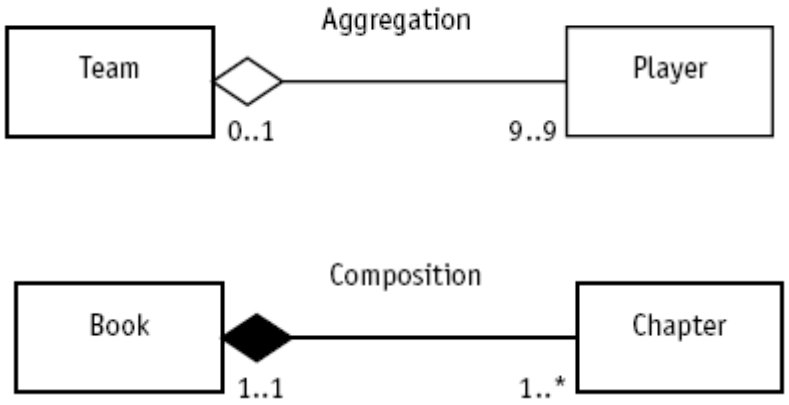
1.1 Class diagram…

- **Elements of a class diagram:**
- **Aggregation relation**
  - **class has features of another class plus some own features**

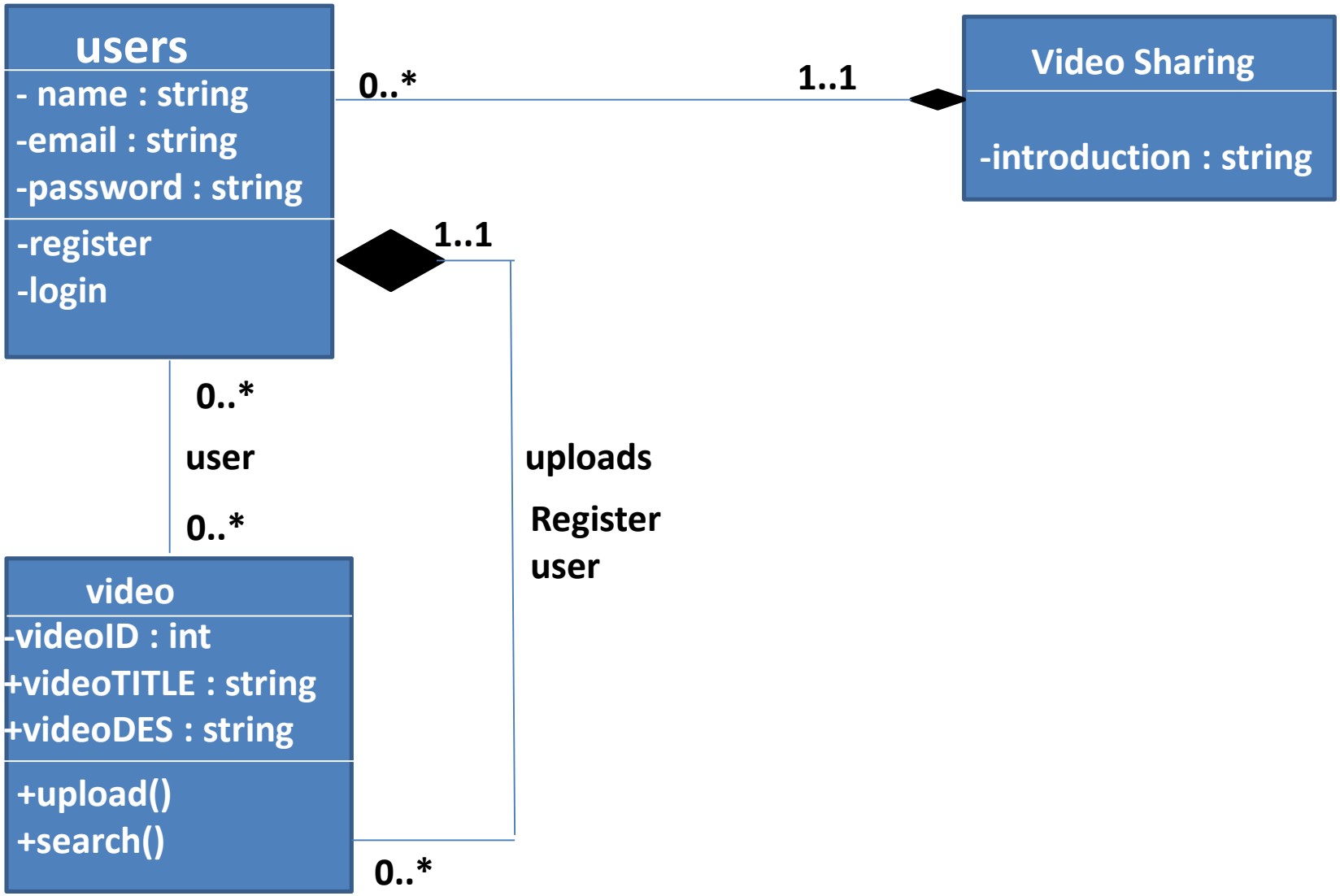


1.1 Class diagram…

- **Elements of a class diagram:**
- **Composition relation**

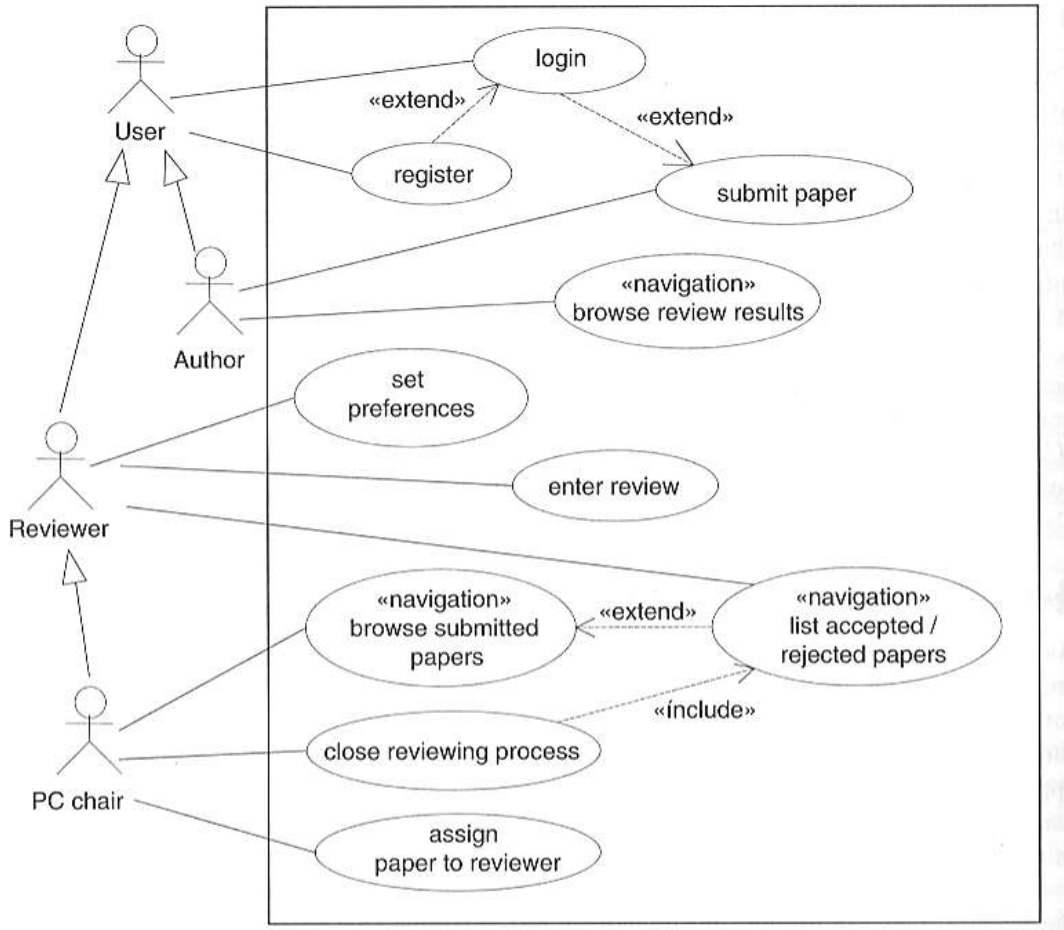


1.1 Class diagram…



1.1 Class diagram…

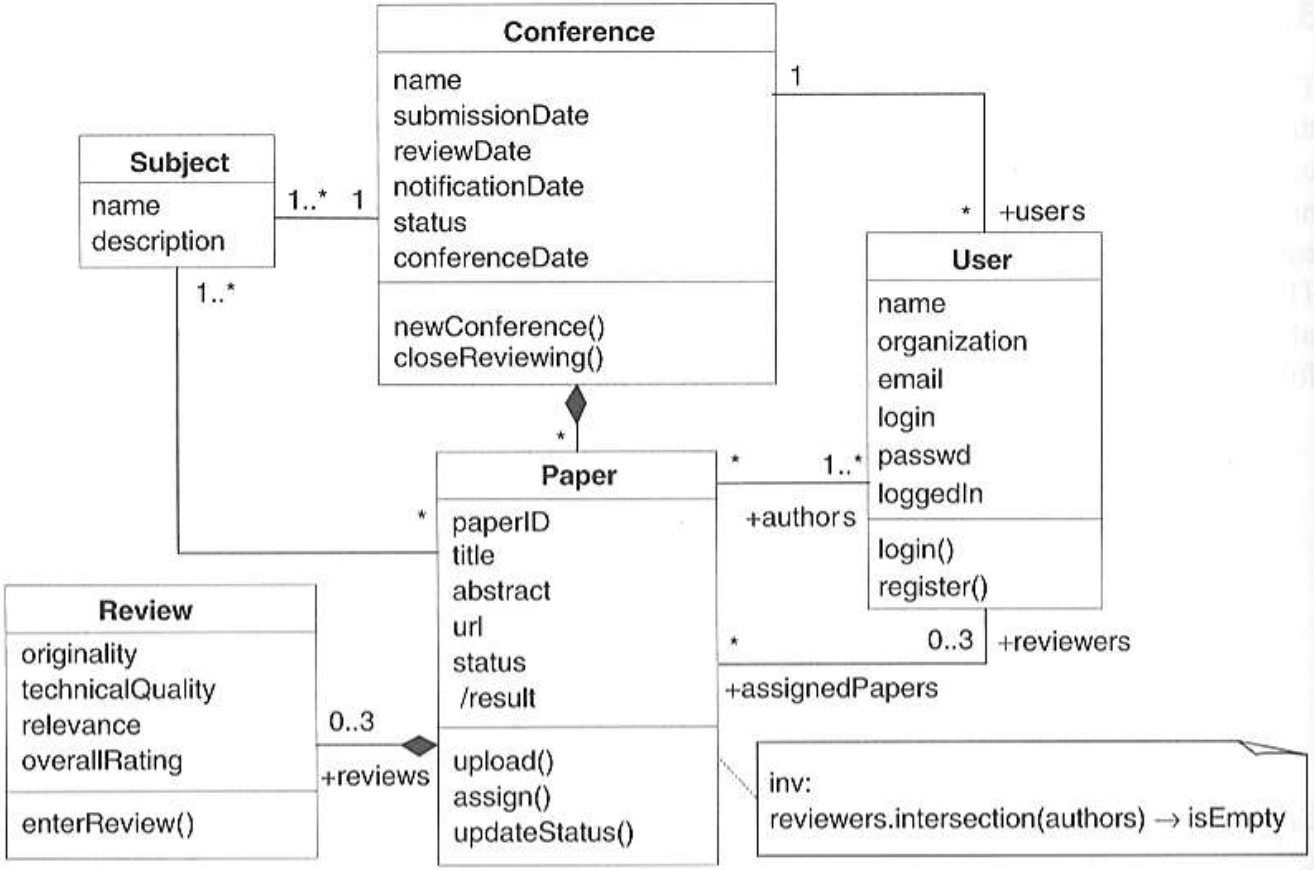
- Use-case diagram : Conference Paper Submission



Source: Web Engineering – Kappel et al.

1.1 Class diagram…

- **Conference Paper Submission System**

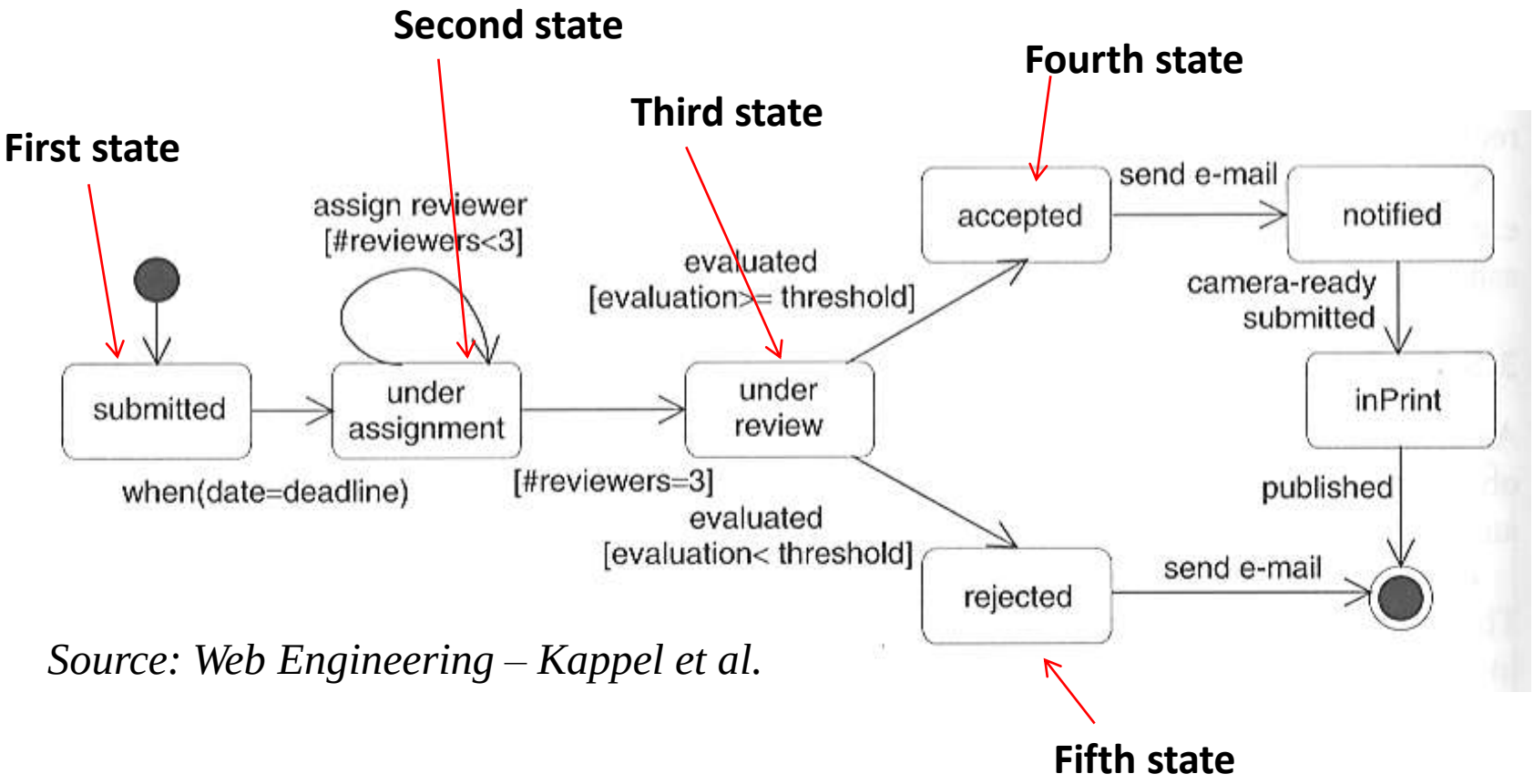


Source: Web Engineering – Kappel et al.

1.2 State machine diagrams

- For dynamic Web applications, they depict important states and events of objects, and how objects behave in response to an event (transitions)
- Show the life-cycle of an object
- Used only for state-dependent objects

1.2 State machine diagram..



Source: Web Engineering – Kappel et al.

2. Navigation Modeling

- **Models how web-pages are linked together**
  - **defines the structure of the hypertext**
    - **Which classes of the content model can be visited by navigation**
  - **Content to navigation**
  - **http://uwe.pst.ifi.lmu.de/teachingTutorialNavigation.html**

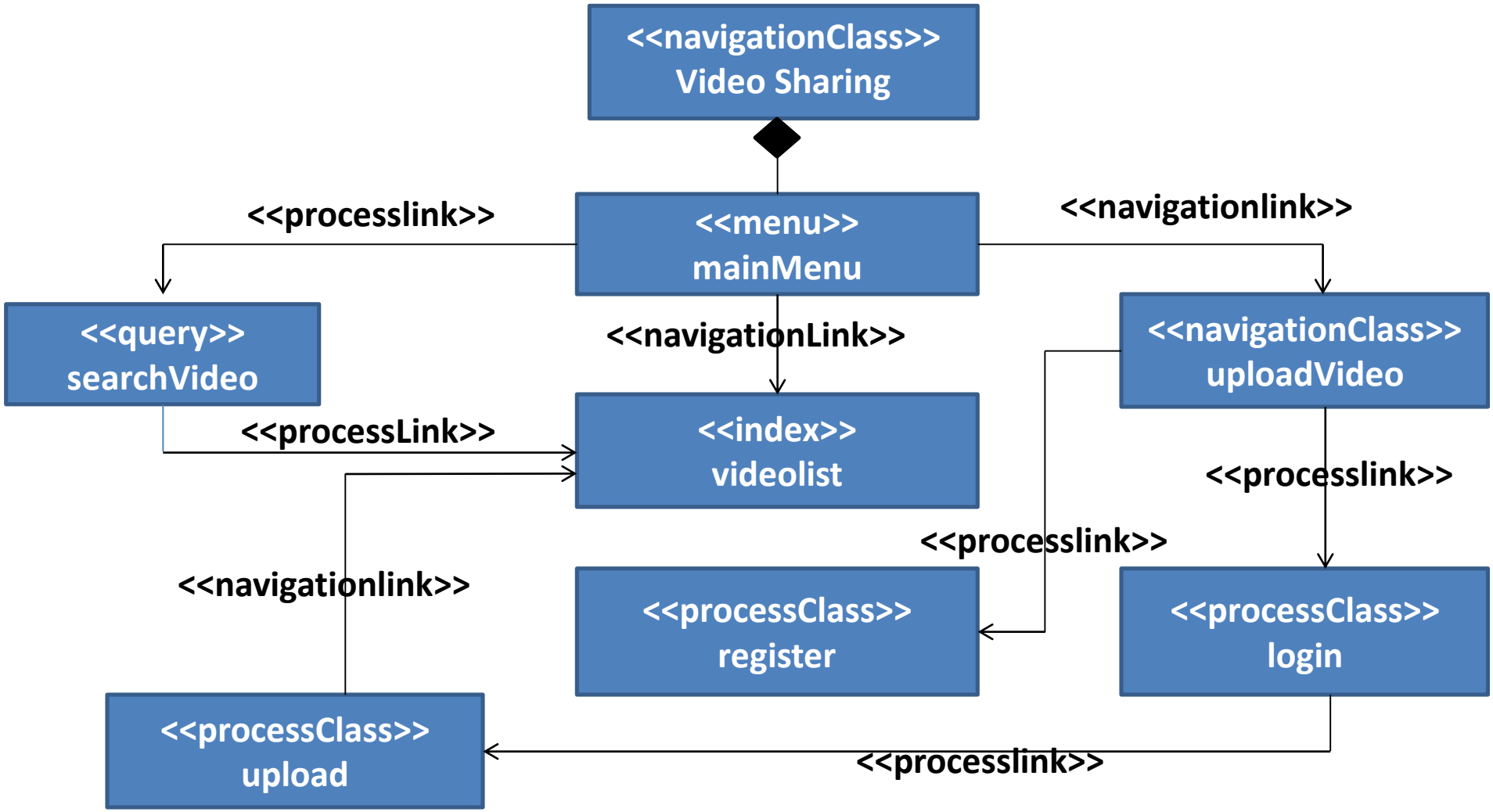
2. Navigation Modeling…

- **UWE navigation modeling**
  - **navigationClass**
  - **menu** 
  - **Index** 
  - **query** 
  - **processClass** 
  - **Processlink**
  - **Navigation link**
  - **External link** 

2. Navigation Modeling

- Online video sharing:
- Home page
  - video list
  - search video
  - upload video
    - register
    - login
      - upload

2. Navigation modeling…



3. Presentation Modeling

- **Purpose:** **To model the look & feel of the Web application at the page level**
- **The design should aim for simplicity and self-explanation**
- **Describes presentation structure:**
  - **Composition & design of each page**

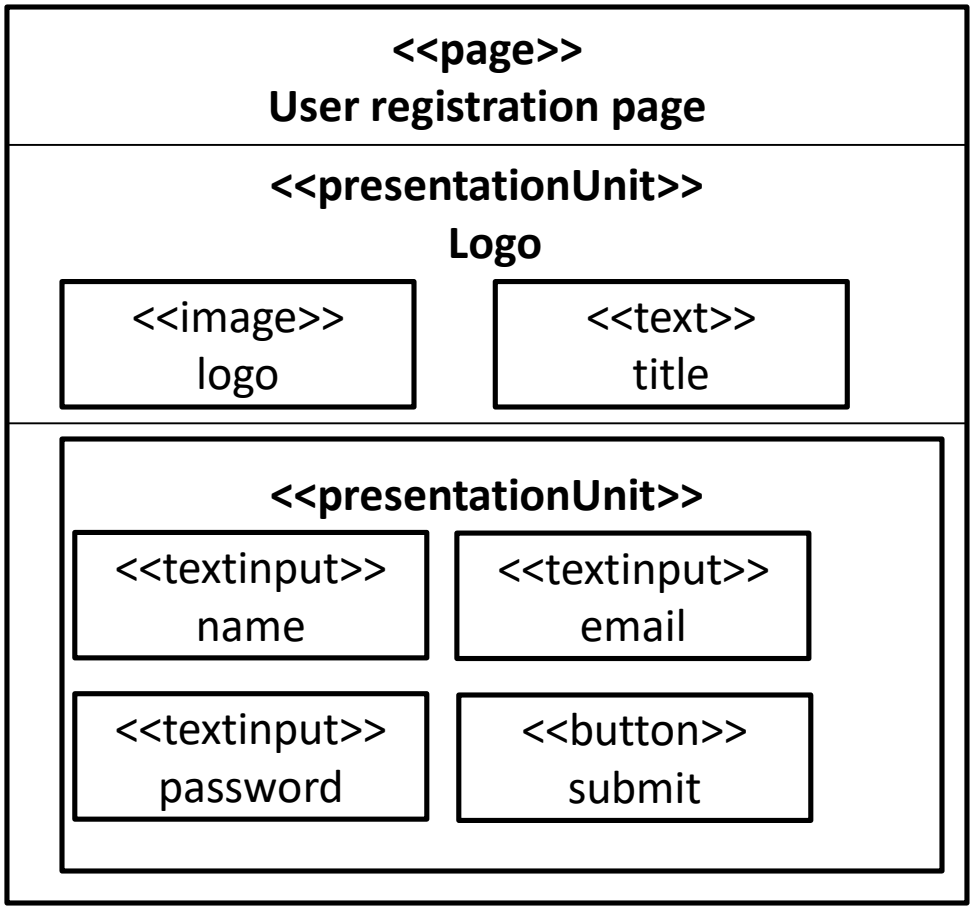
3. Presentation Modeling…

- **Levels:**
- **Presentation Page**
  - **page container**
- **Presentation Unit**
  - **A fragment of the page logically defined by grouping related elements**

3. Presentation Modeling…

- **Levels:**
- **Presentation Element**
  - **A unit's informational components**
  - **Text, images, buttons, fields**

3. Presentation Modeling…



Summary

- **Content modeling**
  - **class diagram**
  - **state machine diagram**
- **Navigation modeling**
- **Presentation modeling**

References

- **Chapter 3, Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). Web Engineering, Hoboken, NJ: Wiley & Sons**
- **Chapter 5, Sommerville, Software Engineering, ISBN-10: 0-13-703515-2 , PEARSON**
- **http://uwe.pst.ifi.lmu.de**

Web application architecture

Summary of the previous lecture

- **System modeling**
- **Requirement Modeling**
  - **use-case diagram, activity diagram**
- **Content modeling**
  - **class diagram, state machine diagram**
- **Navigation modeling**
- **Presentation modeling**

Outline

- Software system architecture
- Specifics of web application architecture
- Layered web architecture
  - 2-layered architecture
  - 3-layered architecture
  - N-layered architecture

1. Software system architecture

- **The architecture of a computer system is the high-level (most general) design on which the system is based**
- **Architectural features include:**
  - **Components**
  - **Collaborations (how components interact)**
  - **Connectors (how components communicate)**

1. Software system architecture…

- Key attributes of an architecture
  - architecture describes structure
  - architecture forms the transition from analysis to implementation
  - different viewpoints (conceptual, runtime, process and implementation)
  - makes a system understandable

1. Software system architecture…

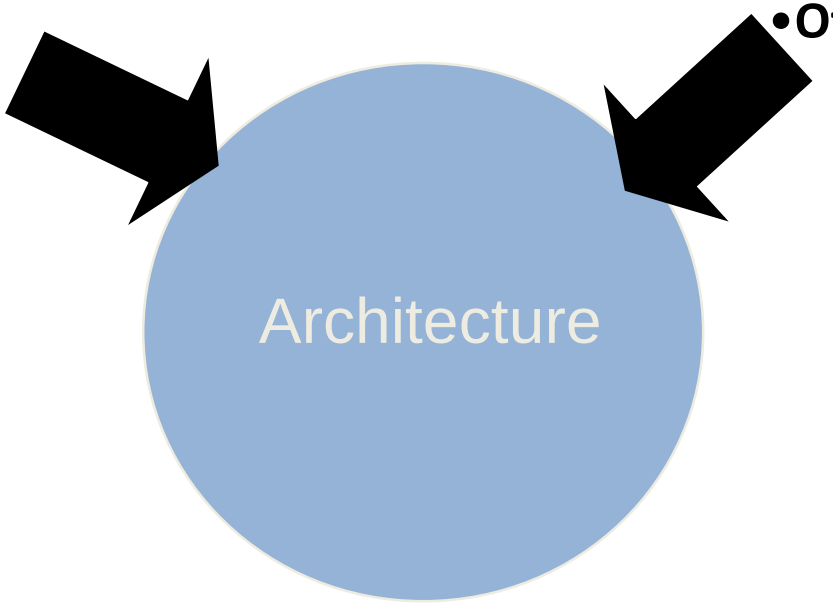
- **Factors influence the system architecture**

Functional Requirements

- Clients
- Users
- Other Stakeholders

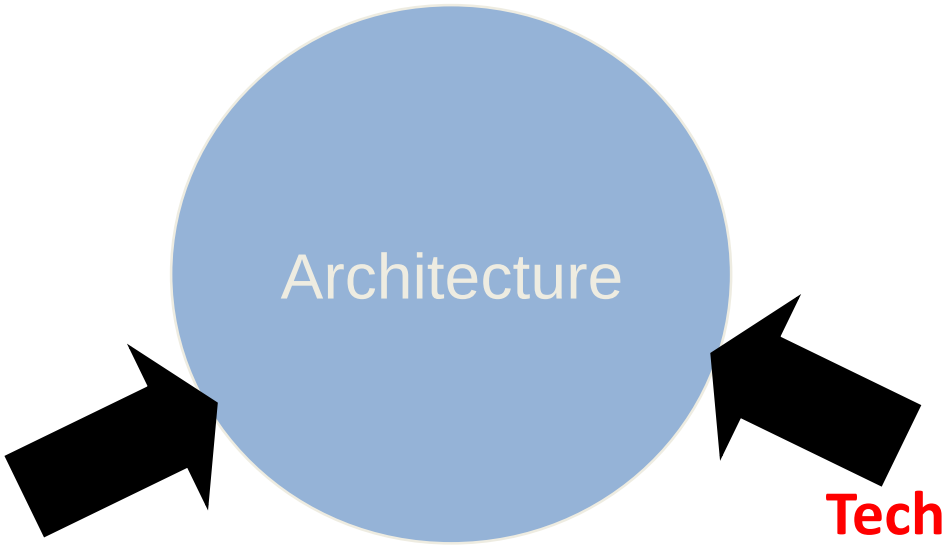
Quality considerations with

- Performance
- Scalability
- Reusability
- Other?



1. Software system architecture…

- **Factors influence the system architecture**



Experience with
- Existing Architecture
- Patterns
- Project Management
- Other?

Technical Aspects
- Operating System
- Middleware
- Legacy Systems
- Other?

2. Specifics in web application architecture

- A number of architectures for specific requirement in several application domain have been developed
- For web application architecture, usually we consider
  - layering aspect: to implement the principle of 'separation of concerns'
  - data aspects: to support processing of structured and non-structured data

2. Specifics in web application architecture…

- **For web applications quality requirements are more demanding as compared to desktop applications**
  - **performance, security, scalability, and availability etc.**
- **Need specific technical infrastructures both for the development and the operation of web applications**

2. Specifics in web application architecture

- **we have to consider**
  - **web infrastructure architecture (WPA)**
  - **web application architecture (WAA)**
- **Web application architecture (WAA) depends on the problem domain of the application, therefore we focus on web platform architecture (WPA)**

2. 1 Components of a web application architecture

- **Client:**
  - generally a browser (user agent) is controlled by a user to operate the web application
  - the client's functionality can be expanded by installing plug-ins
- **Firewall:**
  - a piece of software regulating the communication between insecure networks(e.g., the Internet) and secure networks (e.g., corporate LANs)
  - this communication is filtered by access rules

2. 1 Components of a web application architecture

- **Proxy:**
  - **A proxy is typically used to temporarily store web pages in a cache**
- **Web server:**
  - **A Web server is a piece of software that supports various Web protocols like HTTP, and HTTPS, etc., to process client requests**

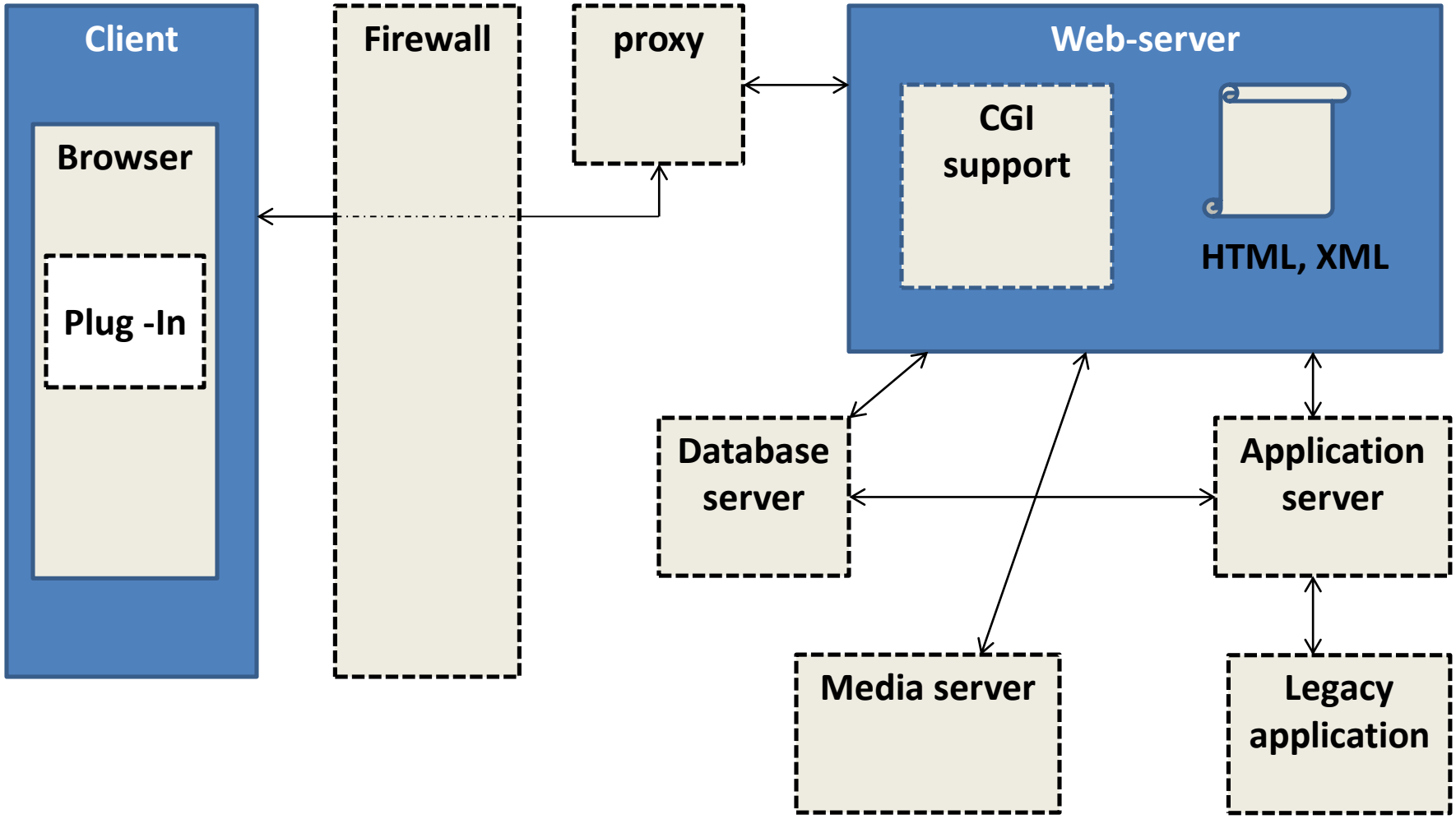
2. 1 Components of a web application architecture

- **Database server:**
  - **this server normally supplies data in structured form, e.g., in tables**
- **Legacy application:**
  - **A legacy application is an older system that should be integrated as an internal or external component**

2. 1 Components of a web application architecture

- **Media server:**
  - **This component is primarily used for content streaming of non-structured bulk data (e.g., audio or video)**
- **Application server:**
  - **An application server holds the functionality required by several applications**

2. 1 Components of a web application architecture



3. Layered architecture for web applications

- **Presentation tier:**
  - **Every web application needs to communicate with external entities, human users or other computers**
    - **allows these entities to interact with the system**
    - **implemented as a GUI interface**
  - **How the data should appear to the user**

3. Layered architecture for web applications

- **Application tier:**
  - **Web applications do more than information delivery, they perform data processing (Business Logic & calculation) behind the results being delivered**
  - **This tier is often referred to as**
    - **Services**
    - **Business logic**

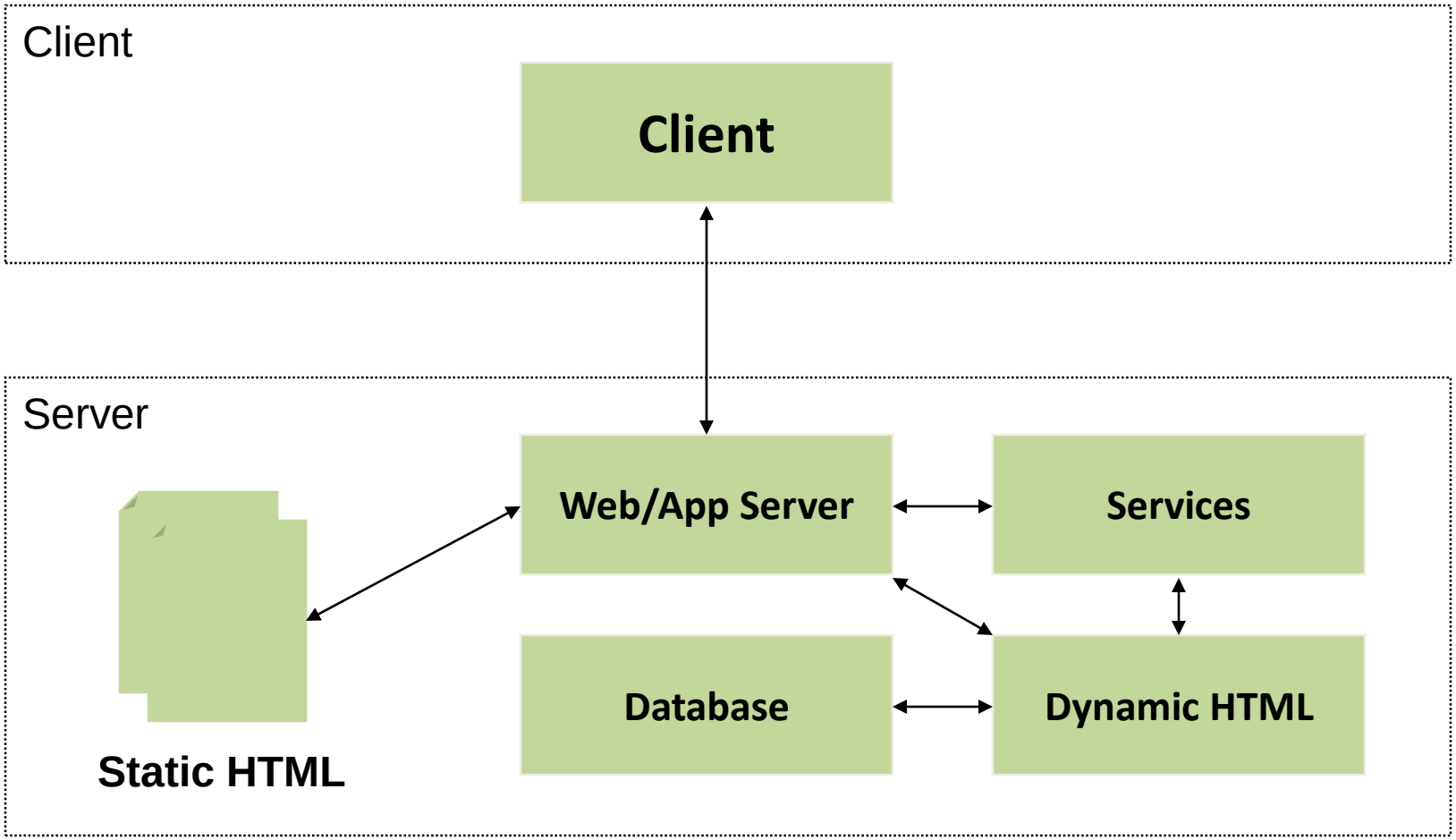
3. Layered architecture for web applications

- **Data layer:**
  - **Web applications needs data to work with**
  - **Data can reside in databases or other information repositories**
  - **Deals with and implements different data sources of Information Systems**

3.1 two-layer web architecture

- **Presents architecture in two layers:**
  - **Layer 1: Client platform, hosting a web browser**
  - **Layer 2: server platform, hosting all server software components**
- **Also called client/server architecture**
- **Client directly send request to the server**
  - **Server respond to the client request**
    - **Static or dynamic requests**

3.1 two-layer web architecture



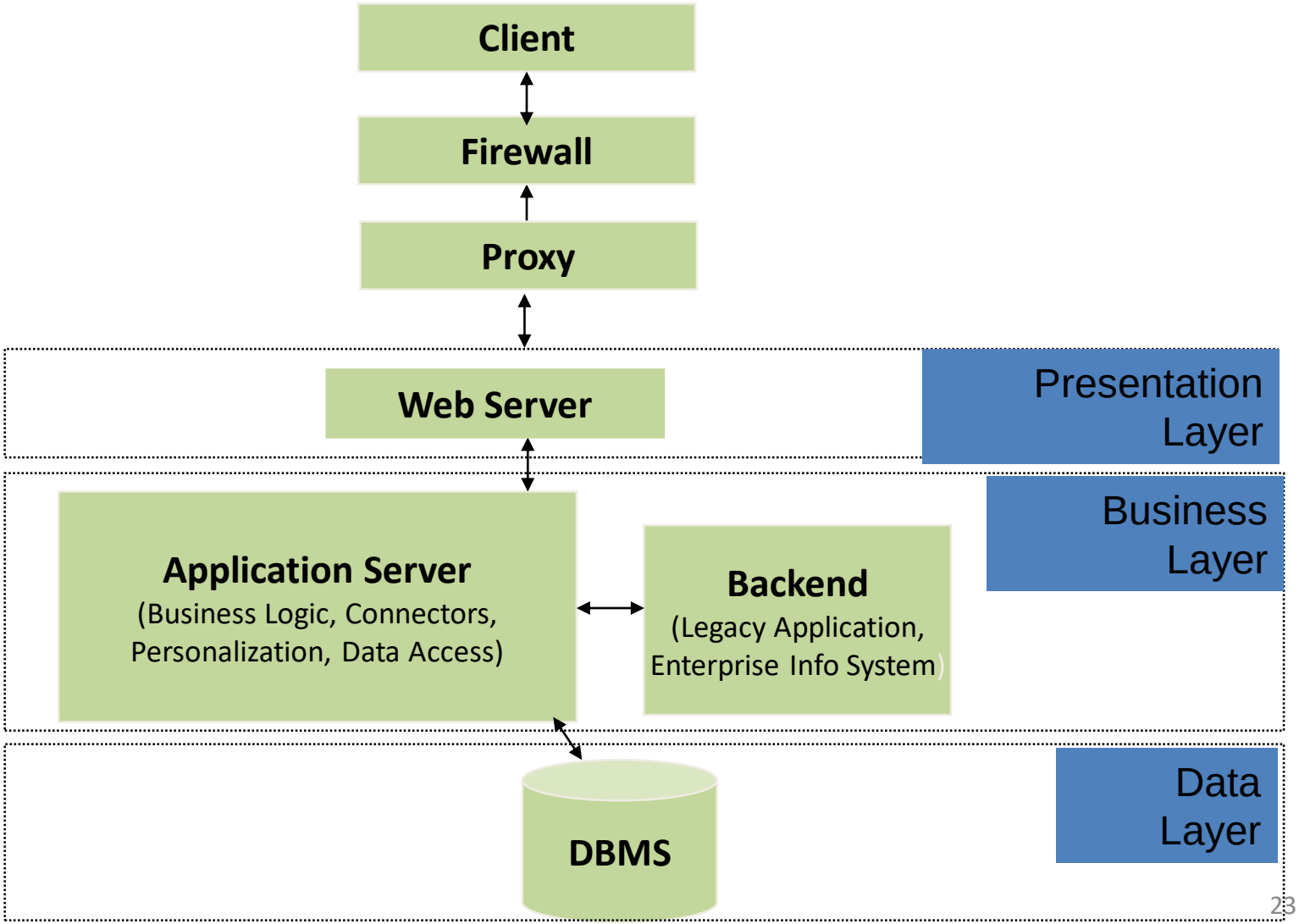
3.1 two-layer web architecture

- Advantage:
  - Inexpensive (single platform)
- Disadvantages:
  - Interdependency (coupling) of components
  - No redundancy
  - Limited scalability
- Typical application:
  - 10-100 users
  - Small company or organization

3.2 three-layer web architecture

- **Usually implemented in 3 layers**
  - **Layer 1: Data**
  - **Layer 2: Application**
  - **Layer 3: presentation**
- **Additionally, security mechanism (Firewall) and caching mechanism (Proxies) can be added**

3.1 three-layer web architecture



3.2 three-layer web architecture

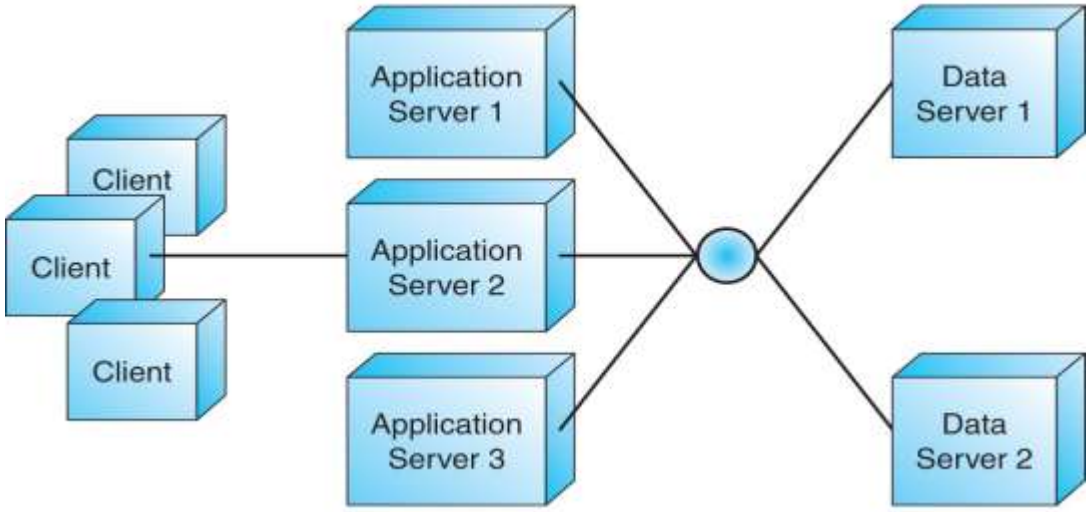
- Advantages:
  - Improved performance
  - Decreased coupling of software components
  - Improved scalability
- Disadvantages:
  - No redundancy
- Typical Application:
  - 100-1000 users
  - Small business or regional organization, e.g., specialty retailer, small college

3.3 N-layer web architecture

- **A multitier (N-layer) architecture is an expansion of the 3-layer architecture, in one of several different possible ways**
  - **Replication of the function of a layer**
  - **Specialization of function within a layer**

3.3 N-layer web architecture…

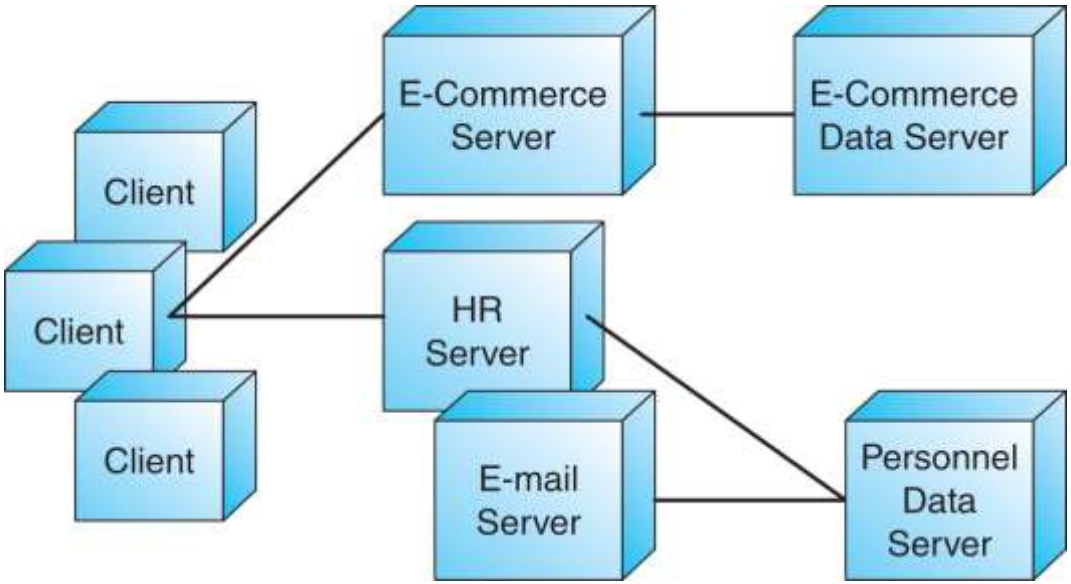
- **Replication:**
- **Application and data servers are replicated**
- **Servers share the total workload**



3.3 N-layer web architecture…

- **Specialization:**
- **Servers are specialized**
- **Each server handles a designated part of the workload, by function**

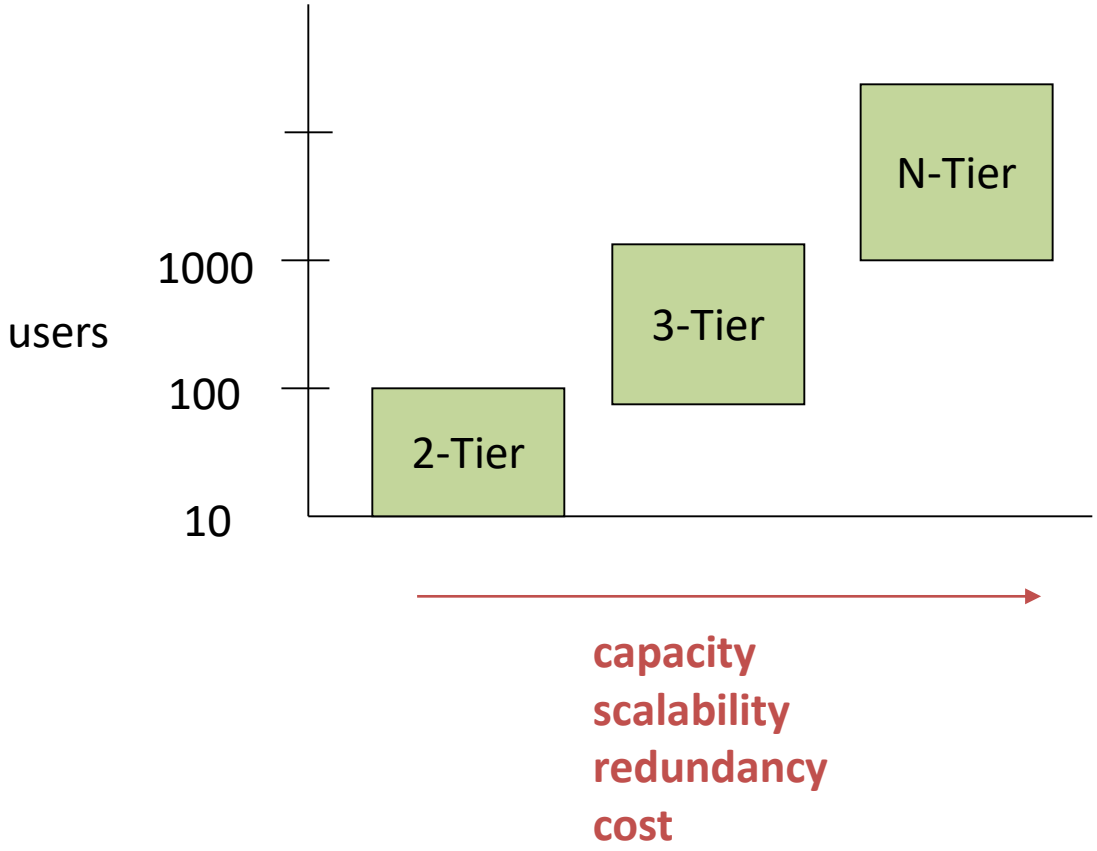
3.3 N-layer web architecture…



3.3 N-layer web architecture…

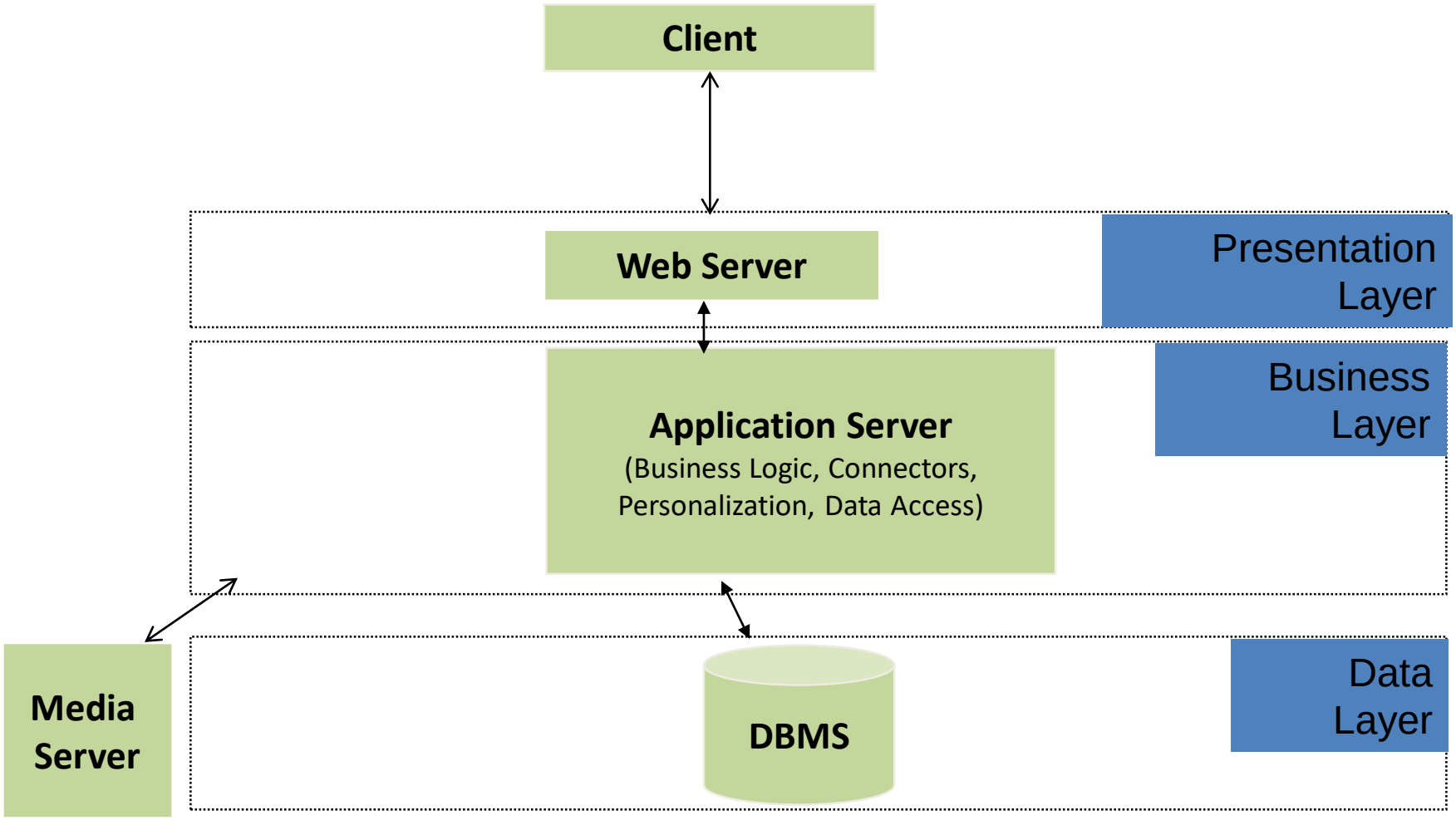
- Advantages:
  - Decoupling of software components
  - Flexibility to add/remove platforms in response to load
  - Scalability
  - Redundancy
- Disadvantages:
  - Higher costs (maintenance, design, electrical load, cooling)
- Typical Application:
  - 1000+ users
  - Large business or organization

3.4 Comparison of layered architecture



- **large e-commerce, business, or organization**
- **small e-commerce, regional business or organization**
- **local business or organization**

3.5 example


Client
Web Server
Presentation Layer
Application Server
(Business Logic, Connectors, Personalization, Data Access)
Business Layer
Media Server
DBMS
Data Layer

The Model-View-Controller (MVC) Architectural Pattern

- Architectural pattern to help **separate *user interface layer* from other parts of the system**
- Great way to have **layered cohesion**, as interfaces or controlled.
- **Coupling** reduced between UI layer and rest of system.

THE MVC pattern separates the

Model: the functional layer (business entities, 'key abstractions,' the objects, relations, ...) from the

View: the user interface and the

Controller; the **director / sequencer** of the activities in response to the user.

MVC

- The ***model*** contains the underlying **classes** whose instances **(objects)** are to be viewed and manipulated
  - —Model will likely contain classes from the **domain** that may be general and form the **application** itself, which may be unique or specialized to the application.
  - —These may be very complicated software objects.

- The ***view*** contains objects used to **render the appearance** of the data from the model in the **user interface** and the **controls** with which an actor can interact.

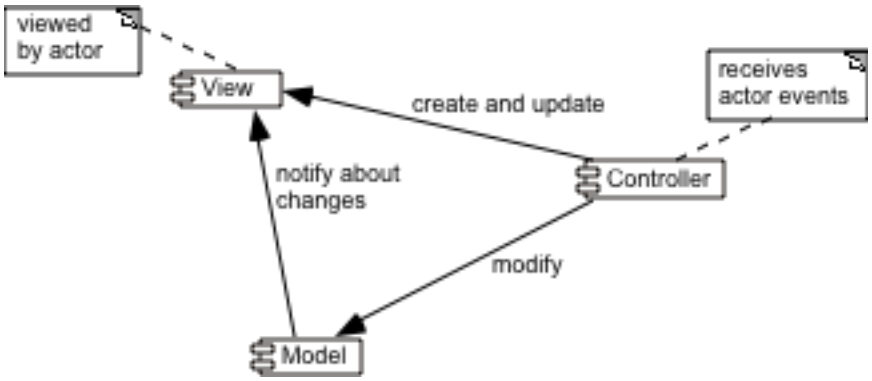
- The ***controller*** contains the objects that **control** and **handle** the user's interaction with the view and the model.
  - —Controller contains business logic…and response to events.

– (The **Observable design pattern** is normally used to separate the model from the view (later))

Example of the MVC architecture for the UI

- MVC exhibits **layer cohesion**, as the model **has no idea**
- what view and controller are attached to it (doesn't care!).
- **Model** is 'passive' in this respect.
- The **View** (UI), business services (controller), and **model**
- (business entities / core abstractions) will reside in **different**
- **architectural layers**.

Example of the MVC architecture for the UI



There may be special cases when no controller component is created, but the separation of the model from the view is still essential.

The MVC Architecture and Design Principles

- 1. *Divide and conquer*: Three components can be somewhat independently designed.
- 2. *Increase cohesion*: Components have **stronger** layer cohesion than if the view and controller were together in a single UI layer.
- 3. *Reduce coupling*: **Minimal** communication channels among the three components.
- 6. *Increase reuse*: The **view** and **controller** normally make ***extensive*** use of reusable components for various kinds of UI controls.
- 7. *Design for flexibility*: It is usually quite easy to change the UI by changing the **view**, the **controller**, or both.
- 10. *Design for **testability***: Can *test* application separately from the UI.

Summary

- **Software system architecture**
- **Specifics of web application architecture**
- **Layered web architecture**
  - **2-layered architecture**
  - **3-layered architecture**
  - **N-layered architecture**

References

- **Chapter 4, Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). Web Engineering, Hoboken, NJ: Wiley & Son**
- **Web-based application development,**

  Ralph F Grove , J and B publishers,

  (www.jbpub.com)