

SWE 363: Web Engineering & Development

Module 6-2

Web Modeling & Architecture



Objectives

- ❑ Learn to model web applications
- ❑ Learn how to build the web architecture

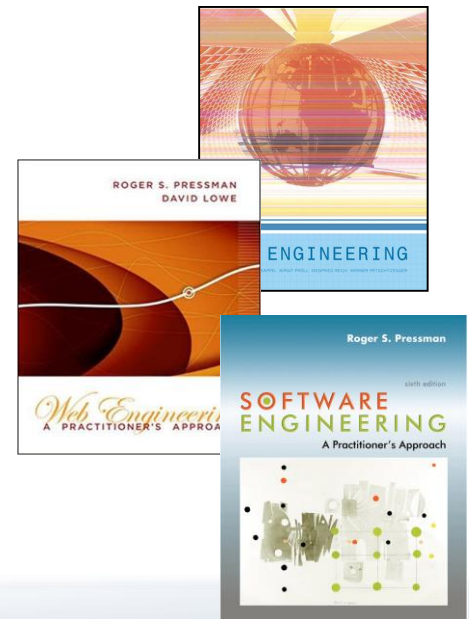
References

□ Papers

- Schwinger, Wieland, and Nora Koch. "Modeling web applications." *Web Engineering* (2006): 39-64.
- "The expressive power of uml-based web engineering." Koch, Nora, and Andreas Kraus. *Second International Workshop on Web-oriented Software Technology (IWWOST02)*. Vol. 16. CYTED, 2002.

□ Books

- "Web Engineering: The Discipline of Systematic Development of Web Applications" by Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006), Wiley & Sons.
- "Web Engineering: A Practitioner's Approach" by Roger S. Pressman and David Lowe, 2008, McGraw-Hill Education
- "Software engineering: a practitioner's approach". Pressman, Roger S. (2005), Palgrave Macmillan.



❑ Web Modeling

- Requirements Modeling
- Content Modeling
- Hypertext modeling (navigation)
- Presentation modeling
- Customization modeling

❑ Web Architecture

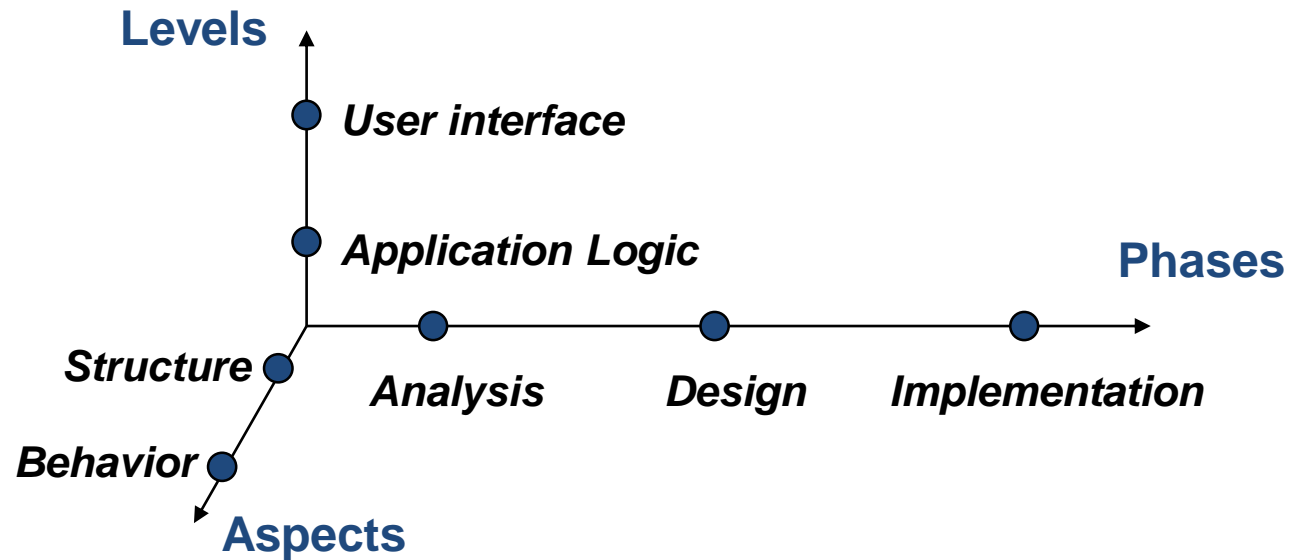
Why Modeling?

- ❑ Purpose: to define an abstract view of a real-world entity
 - Finding & discovering objects/concepts in a domain
 - Assigning responsibilities to objects

- ❑ Modeling addresses one of the major problems of today's development:
little planning of Web Applications prior to implementation

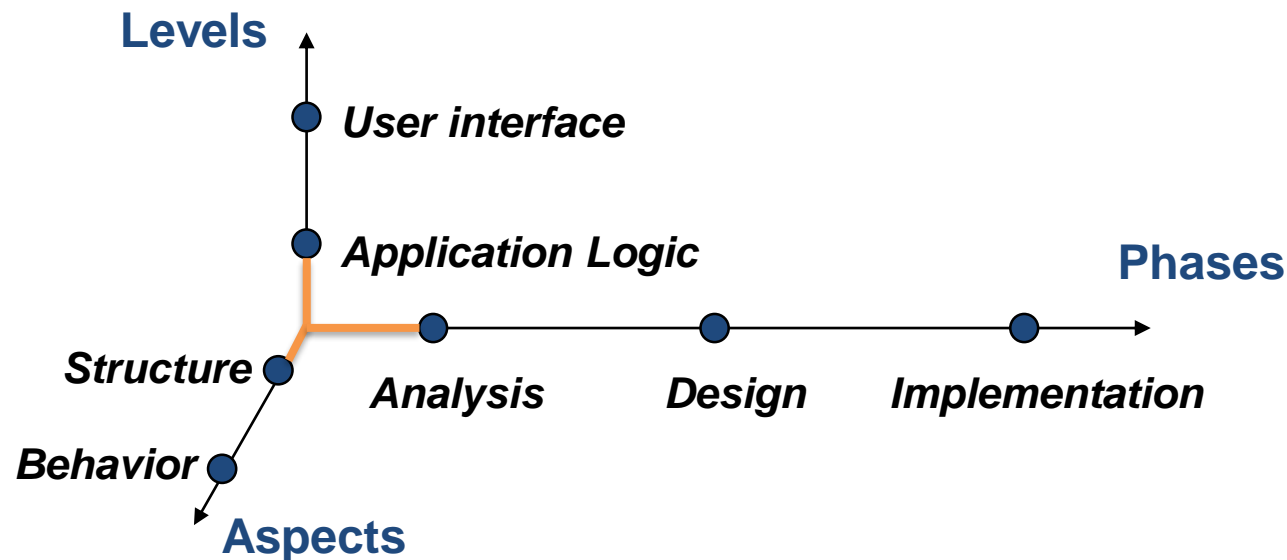
- ❑ Our focus is modeling of
 - static & dynamic aspects of content,
 - hypertext, and
 - presentation

Software Application Modeling



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

Software Application Modeling



❑ Roots of Modelling:

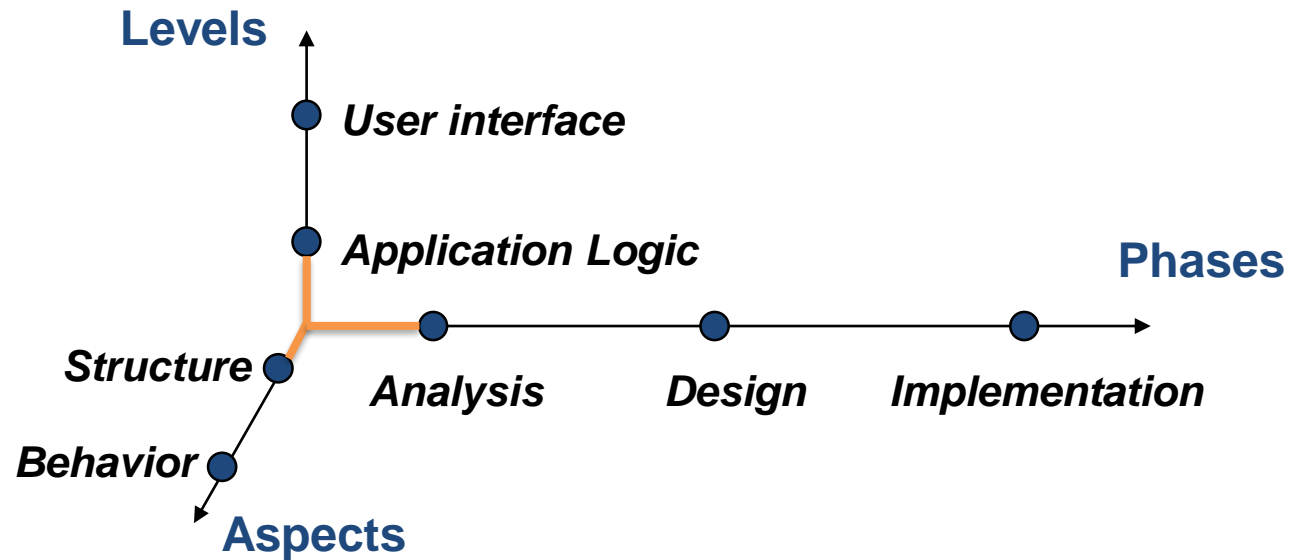
- Data Engineering - focusing on structural aspects
- Software Engineering - focusing on behavioral aspects

❑ Used Modelling Formalisms:

- Entity Relationship Technique (ER)
- UML 2.0

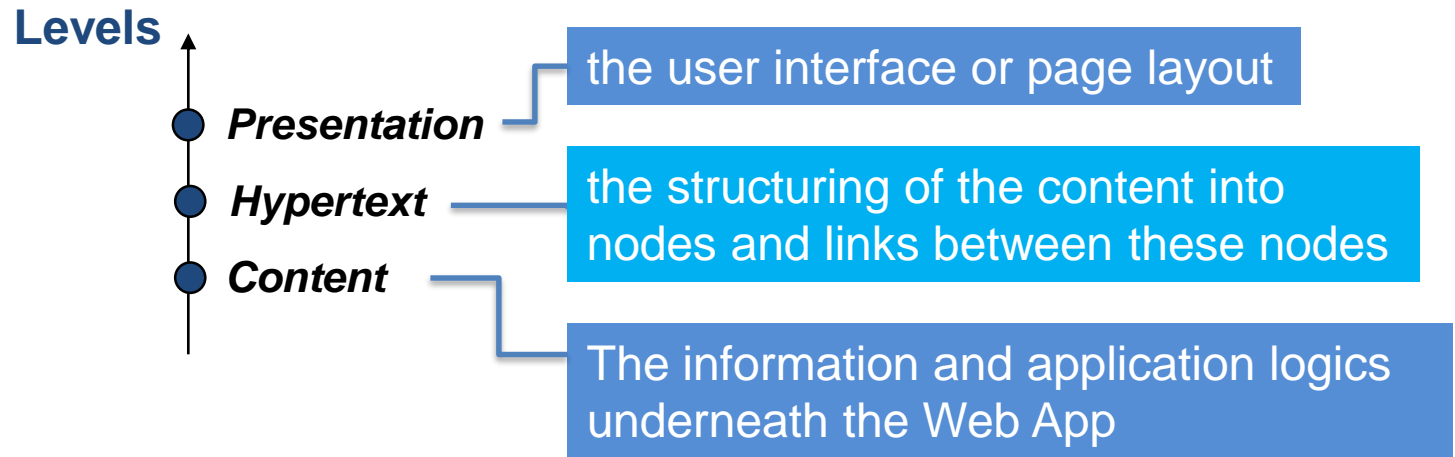
“*Unified Modeling Language* is a visual language for specifying and documenting the artifacts of systems.”

Software Application Modeling



But does not regard one of the major characteristics of Web applications, namely **hypertext**

Requirements Framework for Modeling Web Applications



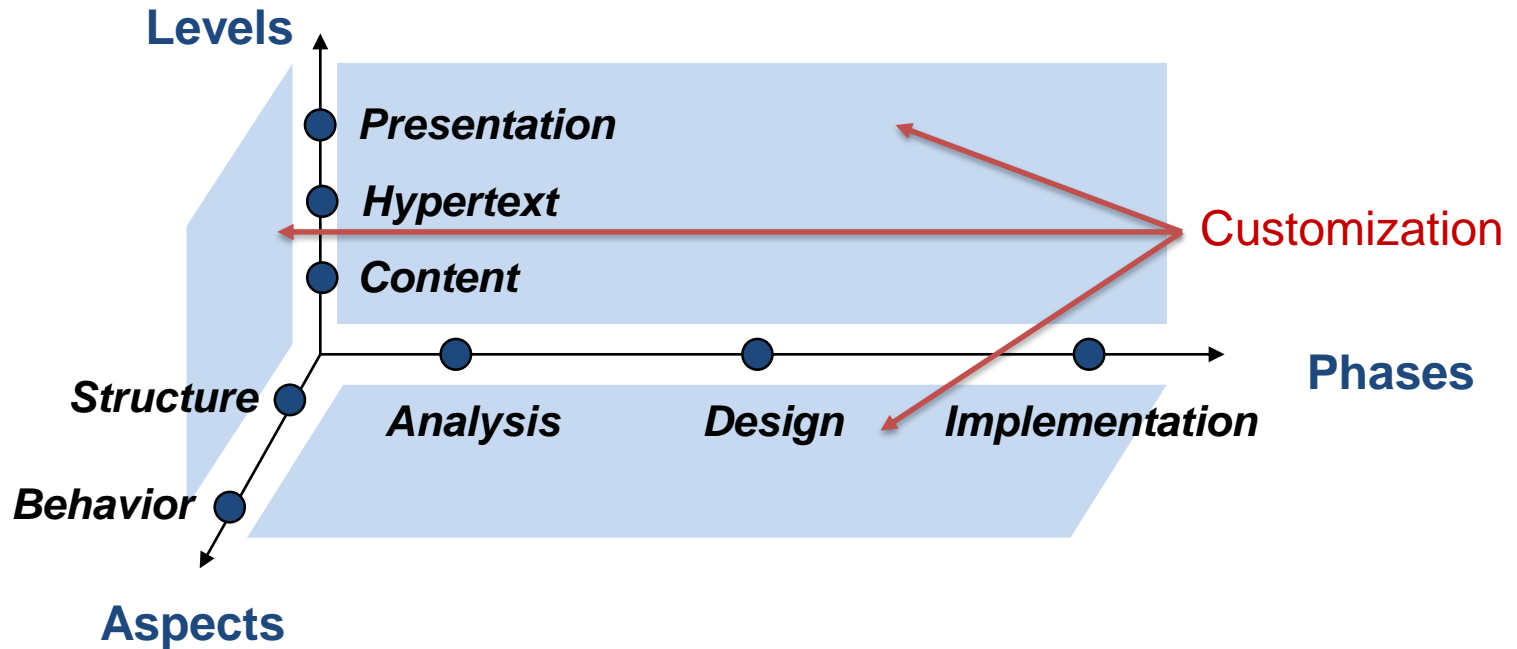
❑ Separation of levels

- explicit inter-dependencies between levels
- allows reuse and helps to reduce complexity

❑ Bottom-Up and Top-Down Design

- bottom-up: starting with the content level (e.g. given database) and derive the hypertext and presentation levels
- top-down: content level is derived from the other levels

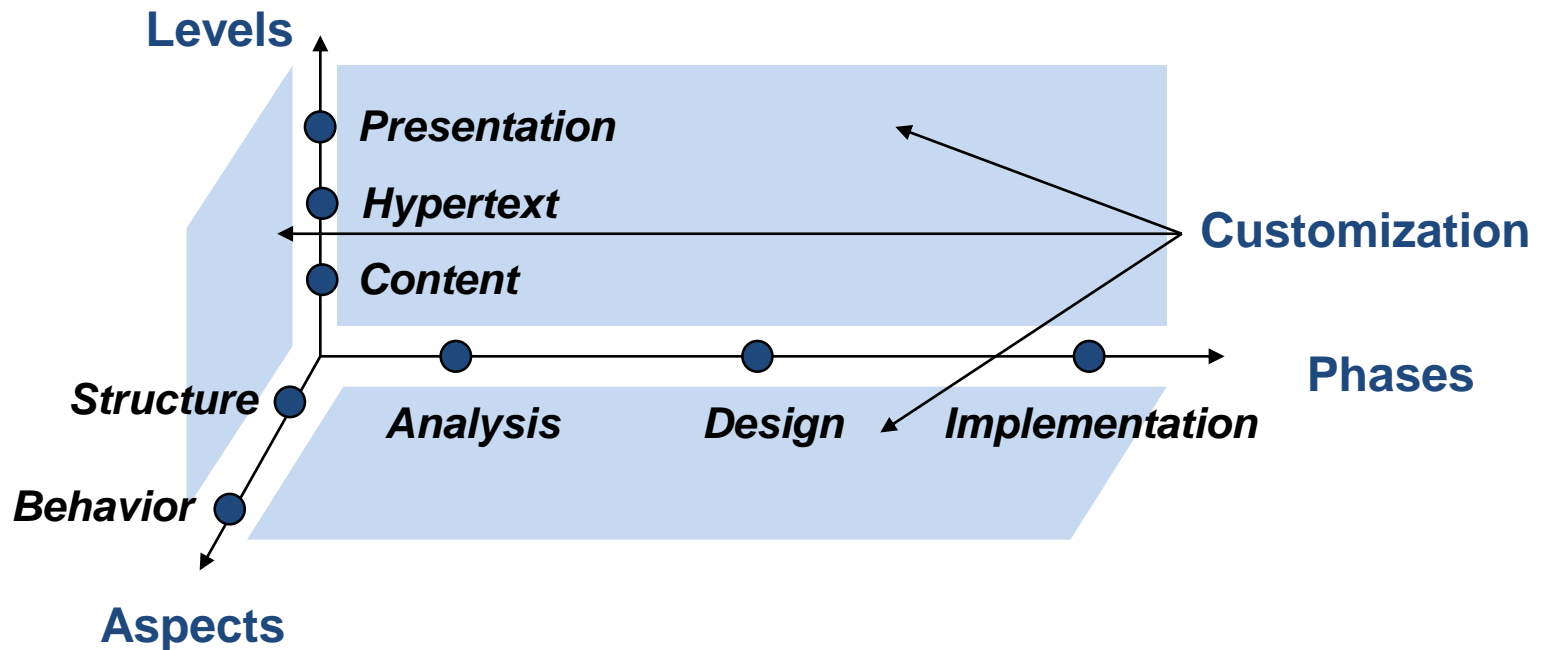
Web Application Modeling



- **Levels** – Information, node/link structure, UI & page layout separate.
- **Aspects** – Same as Software Applications
- **Phases** – Approach depends upon type of application
- **Customization** – Context information

The inclusion of **context information** in the development of Web applications plays a significant role to allow for e.g. *personalization* and location-based services

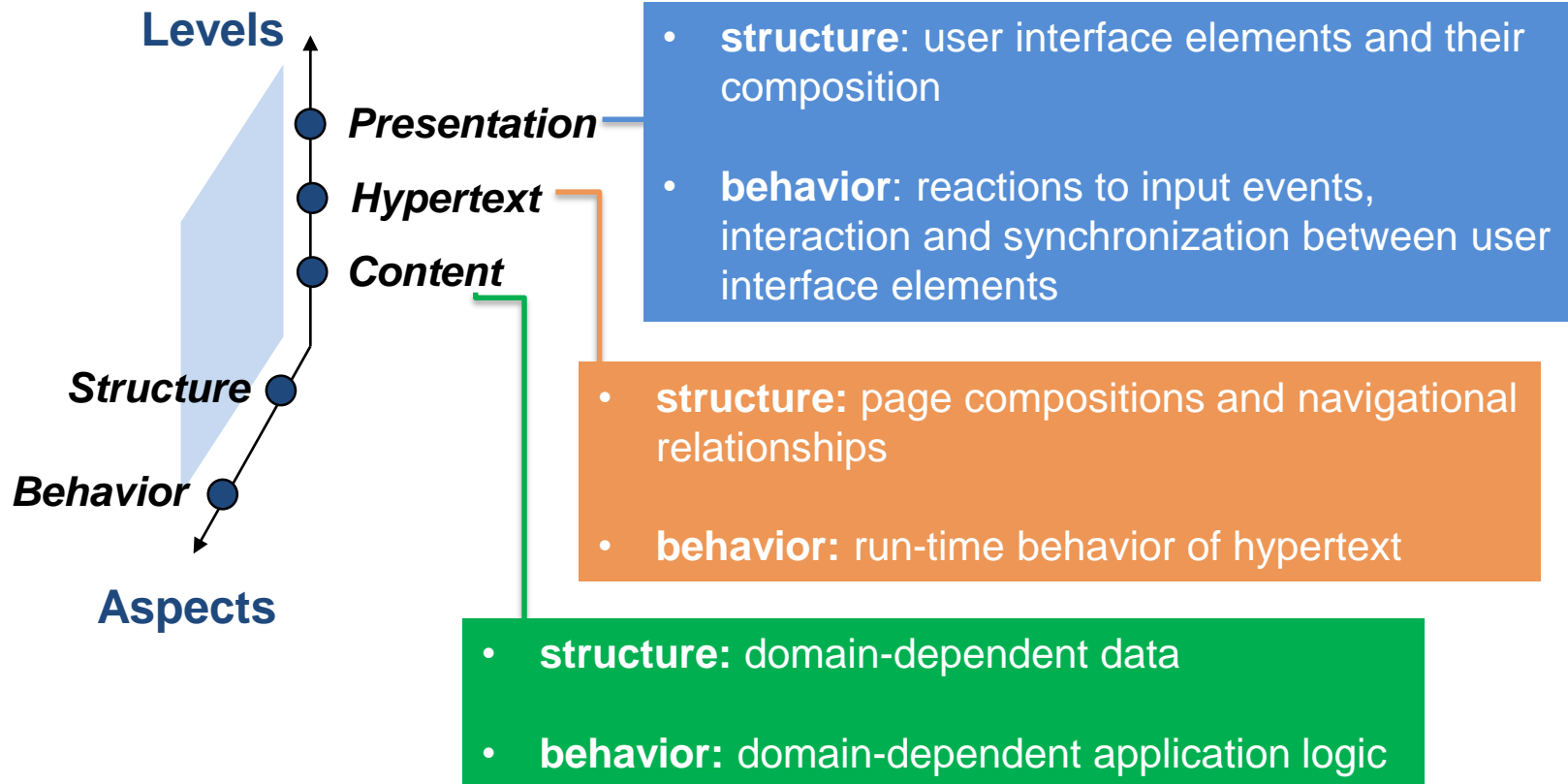
Web Application Modeling



❑ Customization

- considers the context, e.g., users' preferences, device characteristics, or bandwidth restrictions
 - allows to adapt the Web application accordingly.
- ❑ It **influences all three Web modeling dimensions** of content, hypertext, and presentation with respect to structure and behavior and should be taken into account in all phases of the development process.

Web Application Modeling



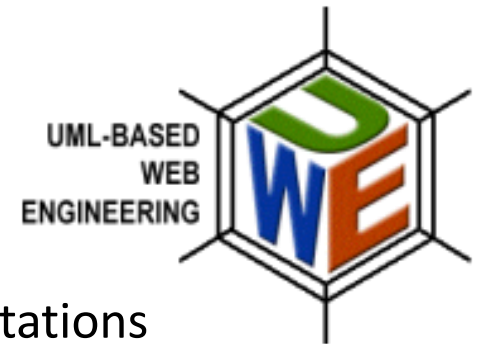
- The relevance of the structure and behavior models depends on the type of Web application to be implemented.

Web Modeling Approaches

- ❑ RMM: Relationship Management Model
- ❑ WebML: Web Modeling Language
- ❑ HDM: Hypertext Design Model
- ❑ WSDM: Web Site Design Method
- ❑ OOHDM: Object-oriented Hypermedia Design Method
- ❑ OOH: Object-oriented Hypermedia
- ❑ WAE: Web Application Extension
- ❑ UWE: UML-based Web Engineering

UWE: UML-based Web Engineering

- ❑ UML Web Engineering (UWE) is a software engineering approach for the **Web domain** aiming to cover the whole life-cycle of Web application development.
 - <http://uwe.pst.ifi.lmu.de/>
- ❑ Light-weight extension of UML
- ❑ For Web-centric modeling, we will employ the UWE notations
 - Relies on Object Management Group (OMG) standards – (i.e., UML-compliant)
 - Comprehensive modeling tool
 - Supports semi-automatic generation of code



Readings:

- ❑ N. Koch, A. Kraus: [The Expressive Power of UML-based Web Engineering](#)

Modeling Support in UWE

- ❑ Requirements Modeling
- ❑ Content Modeling
- ❑ Hypertext modeling (navigation)
- ❑ Presentation modeling
- ❑ Customization modeling

Source:

Schwinger, Wieland, and Nora Koch. "Modeling web applications." Web Engineering (2006): 39-64.

Requirements Modeling

- ❑ Serves as a bridge between **Requirements** & **Design** phases
- ❑ Emphasize the users goals and perspective
- ❑ Two types of requirements:
 - **Functional** (to be found in all software systems). >> UML activity diagrams
 - **Navigational** (typical for web applications)
- ❑ **Use cases** preferred modeling technique for **functional requirements**
 - provides **graphical overview** of a system's use cases, its external actors, and their relationships
 - Can be used for **functional & hypertext requirements**
 - Use “<<navigation>>” **stereotype** to distinguish **hypertext** from **functional**
 - Written details for various use cases
 - Name of use case, primary actor, scope, pre-conditions, post-conditions, related use cases, etc
- ❑ Suggested to keep separating the **functional** from the **navigational** use cases
- ❑ Use cases should be described **in detail**.
 - in **textual form** or
 - by use of a behavior diagram, e.g. an **activity diagram**.

Linking Use Cases

- ❑ *Association* relationships (Between Actor and UCs)
- ❑ *Generalization* relationships
 - One element (child) "is based on" another element (parent)
- ❑ *Include* relationships
 - One use case (base) includes the functionality of another use case (inclusion case)
 - One UC must call another; e.g., Login UC includes User Authentication UC
 - Supports re-use of functionality
- ❑ *Extend* relationships
 - One use case (extension) extends the behavior of another (base)
 - One UC calls another UC under certain condition; think of if-then decision points

Example: Conference paper submission System

□ Actors

- authors, submitting papers for the conference
- program committee members, reviewing papers
- program committee chair

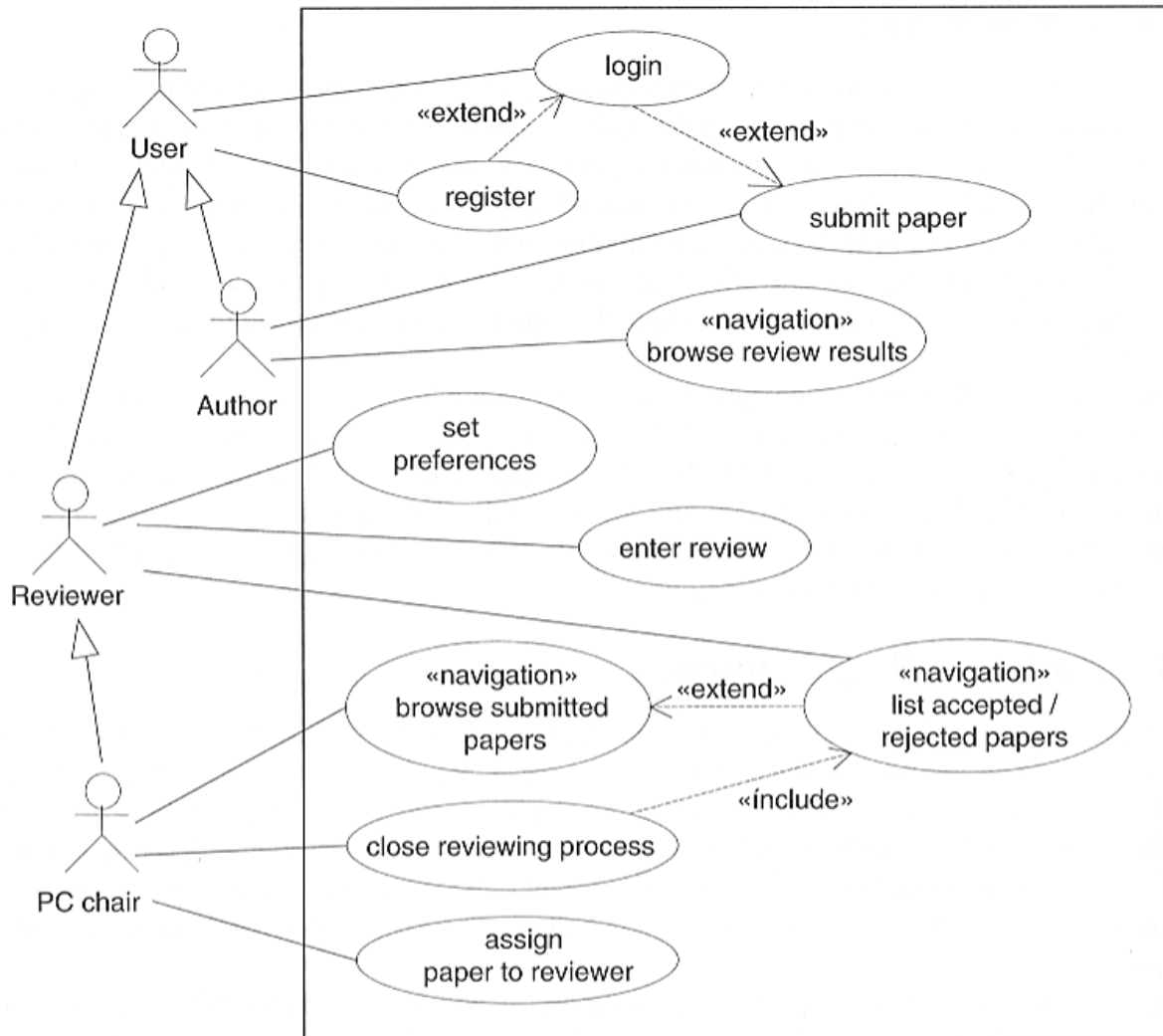
Note: All Web applications have at least one human user, most often anonymous

□ Functional requirements

- submit paper
- assign paper to reviewer
- produce review
- produce list of accepted / rejected papers

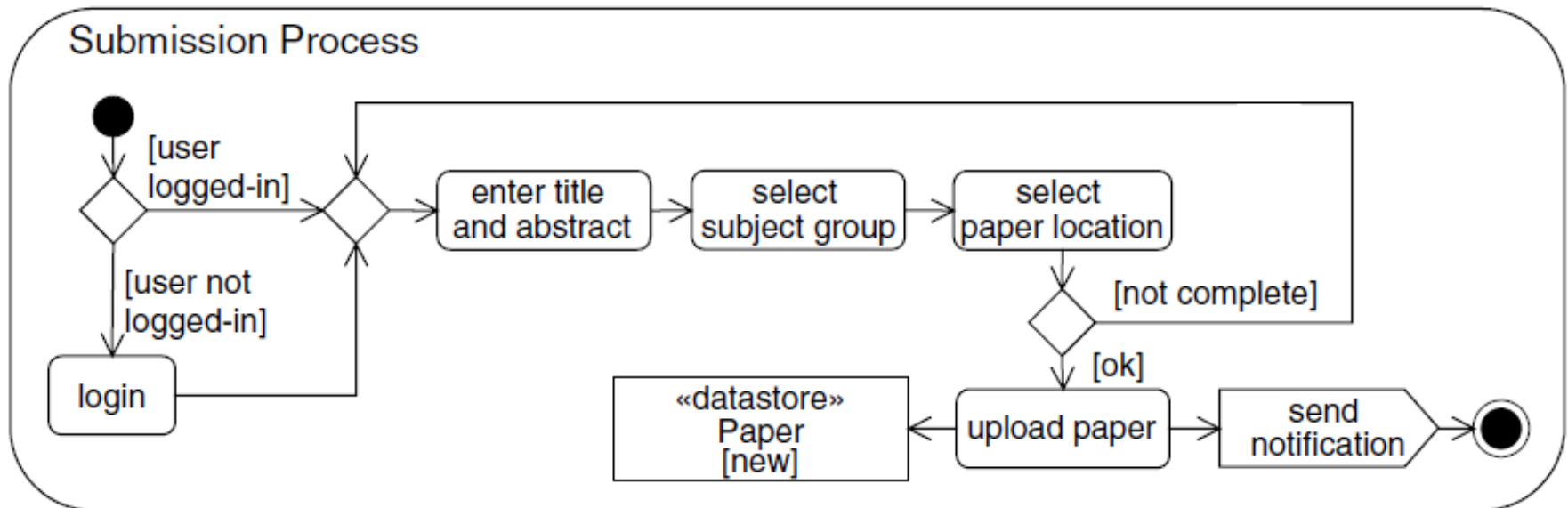
Use Case Diagram (UCD) Example

❑ Conference paper submission system



Activity Diagram (AD) Example

- Activity diagrams are graphical representations of **workflows** of stepwise **activities** and **actions** with support for choice, iteration and concurrency.



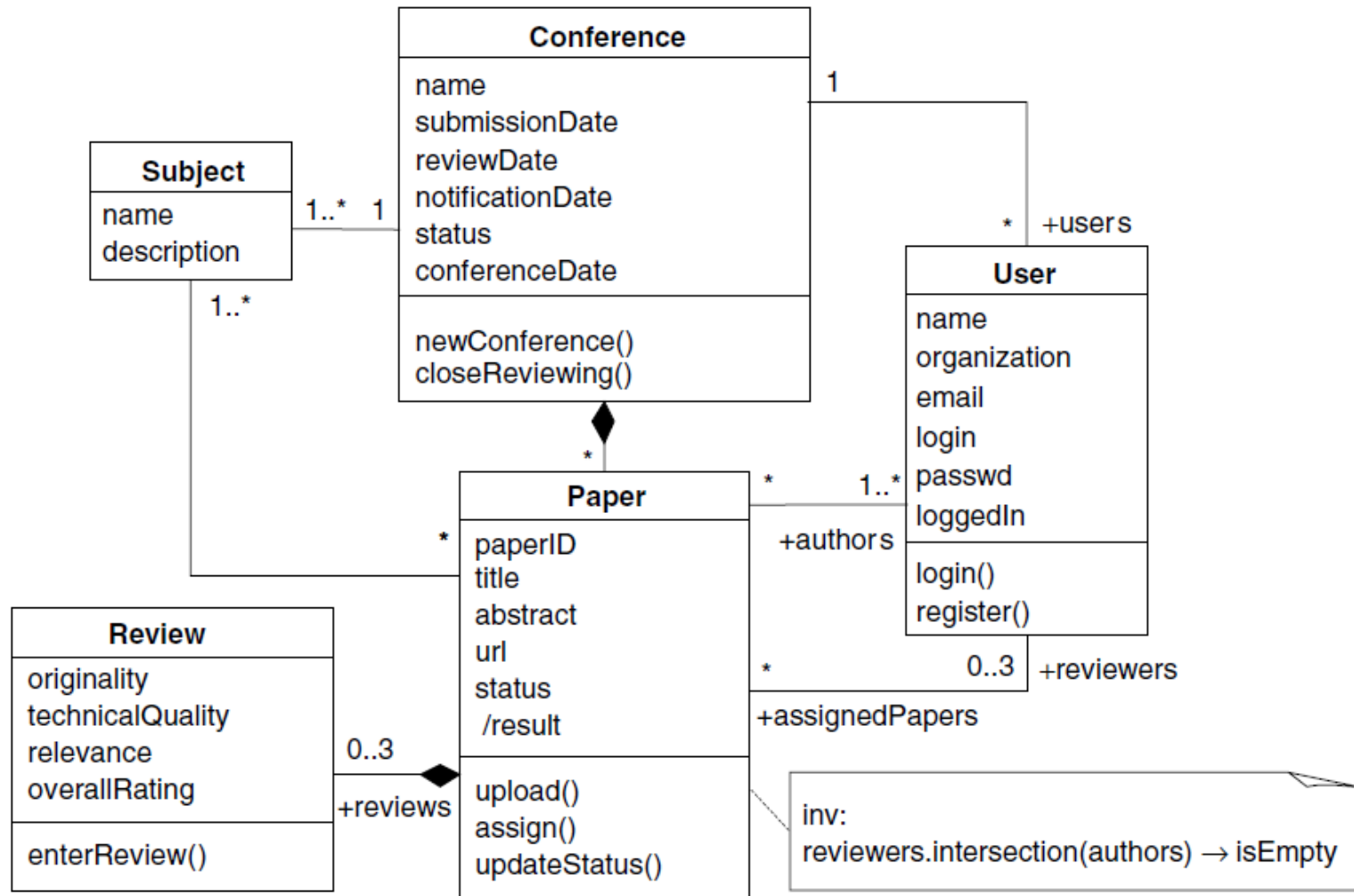
Modeling Support in UWE

- ☐ Requirements Modeling
- ☒ Content Modeling
- ☐ Hypertext modeling (navigation)
- ☐ Presentation modeling
- ☐ Customization

Content Modeling

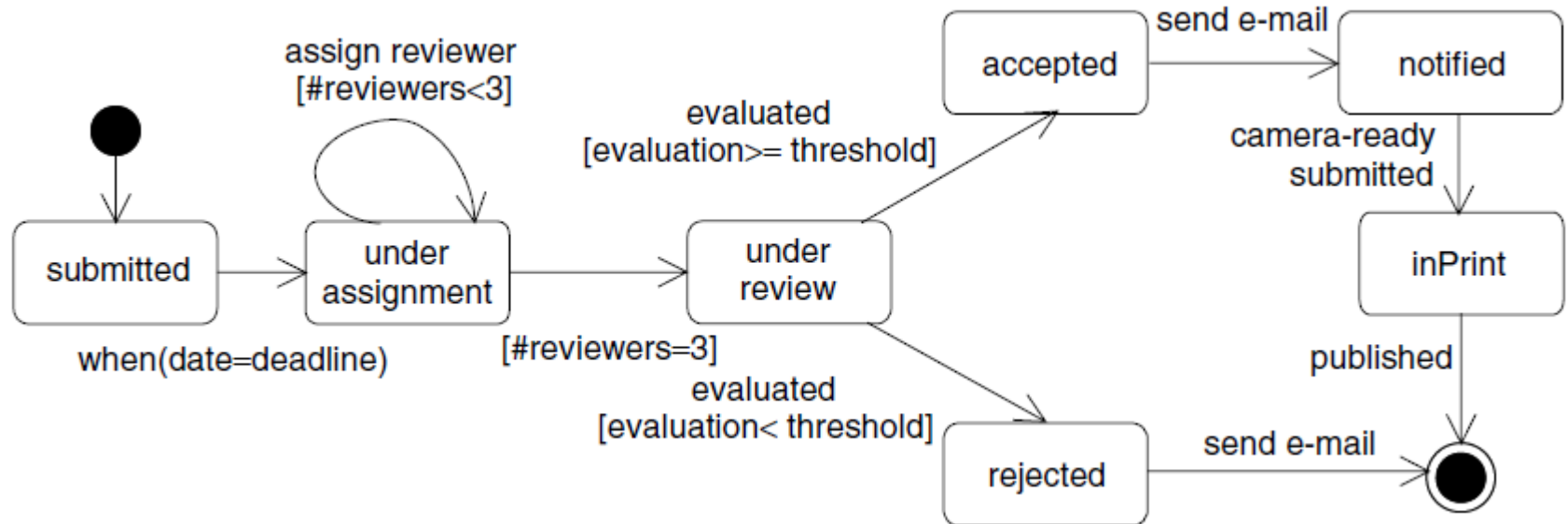
- ❑ Content modeling is aimed at transferring the information and functional requirements determined by requirements engineering to a model.
- ❑ Content modeling ***produces models capturing*** the structural (i.e., information objects) & behavioral aspects of the content of a web application
- ❑ Content modeling builds on the concepts of data modeling or object oriented modeling.
- ❑ Primary Models
 - Class or Entity Relationship (ER) diagrams – captures static aspects.
 - State machine diagrams, UML state charts – captures dynamic aspects.
- ❑ NOT concerned with navigation or presentation, *only content level*.
 - The class diagram will later serve as the basis to model the hypertext and the presentation for the example application.

Content Structure Model- Class Diagram



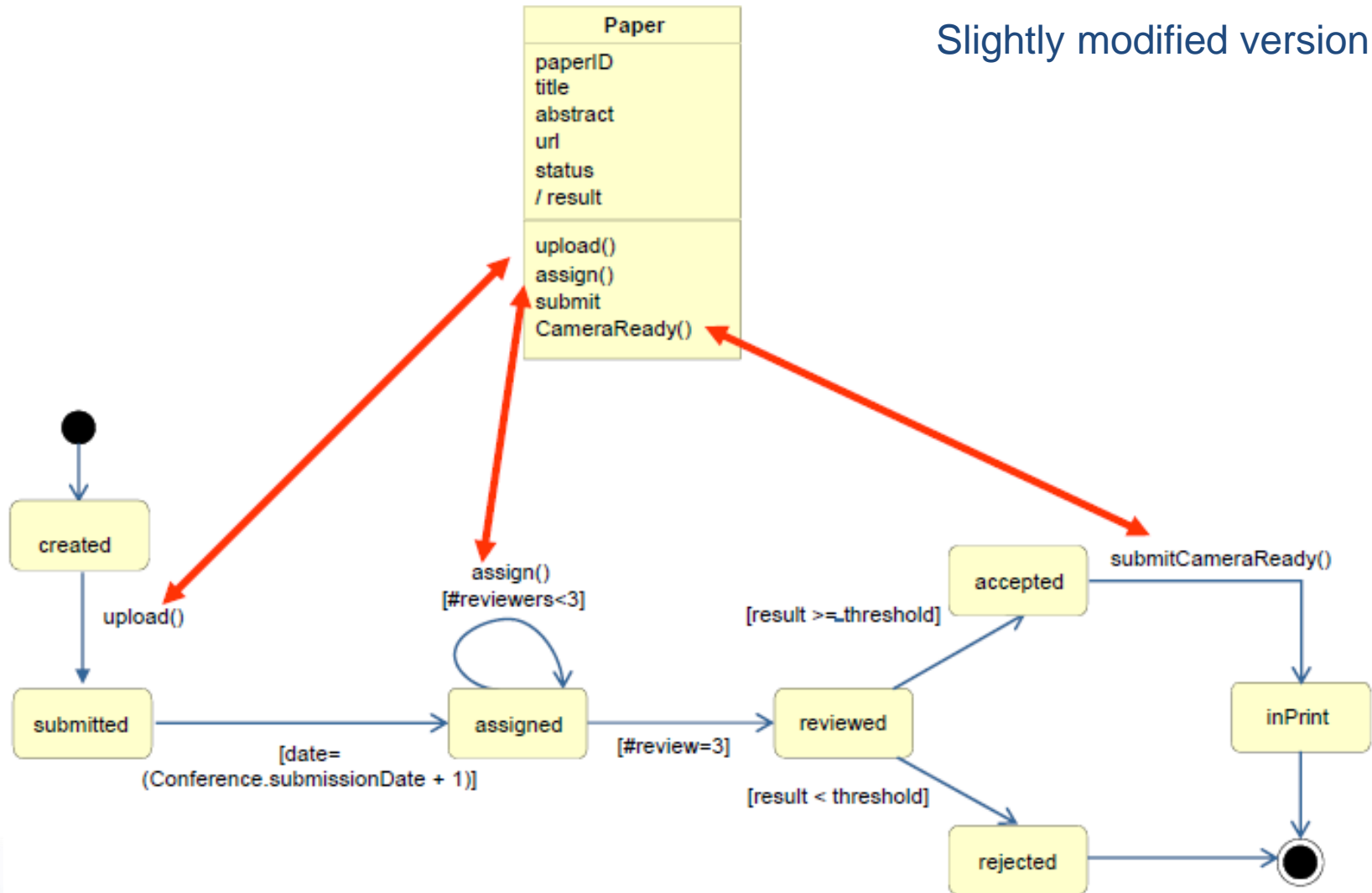
Content Behavior Model- State Machine Diagram

- ❑ For [dynamic Web applications](#), a SMD shows the life-cycle of an object.
 - depict important **states** and **events of objects**, and **how objects behave** in response to an event (transitions)
- ❑ Used only for state-dependent objects



State machine diagram for the states of a paper

Consistency with UML domain model



Modeling Support in UWE

- ☐ Requirements Modeling
- ☐ Content Modeling
- ☒ Hypertext modeling (navigation)
- ☐ Presentation modeling
- ☐ Customization

Hypertext (Navigation) Modeling

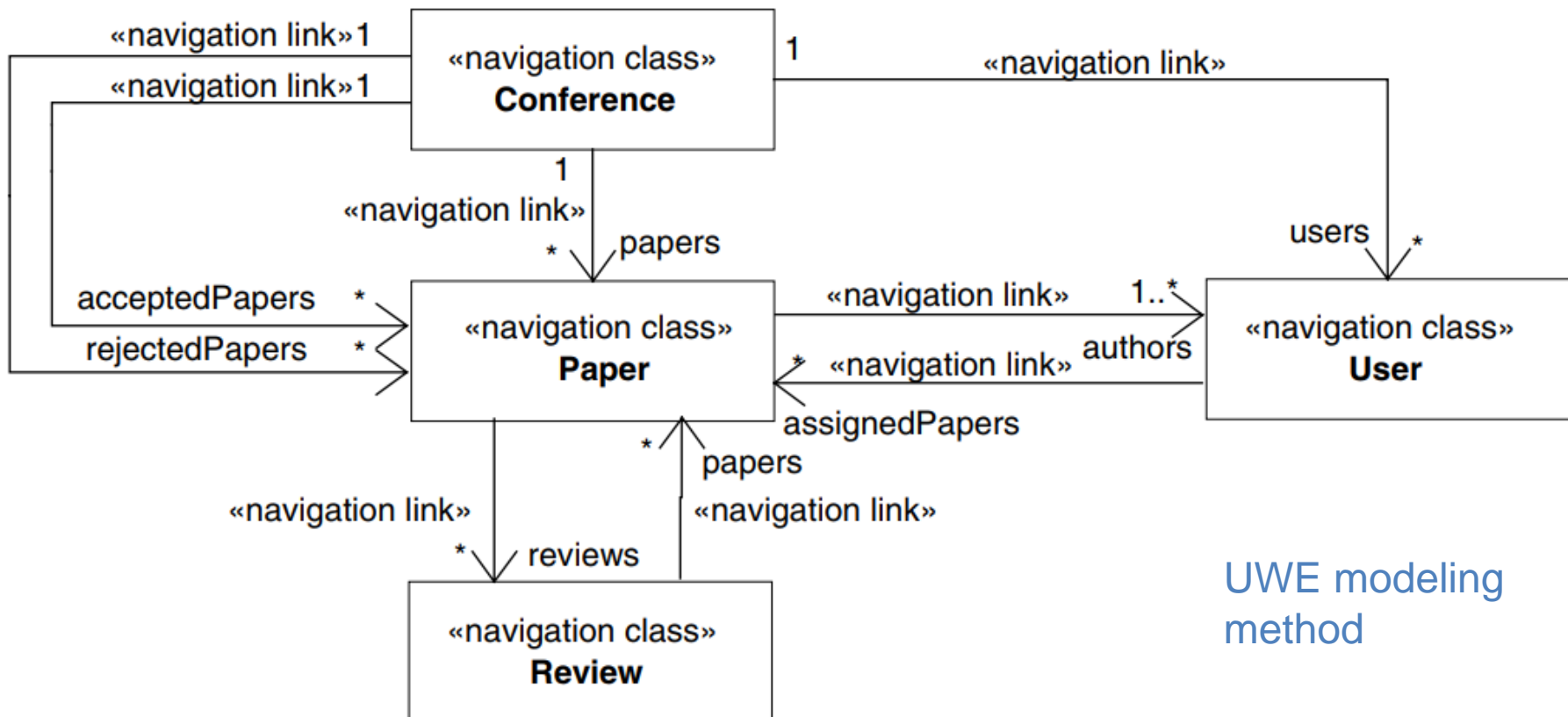
- ❑ To specify the **navigability through the content** of a Web application,
 - For simplicity: to model the **navigation paths** available to users.
- ❑ The **hypertext structure** has to be designed carefully.
 - to avoid the risk of users getting lost and
 - putting them under excessive cognitive stress
- ❑ Hypertext modeling generates two artifacts:
 - **Hypertext structure model** which defines the structure of the hypertext (i.e. navigation among classes)
 - Also called **navigation structure model** or **navigational view**
 - **Access model**- it refines the hypertext structure model by access elements
- ❑ >> Focuses on the **structure of the hypertext & access elements**.
- ❑ Use “<<navigation class>>” stereotype to distinguish from content classes.

Hypertext Structure Model

- ❑ Hypertext structure modeling is based on the **concepts of hypertext**, i.e., on **nodes** (also called pages or documents) and **links** between these nodes.
- ❑ Here, the starting point is usually the **content model** which contains the **classes** and **objects** to be made available as nodes in the hypertext.
 - Often the hypertext structure model is specified as a view on the content model and is therefore sometimes also called the **navigation view**.
 - a node is specified as a **view on the content model** selecting one or more objects from the content.
- ❑ Hypertext modeling concepts in UWE
 - «**navigation class**» for navigation nodes
 - «**navigation link**» for navigation links

Hypertext Structure Model

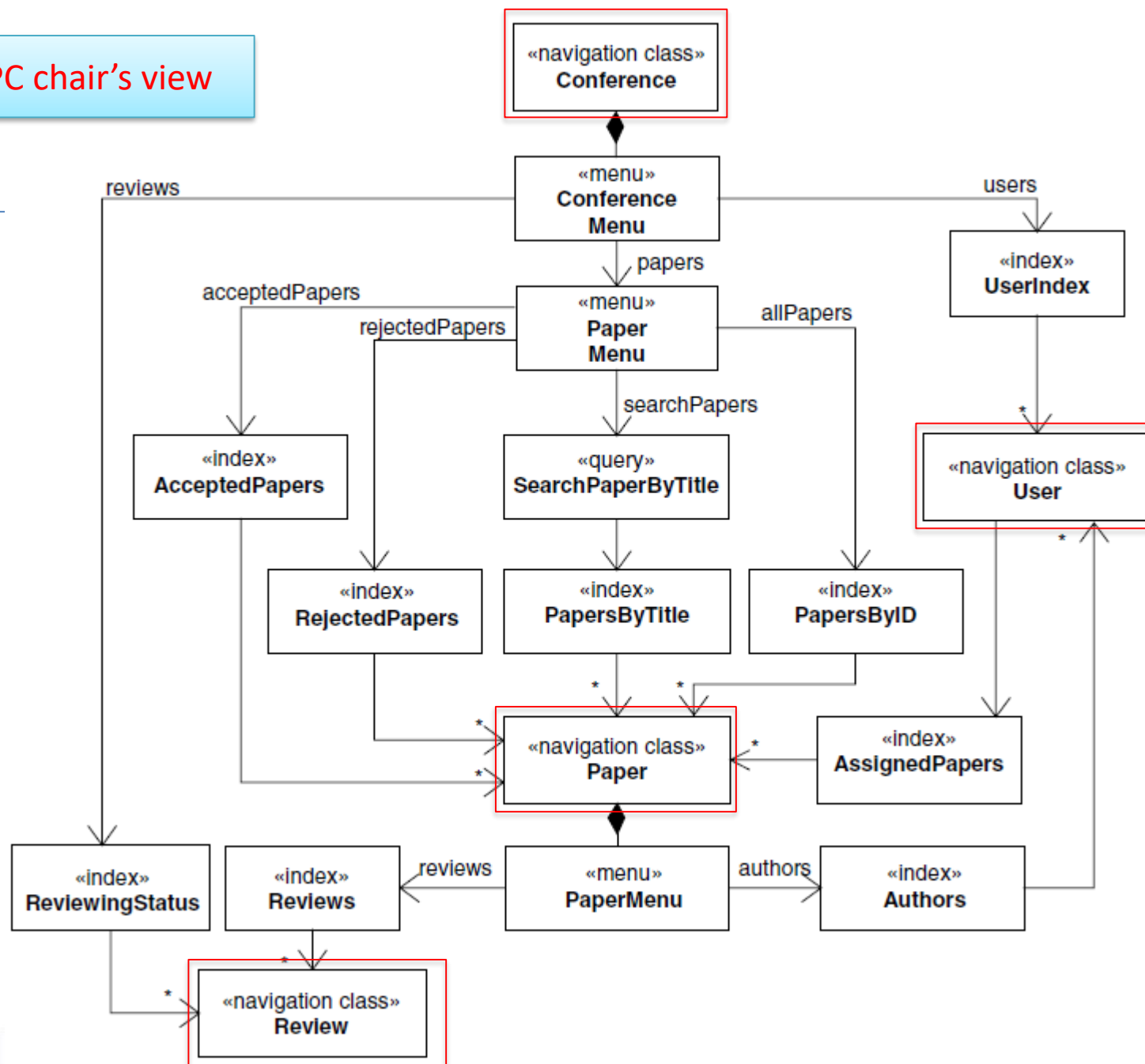
- ❑ Hypertext structure model of the PC's view on the reviewing system.



Access Model

- ❑ In the reviewing system example,
 - If one wants to navigate from a reviewer to a paper assigned to this reviewer, one will have to *identify this specific paper* during navigation.
 - this could be realized in the form of a list showing all papers. Such a selection list for navigational support is also known as an “*index*”.
- ❑ Hypertext structure models describe **navigation**, but **NOT orientation**.
- ❑ Access models describe both, through navigation patterns,
 - <<index>> - select a single object out of a homogeneous list
 - <<menu>> - allow access to heterogeneous nodes or other menus (submenus)
 - <<query>> - search for a node and direct access
 - <<guided-tour>> - allow users to sequentially walk through a number of nodes
- ❑ The use of these navigation patterns helps to increase the quality of the hypertext model extremely.

PC chair's view



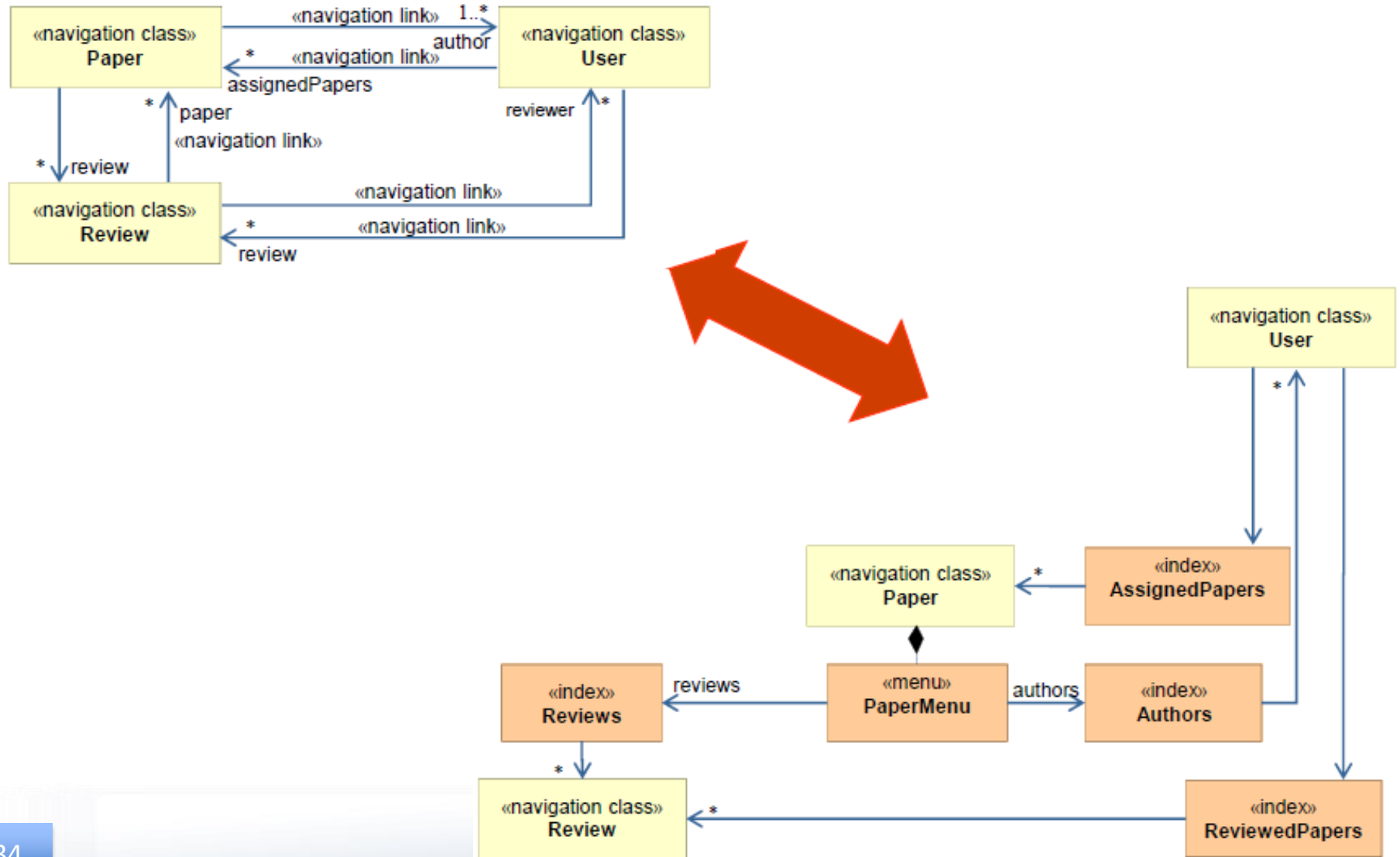
- ❑ A simplified access model of the PC chair's view specified in the hypertext structure model in the reviewing system.
 - Note that a link's default multiplicity is 1.
 - The PC chair has access to all papers, reviews, and users.
 - To access a specific paper, a unique number is used.
 - Alternatively, the PC chair can search for a paper by title.

- ❑ UWE uses UML stereotypes,
 - i.e., <<menu>> (e.g., "Conference"),
 - <<index>> (e.g., "ReviewingStatus"),
 - <<query>> (e.g., "SearchPaperByTitle"), and <<guided tour>>.

How to derive access Model from hypertext structure model

- ❑ Method to derive **access model** from **hypertext structure** model
 - introduce **index** for **all navigation links** with multiplicity >1
 - introduce **menu** for each class with **more than one** outgoing navigation link
 - use **role names** of outgoing navigation links as **menu items**

Consistency of hypertext structure model and access model



Modeling Support in UWE

- ☐ Requirements Modeling
- ☐ Content Modeling
- ☐ Hypertext modeling (navigation)
- ☒ Presentation modeling
- ☐ Customization

Presentation Modeling

- ❑ To model the **structure** and **behavior** of the **user interface**
- ❑ Aims at **simple**, **self-explanatory**, **consistent** interface when interacting with WebApp

- ❑ **Characteristics of presentation modeling**: Hierarchical composition of pages consisting of presentation elements

- ❑ Models the **structure** and **behavior** of user interface
 - **Composition & design of each page**, e.g., text, fields, forms, images, etc.
 - **Identify recurring elements** (e.g. headers/footers)
 - **Describe behavior oriented aspects** (events associated to elements)
 - **Design the graphical layout** for interface (a graphic designer)

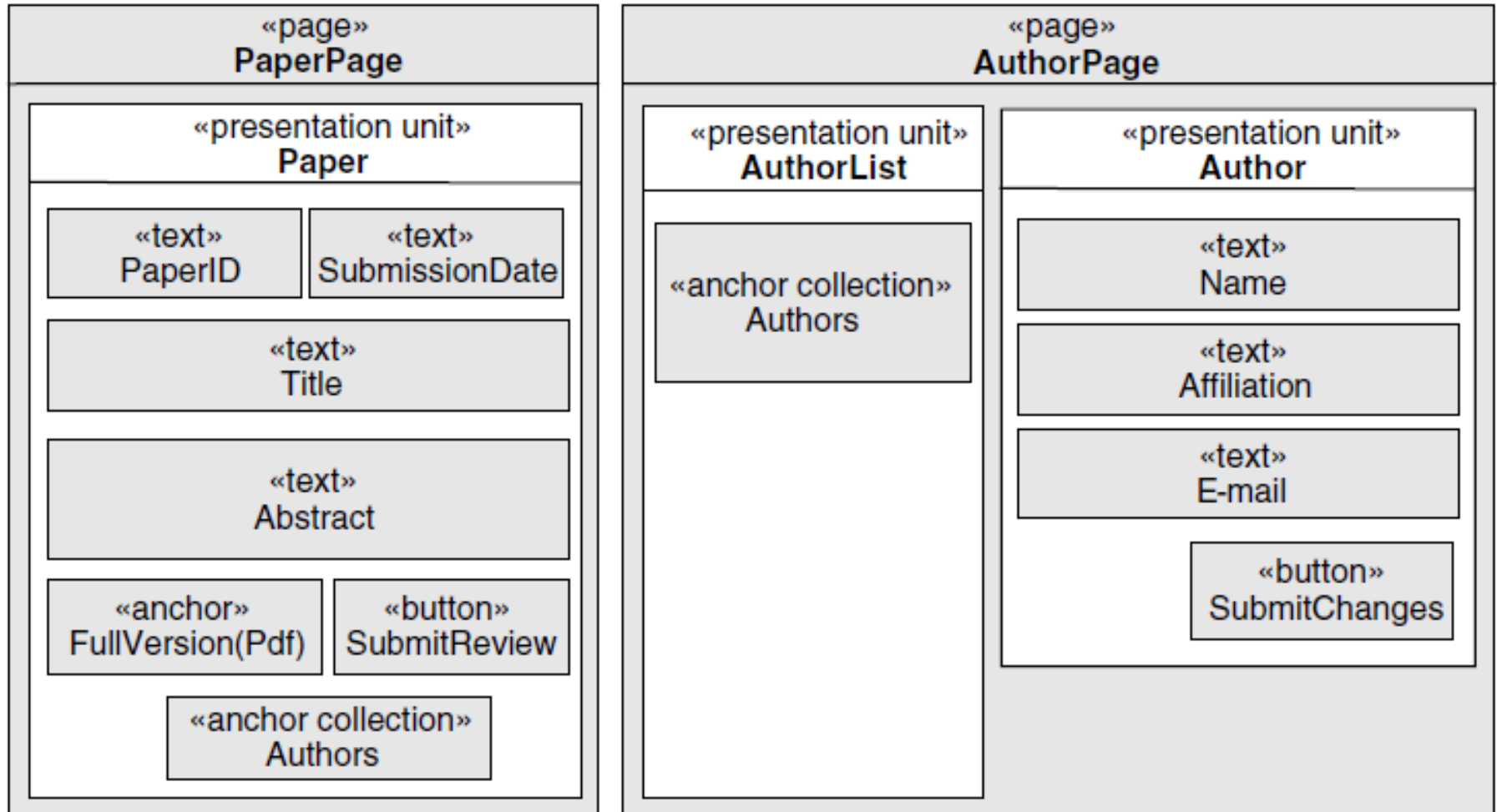
- ❑ **Resulted artifacts**:
 - **Static** presentation model
 - **Dynamic** interaction model

Levels of Presentation Models

- ❑ Composition of presentation pages is described on three hierarchical levels (using a nested UML class diagram)
 - **A Presentation Page** - describes a page presented to the user as a visualization unit (“root” element; serve as a container)
 - Indicated by <<page>>
 - **A Presentation Unit** (fragment of the page logically defined by **grouping** related elements)
 - Indicated by <<presentation unit>>
 - Represents a hypertext model node
 - **A Presentation Element** (basic building block and can include text, images, buttons, fields, etc)
 - such as <<text>>, <<anchor>>, <<image>>, <<button>>, etc

Presentation Modeling: Static

It shows two presentation pages of the reviewing system



Presentation Modeling: Behavior

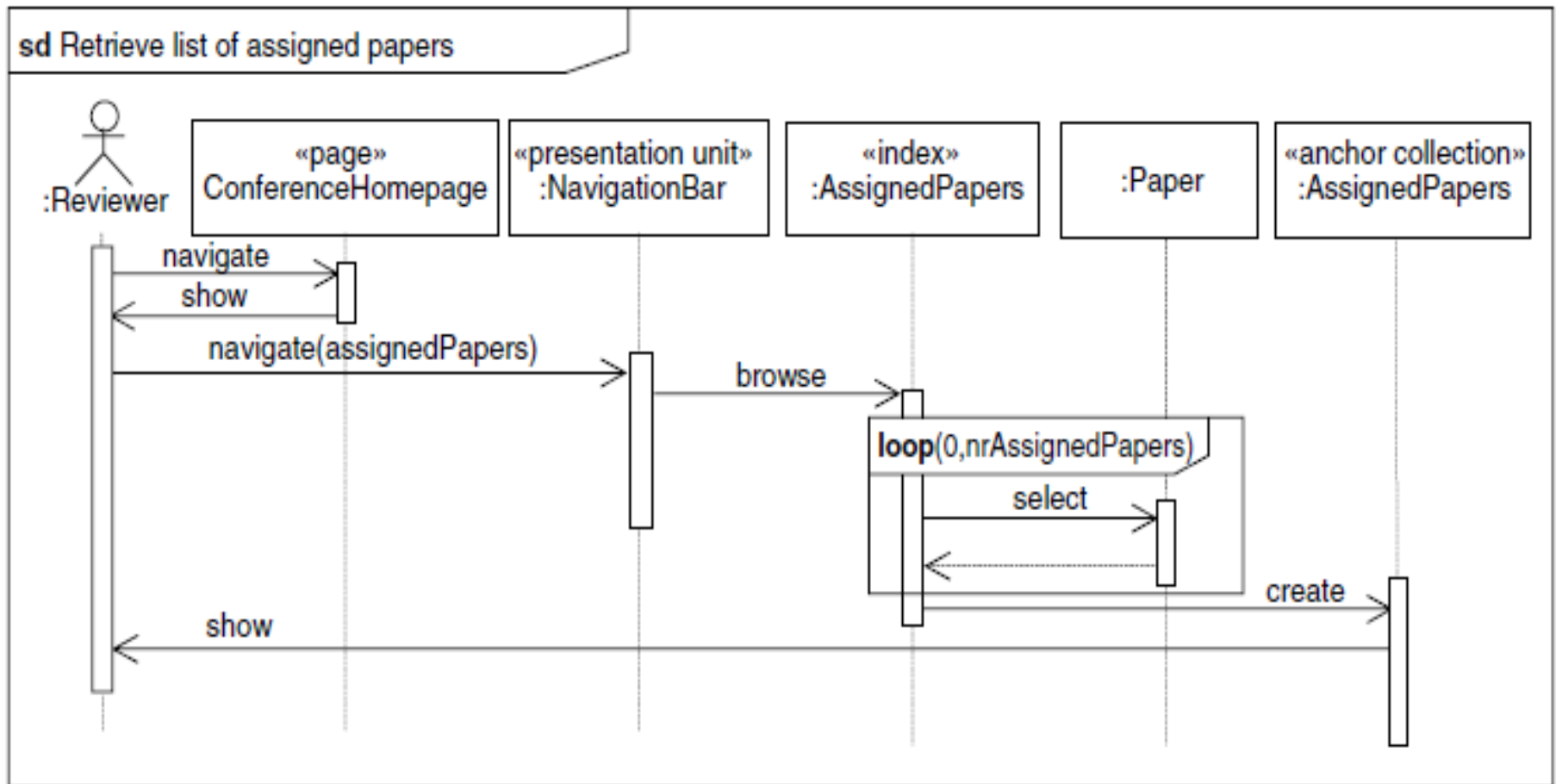
□ Behavioral aspects of the user interface

- such as a *reviewer's interaction* to navigate to the papers assigned to him for reviewing, can be modeled by means of behavior diagrams.

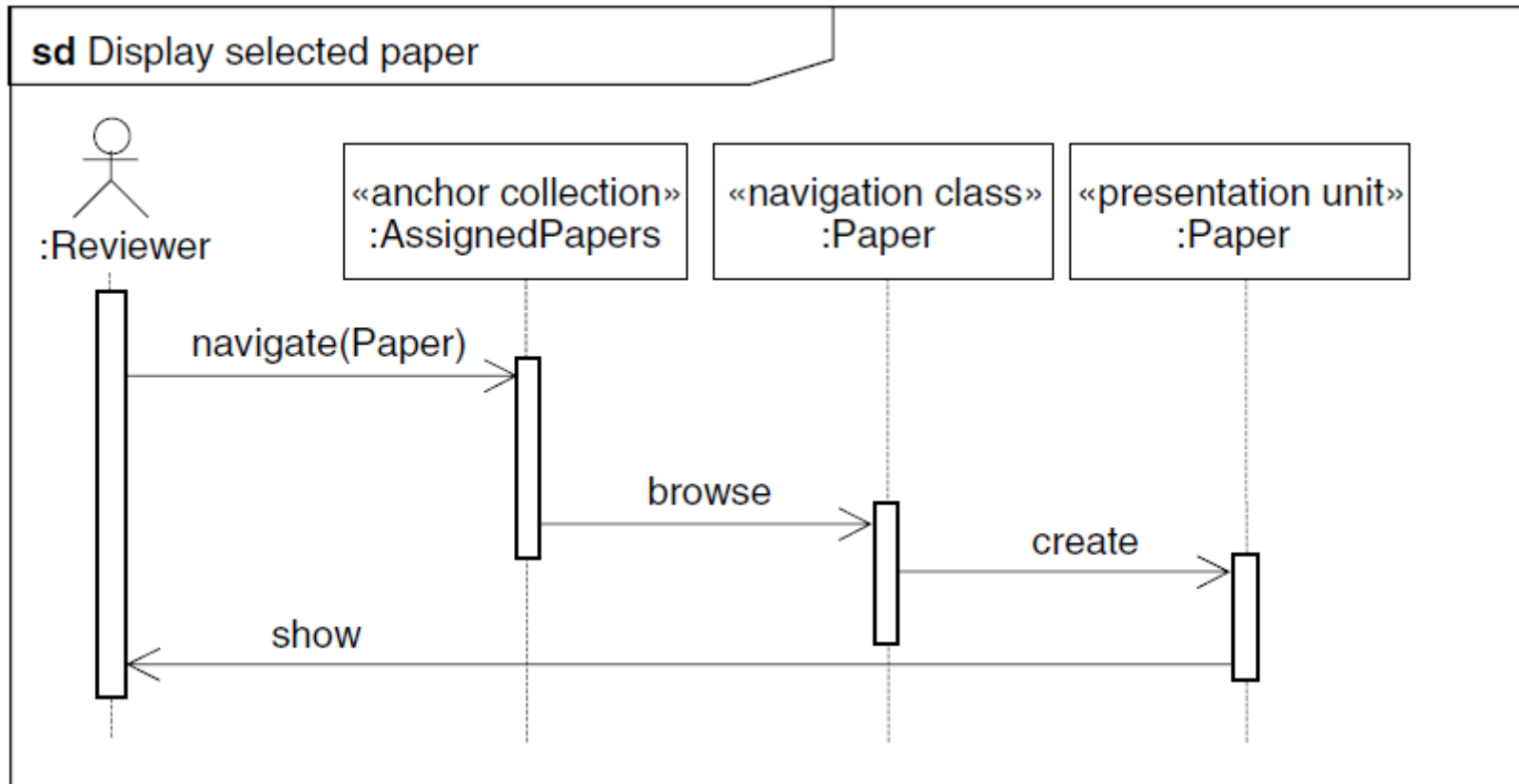
□ Sequence diagrams

- Depict **sequential interactions** (i.e., the flow of logic) **between objects** in an application over time.
 - What messages, what order, and to whom.
 - Ex.: Object A calls method of Object B
 - Ex.: Object B passes method call from Object A to Object C.

Sequence Diagram – Example 1



Sequence Diagram – Example 2



Modeling Support in UWE

- ☐ Requirements Modeling
- ☐ Content Modeling
- ☐ Hypertext modeling (navigation)
- ☐ Presentation modeling
- ☒ Customization modeling

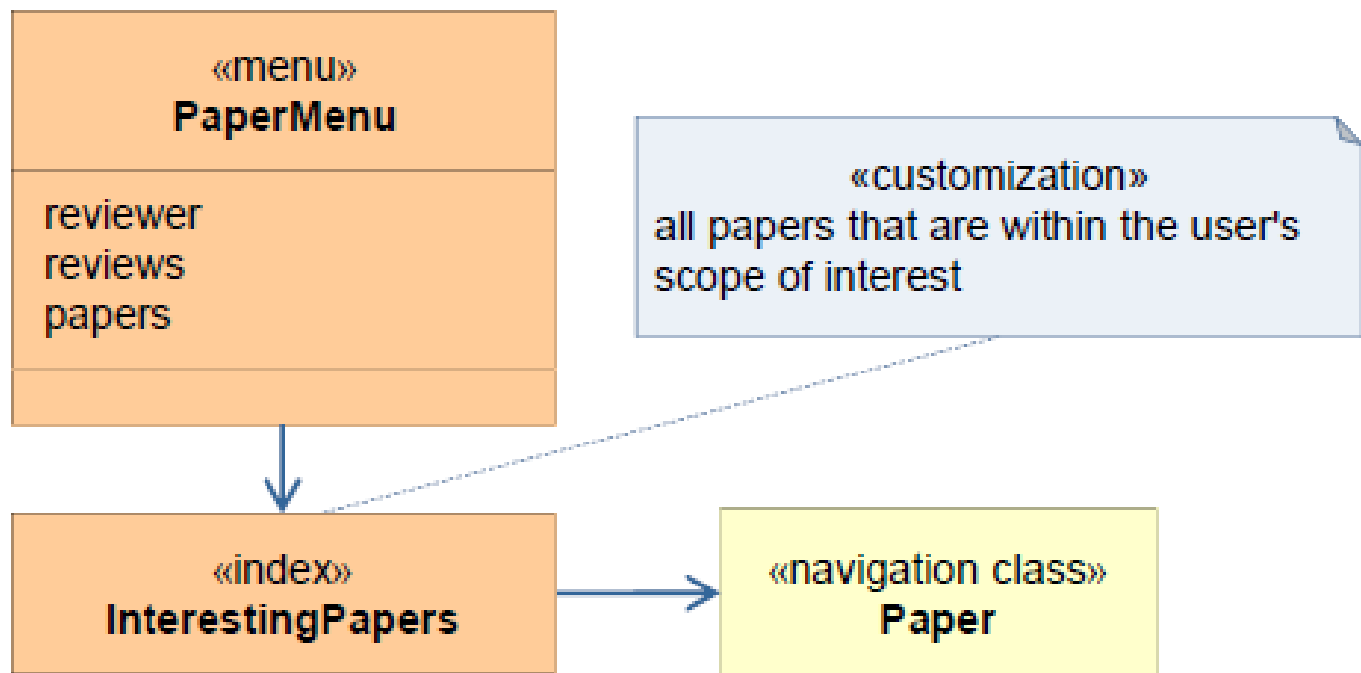
Customization Modeling

- ❑ **Objective:** Explicit representation of **context information**, and related implications on presentation
- ❑ **Origin:** the fields of **personalization** and **mobile computing**
- ❑ **Motivation:** Ubiquitous WebApps increasingly gain importance,
 - the consideration of **context information** and an appropriate **adaptation** of the application as early as possible in the modeling phase are required.
- ❑ **Different approaches**
 - **Static modeling:** different models for different context
 - **Dynamic modeling:** one model + adaptation rules
- ❑ **Result**
 - *Customization model*

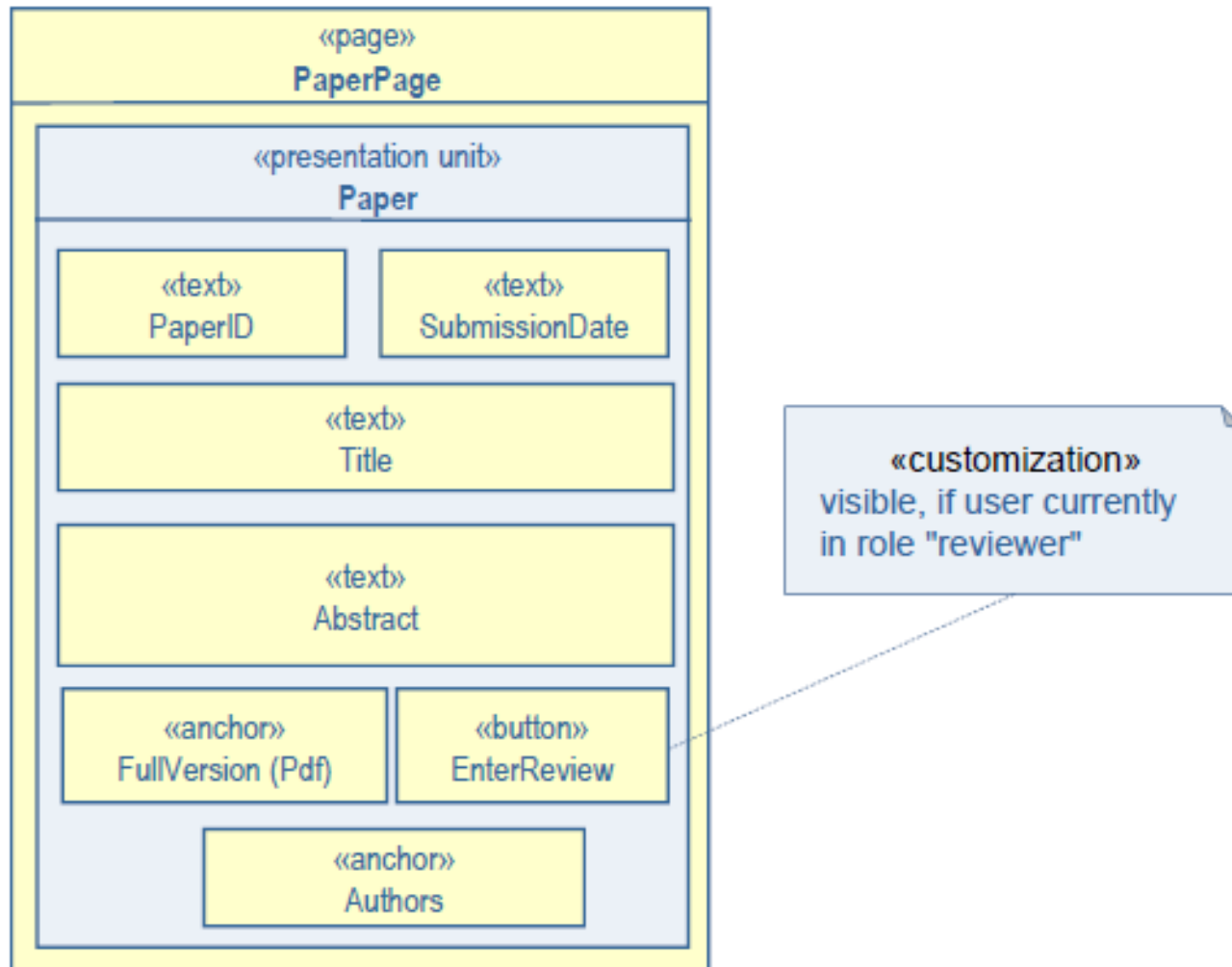
Customization Modeling

- ❑ Customization requires **examining the Web application's usage situation**,
 - i.e., dealing with the questions of “**what**” should be adapted and “**when**”.
- ❑ To **personalize a Web application**
 - You need to model and manage the preferences and characteristics of a user in a so-called *user profile*.
- ❑ For example, to adapt a Web application in the field of mobile computing, we have to consider *device profiles*, *location information*, and *transmission bandwidth*.
 - This information is then represented within the context model in form of a class diagram.
- ❑ Customization is **different** from maintenance or re-engineering.
 - Customization modeling considers context information that **can be predicted** at modeling time which can assume different values when the Web application is run.
 - In contrast, **adaptation** due to changes in the organizational or technological environment is part of maintenance or re-engineering activities.

Dynamic adaptation of an index in the hypertext model



Dynamic adaptation of a page in the presentation model



Interface Design Guidelines I

- ❑ Should optimize the user's work
- ❑ Should be designed to **minimize** the learning time (especially when the application is revisited)
- ❑ Should provide an indication of the **current location** in the content hierarchy
- ❑ Should always help the user **understand his current options**: available functions, relevant content, active links, etc.
- ❑ Should **facilitate navigation**, e.g. provide a site map
- ❑ Should be **consistent** in using navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)

Interface Design Guidelines II

- ❑ All information presented through the interface should be **readable** by young and old
- ❑ Should communicate the status of any activity initiated by the user
- ❑ Whenever appropriate, the state of the user interaction should be **tracked** and stored so that a user can logoff and return later to pick up where she left off.
- ❑ Should use **multi-tasking** in a way that lets the user proceed with work as if the operation has been completed.

Web App Architecture

Developing Architectures

- ❑ **Architecture** describes the structure: **components, their interfaces** and **relationships**
- ❑ **Benefits:**
 - Architecture has **considerable influence** on the **quality** of the web application
 - It makes the **system more understandable** to better manage its complexity
- ❑ **Inappropriate architecture** can lead to
 - Poor performance
 - Low availability
 - Insufficient maintainability and expandability
- ❑ Successful architecture should consider
 - the use of **multi-tier architectures**
 - **integration** with existing systems: web servers, application servers, data repositories, etc.

Developing Architectures

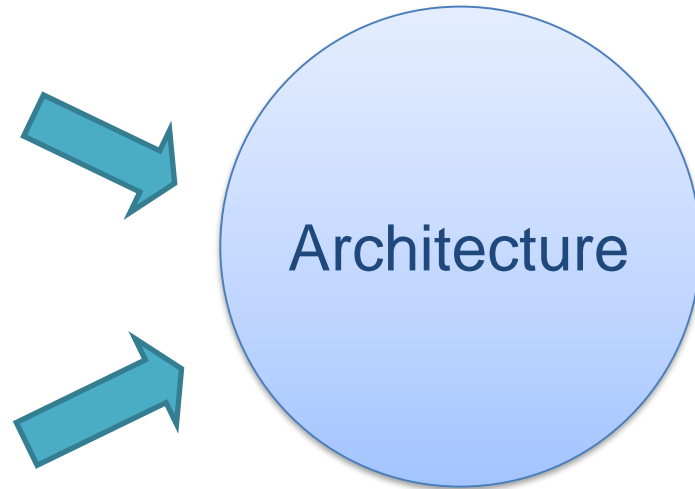
- ❑ Different factors and constraints that has influences on the development of an architecture

Functional Requirements

- Clients
- Users
- Other Stakeholders

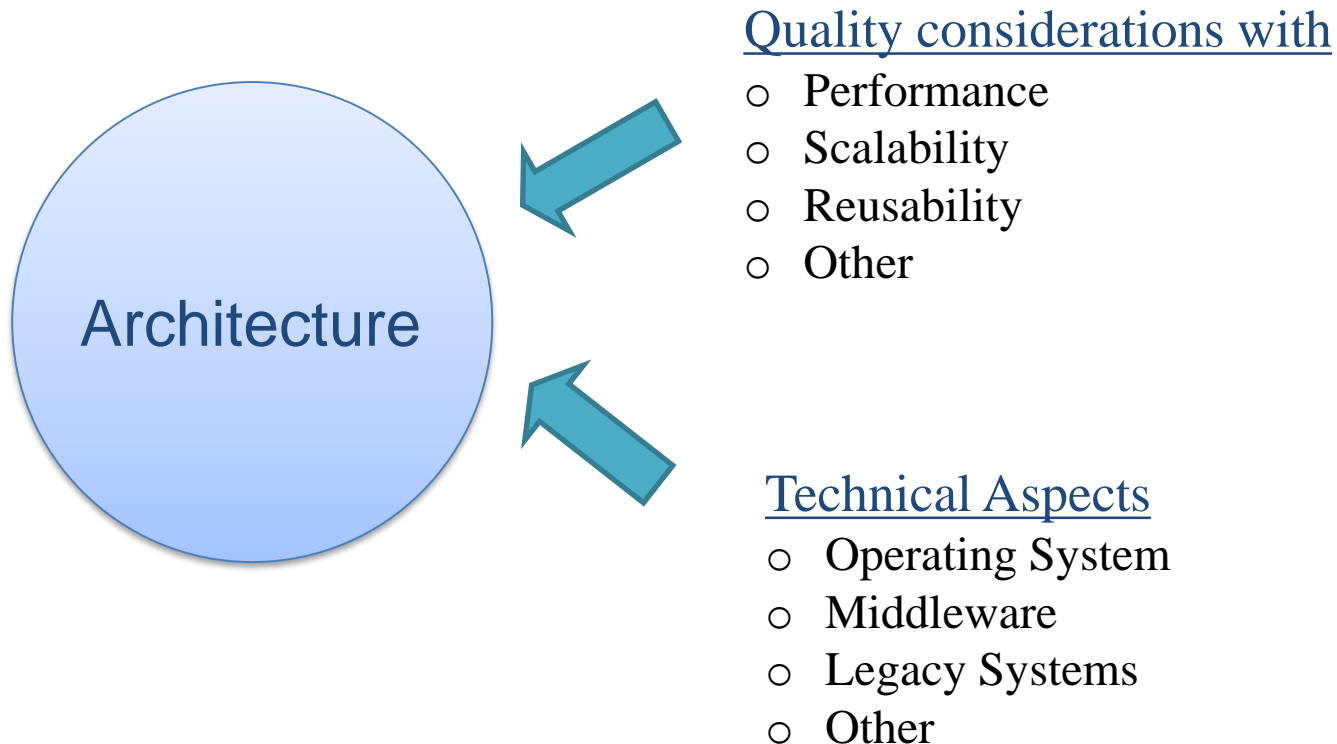
Experience with

- Existing Architecture
- Patterns
- Project Management
- Other



Developing Architectures

- ❑ Different factors and constraints that has **influences on the development of an architecture**

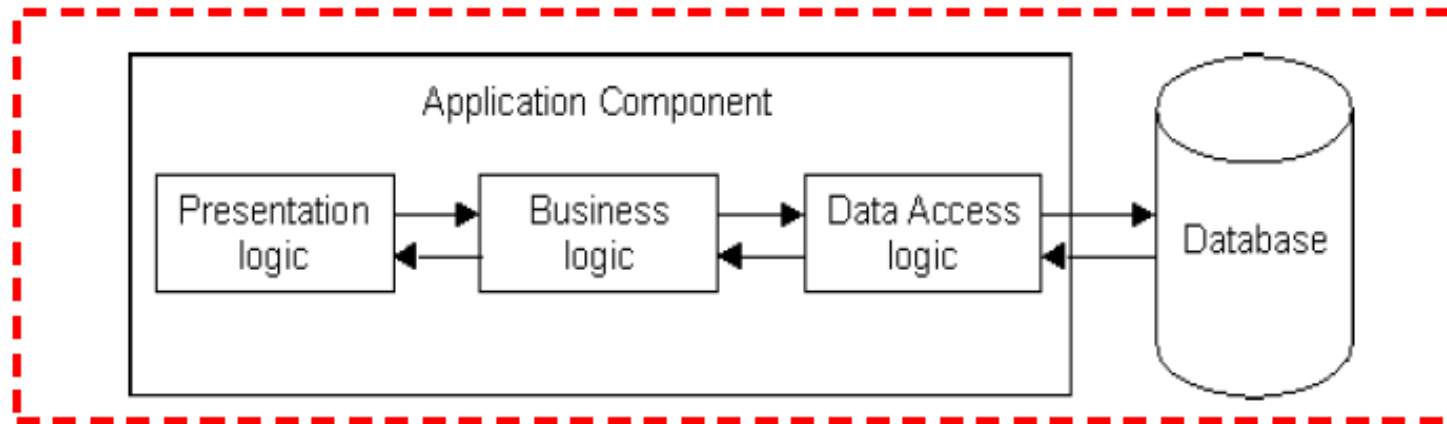


Significance of “Tiers”

- ❑ N-tier architectures, is also called multi-tier architecture, have the same components
 - Presentation
 - Business Logic
 - Data management

- ❑ N-tier architectures try to separate the components into different tiers/layers
 - Tier: physical separation
 - Layer: logical separation

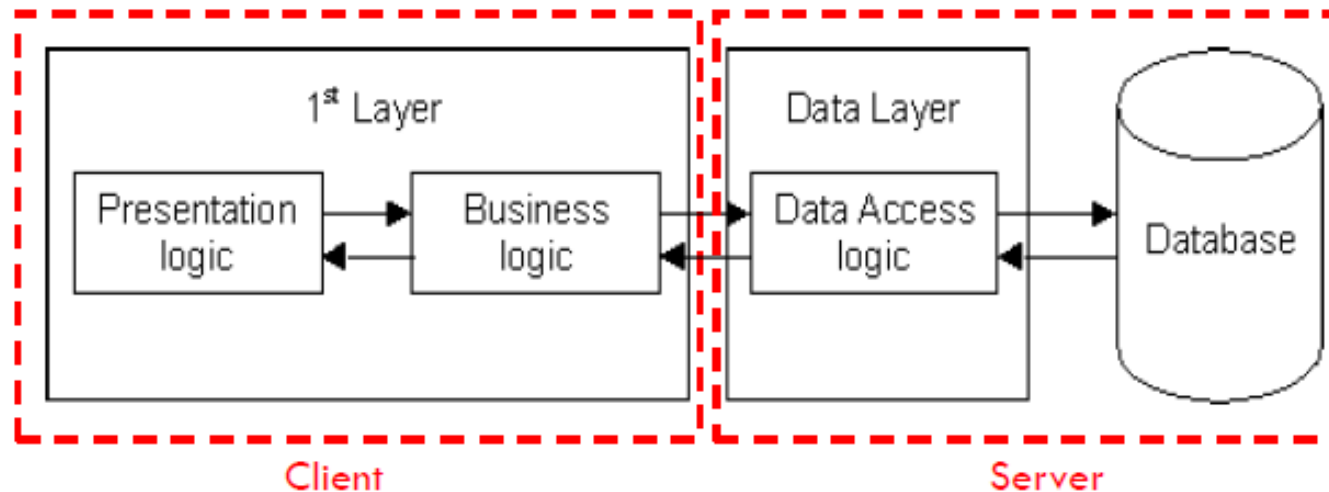
1-tier Architecture



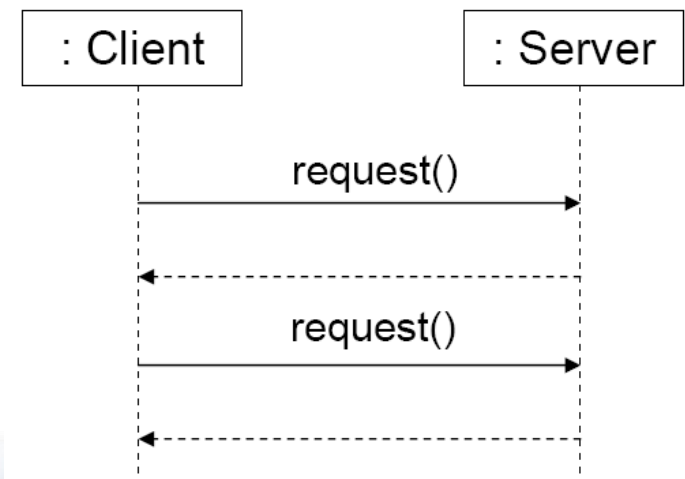
- ❑ All 3 layers are on the same machine
 - All code and processing kept on a single machine
 - Presentation, Logic, Data layers are tightly connected
- ❑ **Scalability:** Single processor means hard to increase volume of processing
- ❑ **Portability:** Moving to a new machine may mean rewriting everything
- ❑ **Maintenance:** Changing one layer requires changing other layers

2-tier Architecture

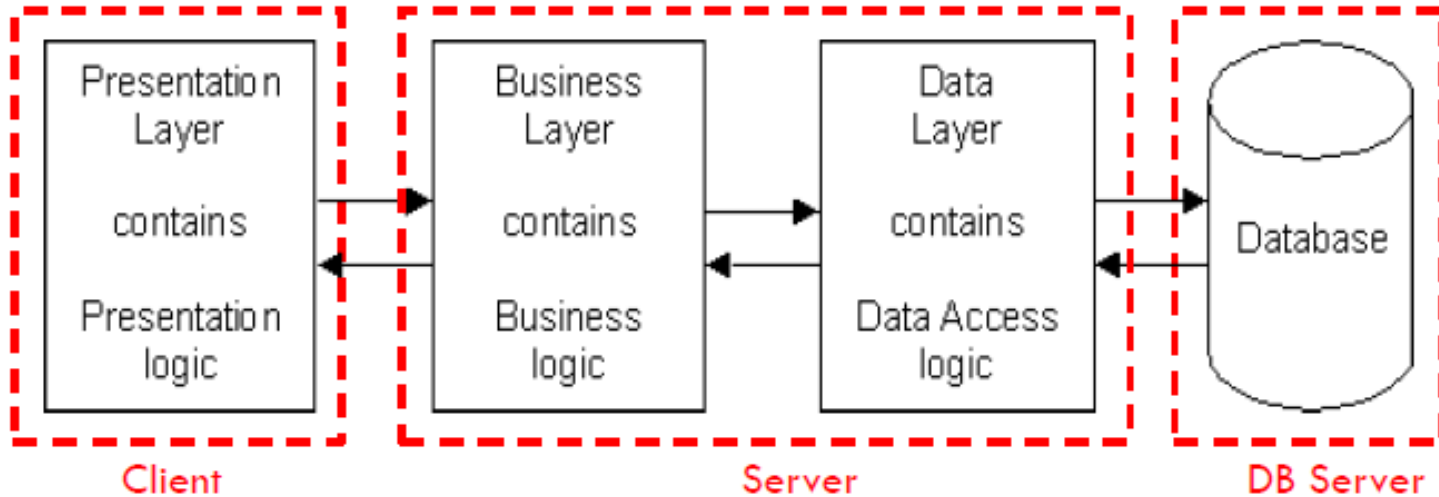
Client/Server



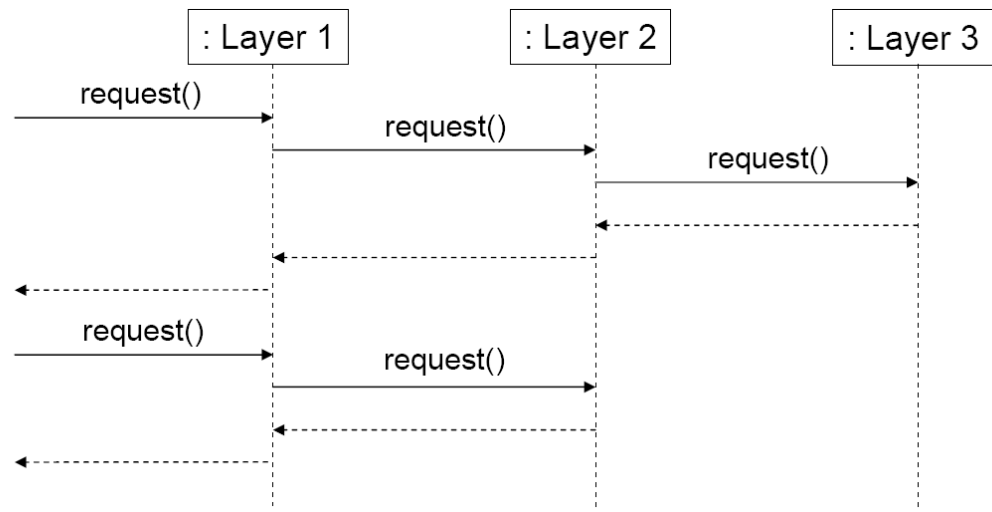
- ❑ Database runs on Server
 - Separated from client
 - Easy to switch to a different database
- ❑ Presentation and logic layers still tightly connected
 - Heavy load on server
 - Potential congestion on network
 - Presentation still tied to business logic



3-tier Architecture



- ❑ Each layer can potentially **run on a different machine**
- ❑ Presentation, logic, data layers **disconnected**



A Typical 3-tier Architecture

❑ Presentation Layer

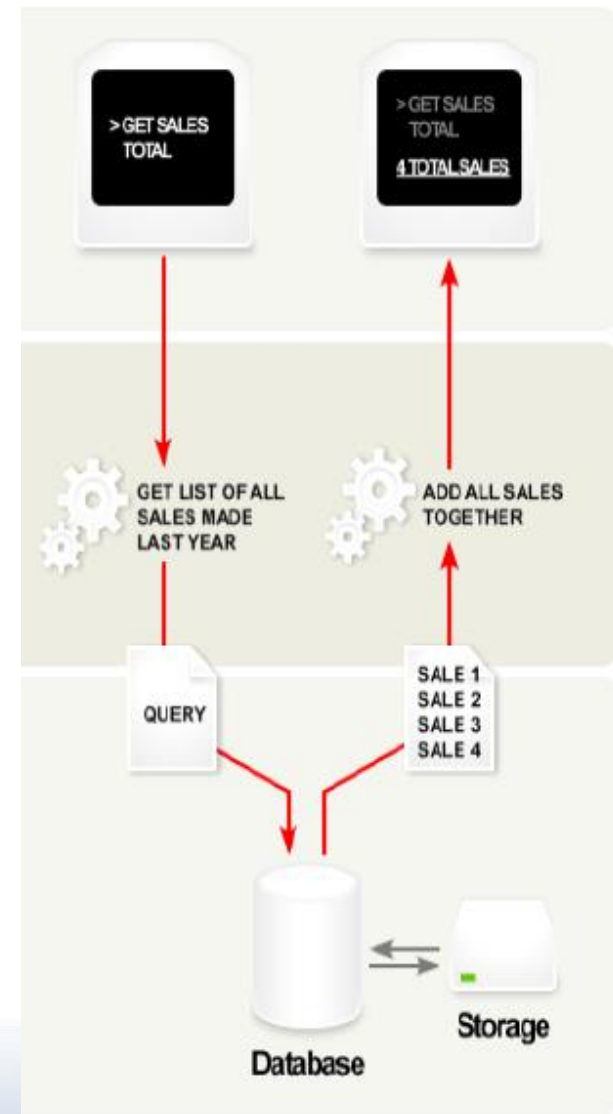
- Provides user interface
- Handles the interaction with the user
- Sometimes called the GUI or client view or front-end
- Should not contain business logic or data access code

❑ Logic Layer

- The set of rules for processing information
- Can accommodate many users
- Sometimes called middleware/ back-end
- Should not contain presentation or data access code

❑ Data Layer

- The physical storage layer for data persistence
- Manages access to DB or file system
- Sometimes called back-end
- Should not contain presentation or business logic code



A Typical 3-tier Architecture

Architecture Principles:

- ❑ Client-server architecture
- ❑ Each tier (Presentation, Logic, Data) should be **independent** and should not expose dependencies related to the implementation
- ❑ Unconnected tiers should not communicate

Advantages of 3-tier Architecture

- ❑ Independence of Layers
 - Easier to maintain
 - Components are reusable
 - Faster development (division of work)
 - Web designer does presentation
 - Software engineer does logic
 - DB admin does data model
 - Change in **platform affects only the layer running on that particular platform**

