Introduction to Web Engineering SENG2050/6050

Web Engineering Design

Overview

- From Requirements Analysis to Design
- Web Modelling
  - Hypertext Structure
  - Access Model
  - Adaption Model

Traditional Software Engineering (TSE) Analysis and Design

- The analysis and design activities help on transforming the requirements of the system into a design that can be realized in software
- Analysis comprises those activities that take the use cases and functional requirements to produce an analysis model of the system.
- The analysis model is made up of classes and collaborations of classes that exhibit the dynamic behaviors detailed in the use cases and requirements.
- Analysis focuses on the functional requirements of the system, ignoring for the moment the architectural constraints of the system.
- The emphasis is on ensuring that all the functional requirements, as expressed by the use cases and other documents, are realized somewhere in the system.

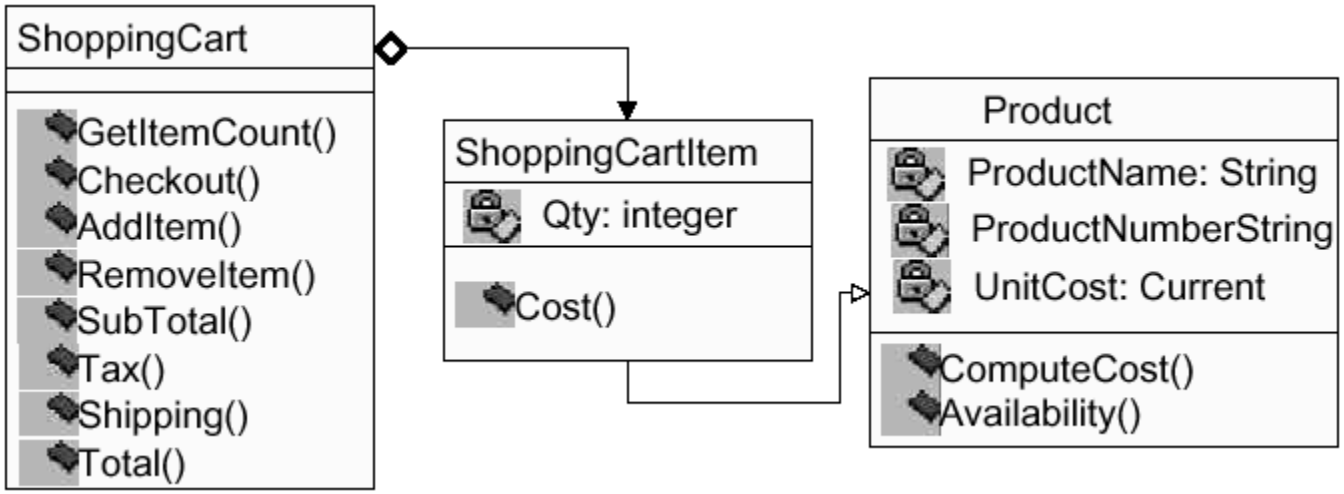
TSE Analysis

- First thing first — create the package hierarchy of the analysis model
- A package is nothing more than a “chunk” of the model: small enough that one can understand its purpose and significance in the model.
- A package contain elements of the model: classes, diagrams and interfaces.

TSE Analysis

In each package:

- Identify the classes and objects that will perform a use case's flow of events.
- Identify the responsibilities, attributes and associations of the classes.



TSE Analysis

How to identify classes and objects?

- noun/verb analysis
- Other techniques include:
  - Brainstorming
  - Role playing different “parts” of the system
- Output analysis:
  - Analysis level objects, and
  - Use case realizations consists of sequence diagrams, class diagrams and textual descriptions

TSE Design

- Design is primarily a refinement of the analysis model.
- Design uses the non-functional requirements of the system and the constrains of the architecture to refine the analysis model into something that can be coded.

Web Modelling

- Modelling static & dynamic aspects of content, hypertext, and presentation.
- Extra things need to model
  - Navigation (hypertext structure)
    - Good navigation structure of an application helps user to find relevant information fast and avoids him to be “lost” in hyperspace
  - Adaption
    - Makes Web System more suitable for individual users
    - Based on: *User’s platform; User’s preferences*; *User’s behaviour.*
  - Access structure, presentation structure Etc.

Unified Modelling Language (UML)

- "The Unified Modelling Language is a visual language for specifying and documenting the artifacts of systems." [OMG03a]
- Language of choice (and ISO standard) for *diagramming notation* in OO development.
  - *Structural* – Class diagrams (domain models)
  - *Behavioral* – Use Cases, Sequence diagrams, activity diagram
- Currently at version 2.5 (http://www.omg.org/spec/UML/2.5/PDF/), although many analysts and designers still use 1.0.

UML for Web Engineering

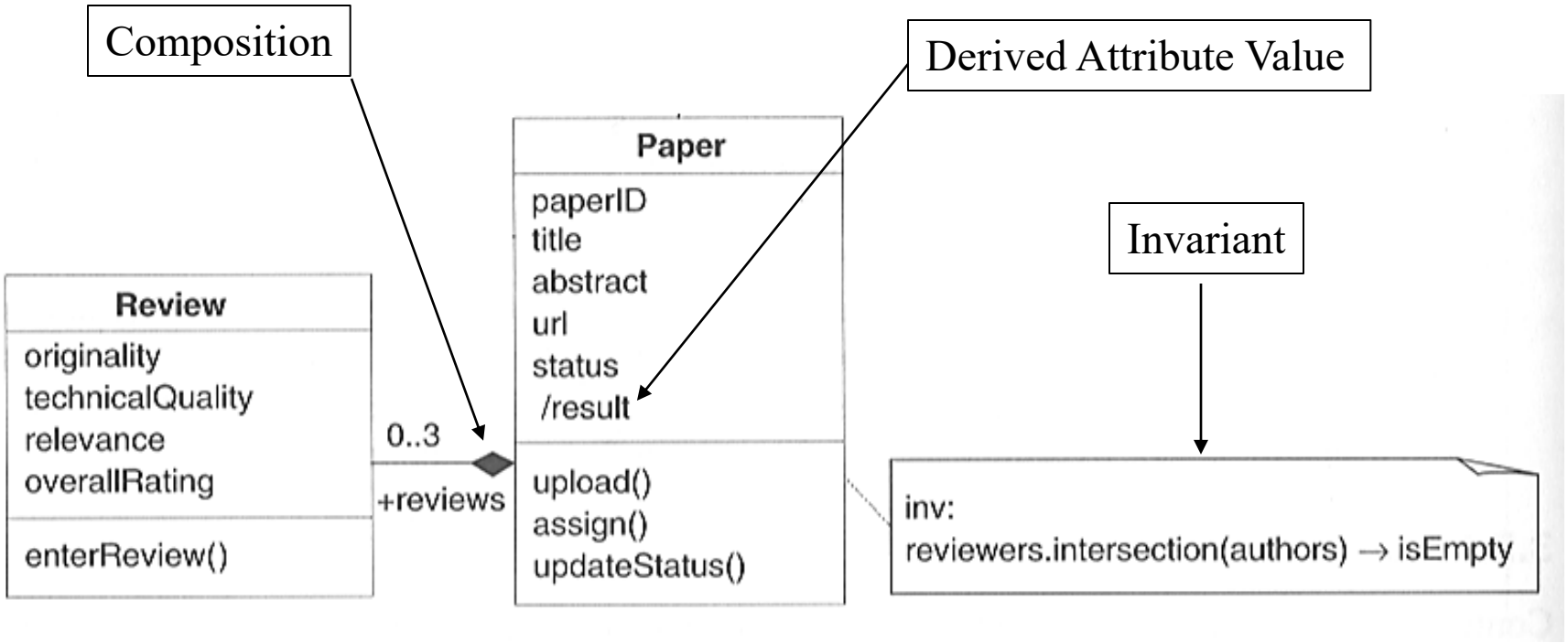
- For Web-centric modelling, we will employ the UML Web Engineering (UWE) notation.
  - http://uwe.pst.ifi.lmu.de/
  - UML-compliant comprehensive modelling tool
- Not the only modelling language
  - WebML, OOHDM, etc.
- We assume you have some basic knowledge of UML diagram

Content Modelling

- *Purpose*: To model the information requirements of a Web application
  - Diagramming the structural (i.e., information objects) & behavioral aspects of the information.
  - NOT concerned with navigation.
- Primary Models
  - Class diagrams – enough for static applications.
  - State machine diagrams – captures dynamic aspects

Class Diagrams

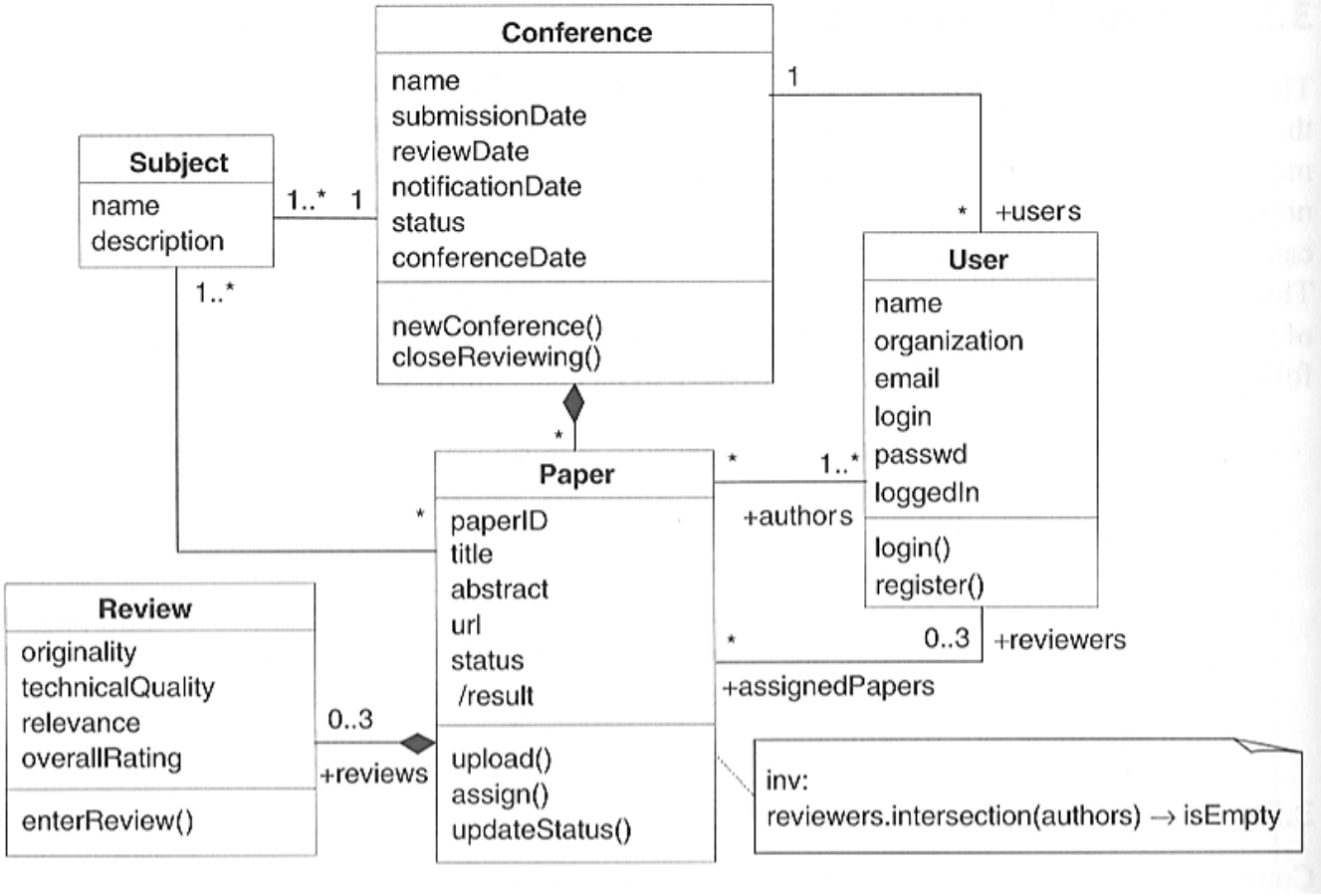
- Notations



Source: Web Engineering – Kappel et al.

Class Diagram – Example 1

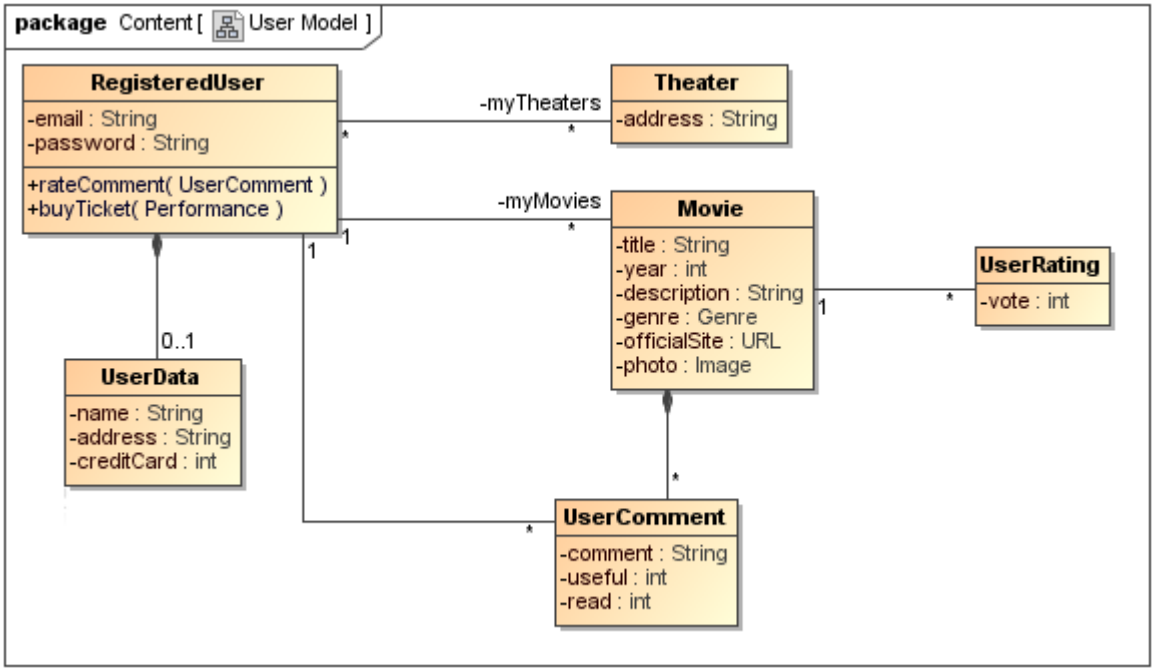
- Conference Paper Submission System



Source: Web Engineering – Kappel et al.

Class Diagram – Example 2

- IMDB



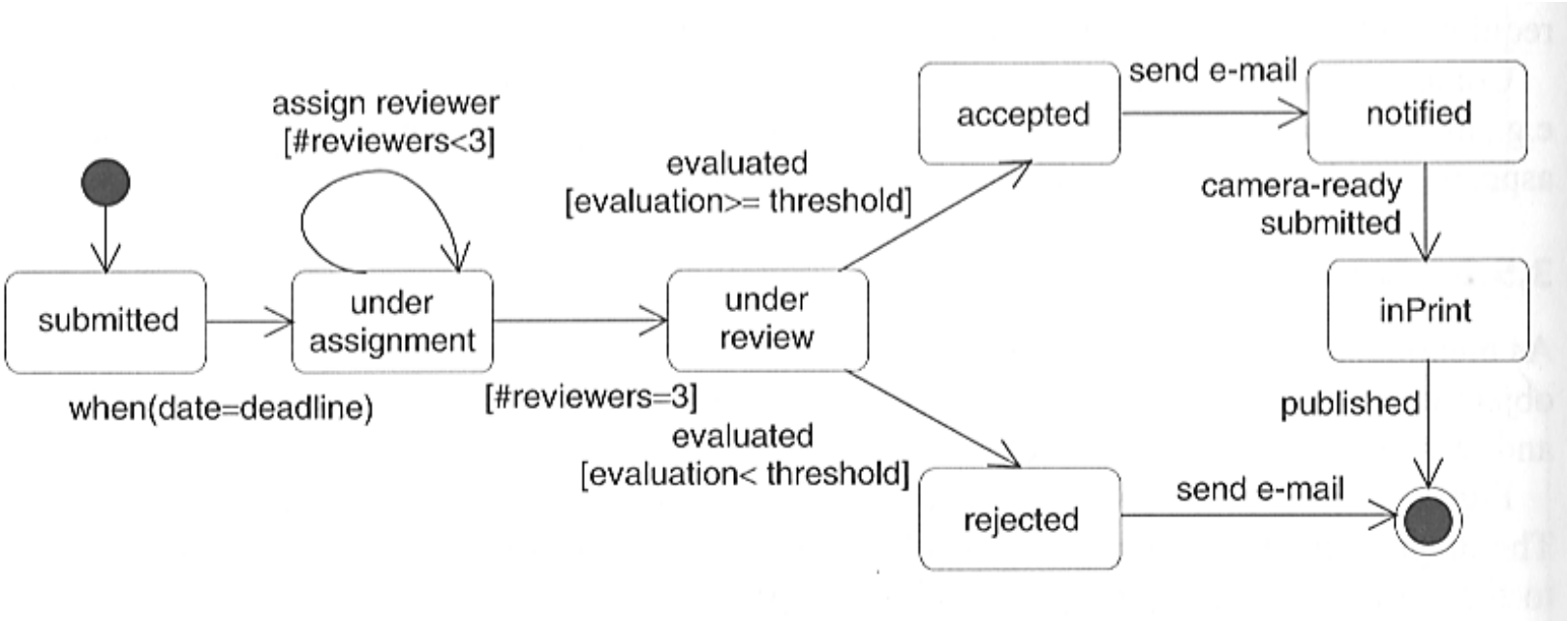
http://uwe.pst.ifi.lmu.de/exampleIMDB.html

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

State Machine Diagram - Example

- The life-cycle of a Paper object in the conference paper submission system.



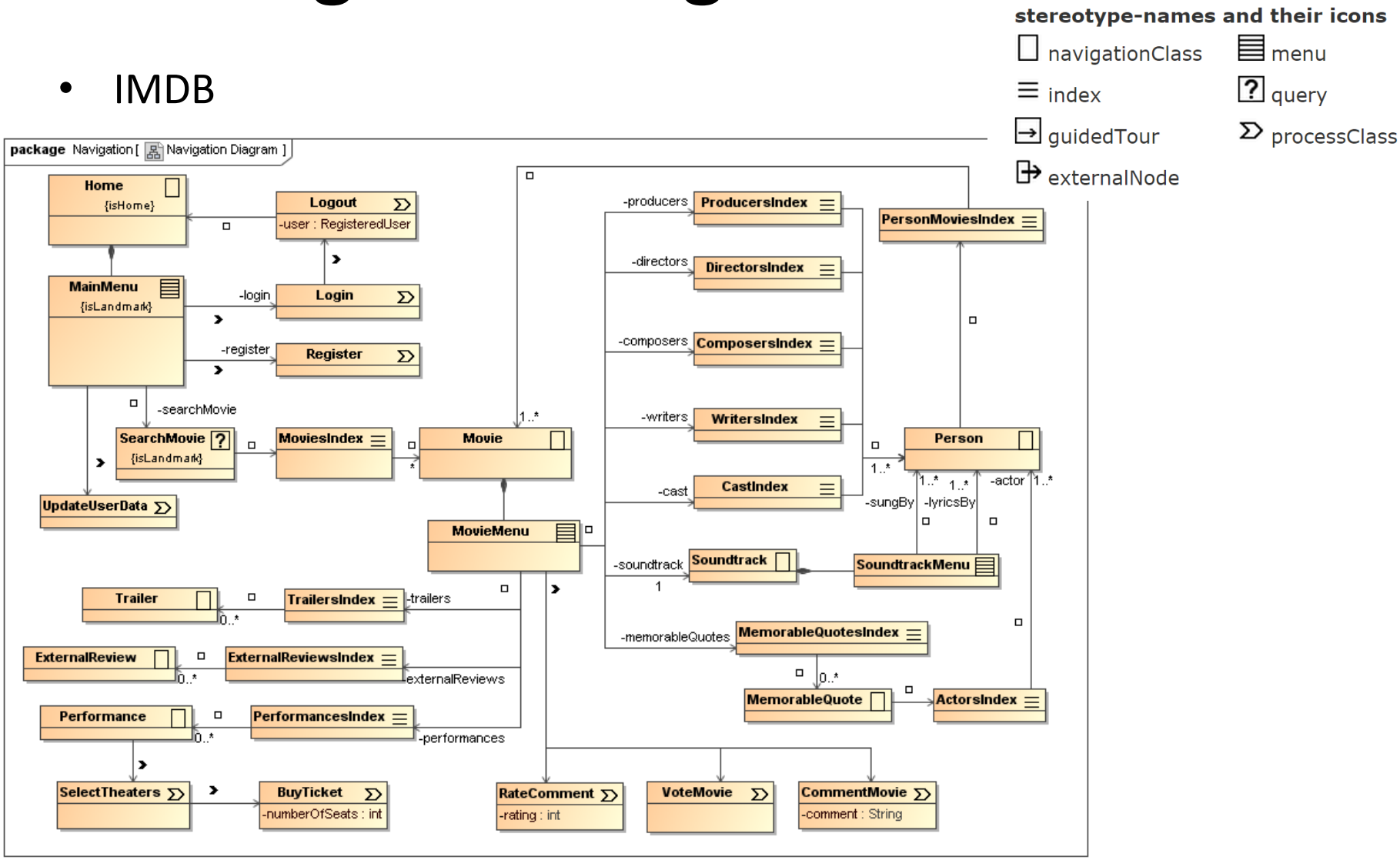
Source: Web Engineering – Kappel et al.

Hypertext Modeling

- *Purpose*: To model the navigation paths available to users.
  - How pages link to other pages
  - In the form of a Navigation Diagram
    - NavigationClass – A node (class or page)
    - Index – A list of something
    - GuidedTour – Like a wizard (keep pressing next)
    - ExternalNode – A node hosted on a different domain
    - Menu – A bunch of links
    - Query – Search functionality
    - ProcessClass – Does some processing but doesn't directly output to the user

Navigation Diagram – Example

- IMDB



http://uwe.pst.ifi.lmu.de/exampleIMDB.html

Presentation Modelling

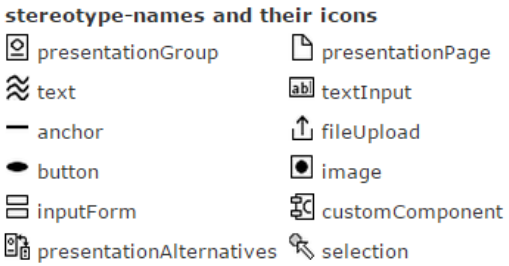
- *Purpose*: To model the look & feel of the Web application at the page level.
- The design should aim for *simplicity* and *self-explanation*.
- Describes presentation structure:
  - Composition & design of each page
  - Identify recurring elements (headers/footers)
- Describes presentation behavior:
  - Elements => Events

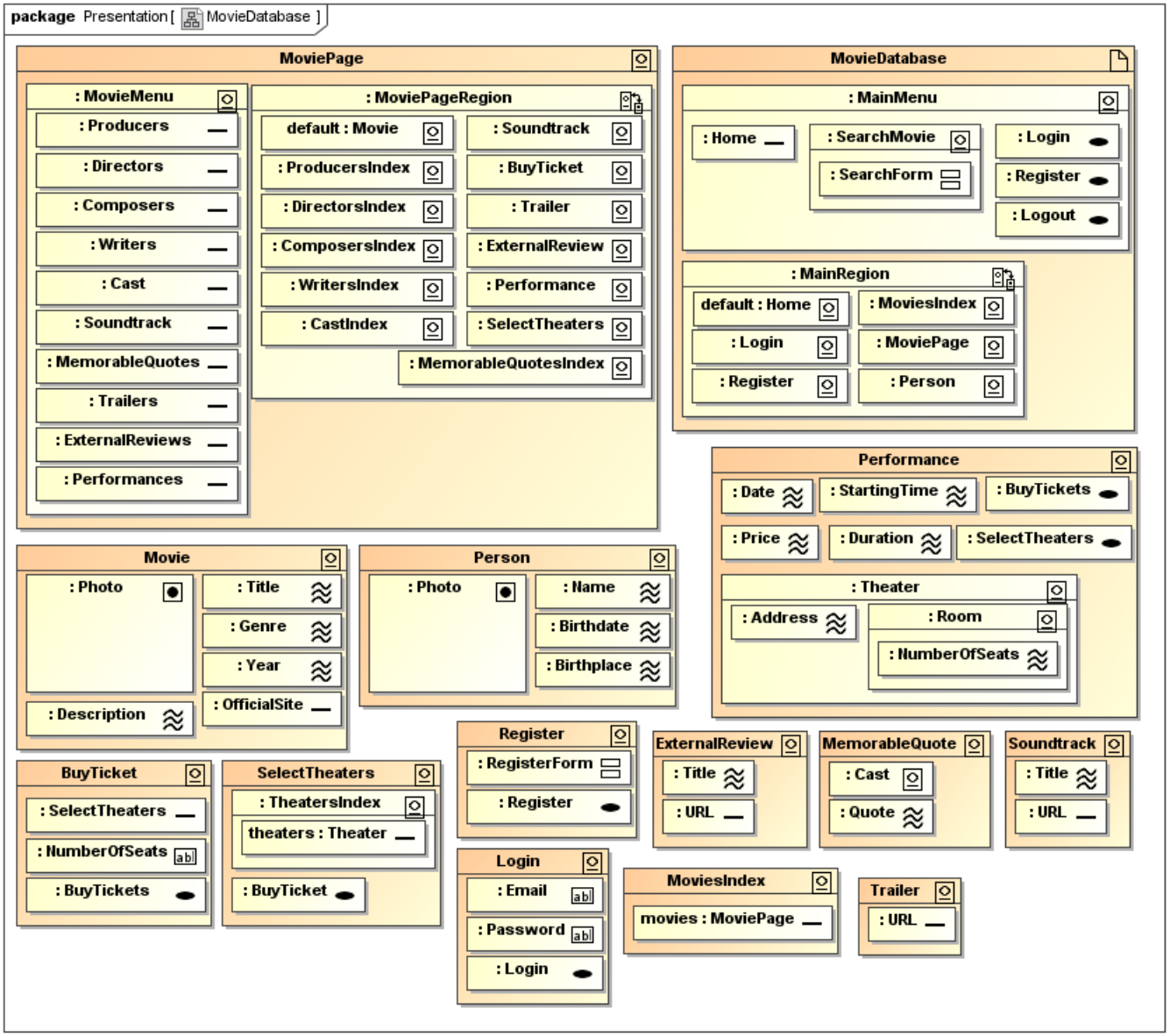
Levels of Presentation Models

- *Presentation Page* – “root” element; equivalent to a page container.
- *Presentation Unit*
  - A fragment of the page logically defined by grouping related elements.
  - Represents a hypertext model node
- *Presentation Element*
  - A unit’s (node’s) informational components
  - Text, images, buttons, fields

Presentation Model – Example

- IMDB

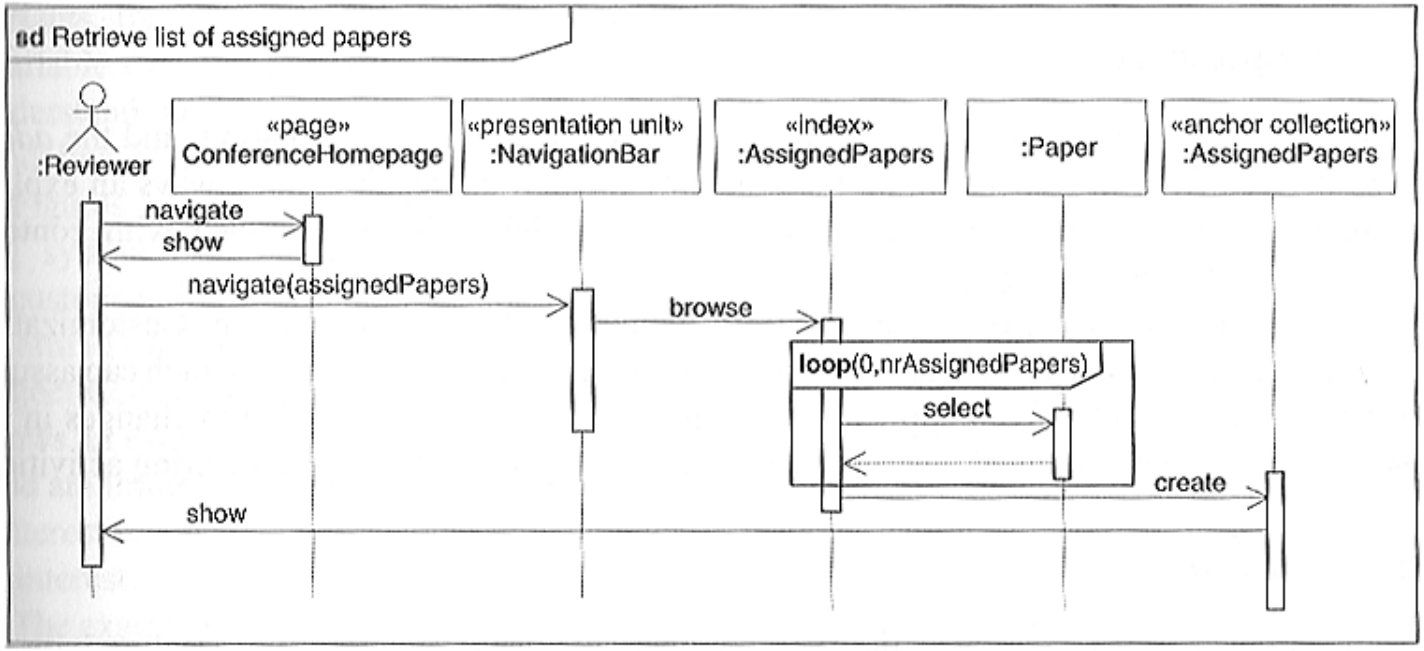




Sequence Diagrams

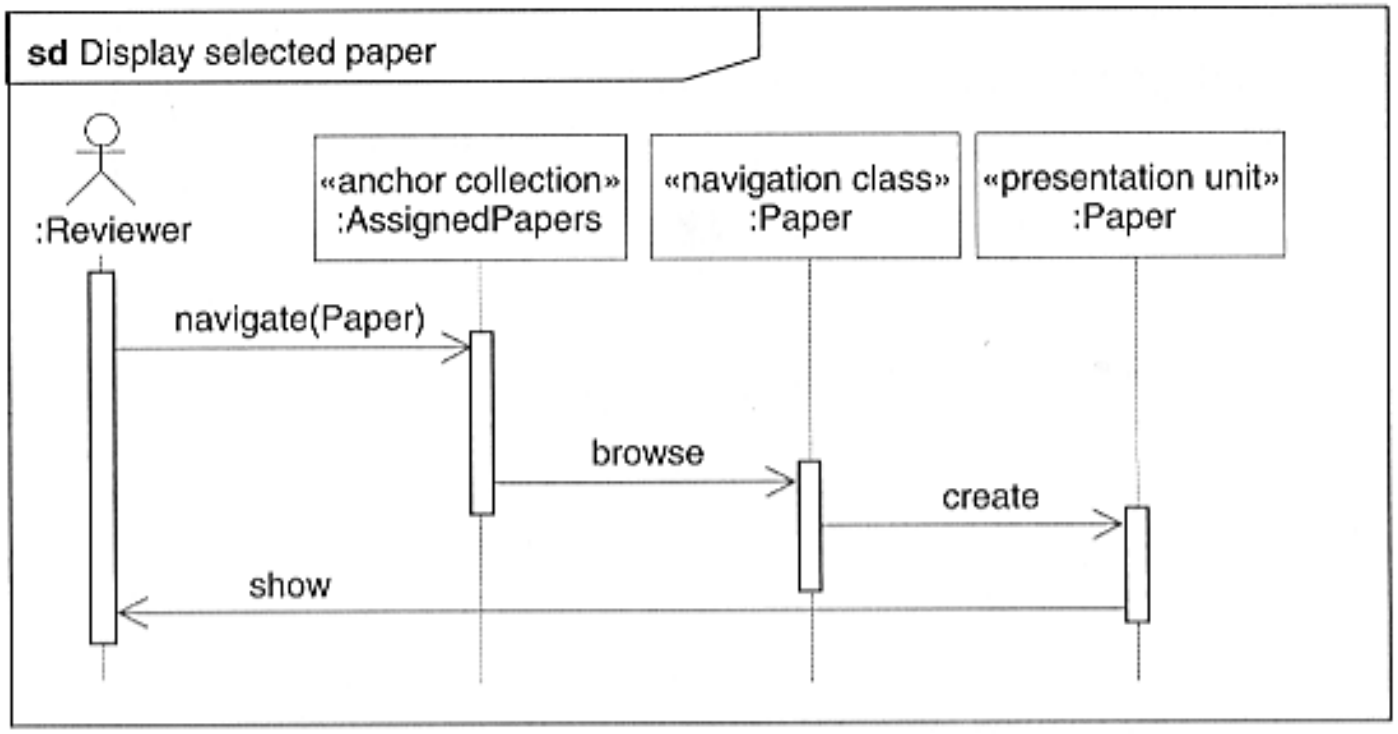
- *Purpose*: Depicts sequential interactions (i.e., the flow of logic) between objects in an application over time.
  - What messages, what order, & to whom.
  - Ex.: Object A calls method of Object B
  - Ex.: Object B passes method call from Object A to Object C.
- *Result:* Dynamic system interactions diagrammed in a “fence” format.

Sequence Diagram – Example 1



Source: Web Engineering – Kappel et al.

Sequence Diagram – Example 2



Source: Web Engineering – Kappel et al.

Customization Modeling

- Ubiquitous applications often require *multi-platform* and/or *personalized* delivery.
- *Purpose*: To represent content adaptation based on context information.
- Customization models are often integrated with other diagrams.
- Use <<customization>> annotation to denote customization rules in adapted classes.
  - Use customComponent and presentationAlternatives icons in presentation model

Customization Modeling Concepts

- *User Profile* – preferences & characteristics
- *Physical context* – user or location
- *Logical context* – situational knowledge
- *Context adaptation*
  - *Static* – A unique model for every variant
  - *Dynamic* – Rule-based transformation
    - Events/Conditions/Actions
    - Actual variants created at runtime, so model may be harder to understand.

More Design Issues

- Other design issues for Web Engineering
  - Performance
  - Reliability
  - Robustness
  - Scalability
  - Security
  - Maintainability

A Word on Iteration

- There is no linear order through the processes of refining Interaction Diagrams and Class Diagrams, identifying responsibilities and associations, weakening coupling and defining navigability
- Everything feeds back into everything else
- And it all feeds back into the requirements specification!

A Word on Iteration

Suggestions

- Small steps – do one interaction diagram, then update the class diagram, then reduce coupling, and repeat
- Do the use-cases for the areas of highest risk first – any problems can be fixed before they affect the rest of the design

Some Suggestions

- It is tempting to implement straight from the first-draft design

DON'T!

Some Suggestions

- Before implementing, consider the following
  - Does the design retain/encapsulate the responsibilities of the use-cases?
  - Is the system robust to change, and does it protect the user (actors) from such change?
  - Does it have good locality of modification?
  - E.g. what if it was decided to encrypt the balance – how much code would need changing?

THE END

QUESTIONS??

THANKS!!