INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 6: Web Engineering Fundamentals

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the role of web engineering
  - Learn a systematic process for web applications development
- Outline
  - Introduction
  - Requirements Analysis
  - Web Modeling
  - Web Design and Architectures
  - Web Accessibility

Resources

- Books
  - Roger S. Pressman, David Lowe (2009). *Web Engineering: A Practitioner's Approach*, McGraw-Hill. http://highered.mcgraw-hill.com/sites/0073523291/
  - Roger Pressman (2005). *Software Engineering: A Practitioner's Approach*, 6/e, McGraw-Hill Higher Education. Chapters 16-20. http://highered.mcgraw-hill.com/sites/0072853182/information_center_view0/
  - G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger (eds), *Web Engineering - The Discipline of Systematic Development of Web Applications*, John Wiley & Sons, 2006. http://www.web-engineering.at/eng/
- Online material
  - INFSCI 2955: Web Engineering
  - Department of Information Science and Telecommunications, University of Pittsburgh http://www.sis.pitt.edu/~jgrady/

6.3 MODELING WEB APPLICATIONS

Why Create Models?

- Define an abstract view of a real-world entity
  - Finding & discovering objects/concepts in a domain
  - Assigning responsibilities to objects
- Tool of thought
  - Reduce complexity
  - Document design decisions
- Means of communication

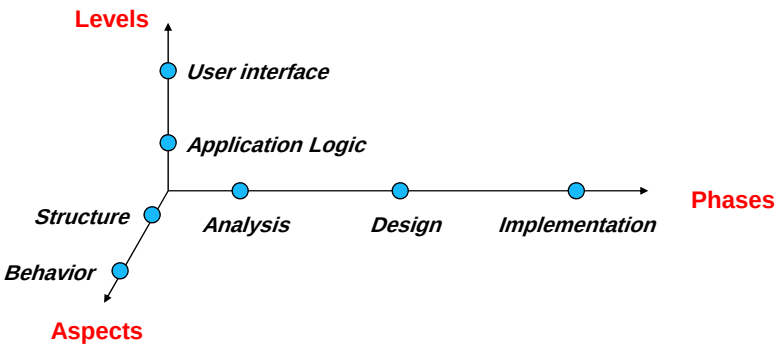
Web Modeling

- Modeling static & dynamic aspects of content, hypertext, and presentation.
- We focus on object-oriented analysis & design
- There are two types of modeling
  - Analysis model
    - establishes a basis for design
    - finds & discovers objects/concepts in a domain
  - Design model
    - represents key WebApp elements
    - defines software objects & how they interact to fulfill requirements
- Key skill: Assigning responsibilities to objects

Assigning Responsibilities

- Responsibilities are obligations or specific behaviors related to its role.
- What does an object do?
  - Doing something itself
  - Pass actions (messages) to other objects
  - Controlling & coordinating the activities in other objects
- What does an object know?
  - Private, encapsulated data
  - Its related objects
  - Items it can derive or calculate

Software Application Modeling

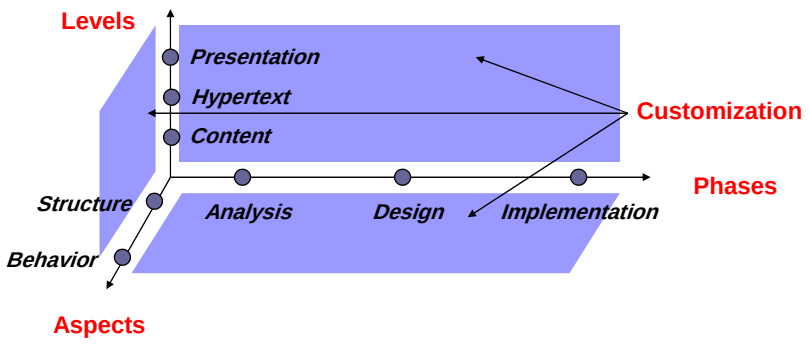


- Levels – the "how" & "what" of an application
- Aspects – objects, attributes, and relationships; function & processes
- Phases – development cycle

Unified Modeling Language (UML)

- "The Unified Modeling 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.
  - Functional requirements view: emphasizes the functional requirements of the system from the user's point of view.
    - includes use case diagrams
  - Static structural view: emphasizes the static structure of the system using objects, attributes, operations and relationships.
    - includes class diagrams and composite structure diagrams.
  - Dynamic behavior view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.
    - includes sequence diagrams, activity diagrams and state machine diagrams.
- It can also be used for code generation from models and model generation from code (round-trip engineering)

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

Web Application Modeling (cont.)

- For Web-centric modeling, we will employ the UML Web Engineering (UWE) notation.
  - http://www.pst.ifi.lmu.de/projekte/uwe/
  - Relies on Object Management Group (OMG) standards – (i.e., UML-compliant)
    - UWE's notation is defined as a "lightweight" extension of the Unified Modeling Language (UML) providing a so called UML Profile for the Web domain.
  - Comprehensive modeling tool
  - Supports semi-automatic generation of code

Requirements Modeling

- Serves as a bridge between Requirements & Design phases
- Uses cases – functional requirements written as a collection of related success & failure scenarios.
  - *Scenario* – a sequence of actions & interactions between actors and a system.
- Preferred means of modeling requirements
  - Written descriptions are easy to understand
  - Emphasize the users goals and perspective

Use Cases

- Defining valid use cases:
  - *The Boss Test* – measurable value
  - *The EBP Test* – one person, one place, one time
  - *The Size Test* – more than one step
- Which is a valid use case?
  - Negotiate a Supplier Contract
  - Handle Returns
  - Log In
  - Move Piece on Game Board

Use Cases (cont.)

- Critical components
  - *Use Case Name* – starts with a verb
  - *Level* – "user-goal" or "subfunction"
  - *Primary Actor* – the user whose goal is fulfilled
  - *Stakeholders & Interests* – Who cares, and what do they want?
  - *Preconditions* – What must be true at the start
  - *Success Guarantee* – defines the successful completion of the use case for <u>all</u> stakeholders

Use Case – Example 1

- Use Case 1: Create User
- Scope: University or business network
- Level: user goal
- Primary Actor: user (system administrator)
- Stakeholders and Interests:
  - System Administrator: Wants control over users' access to system resources.
  - New User: Wants access to system resources for communication, business, and research.
  - Organization: Wants security and controlled access of organization resources, data, intellectual property; wants employees/students to have appropriate system access to fulfill the goals of the organization.
- Preconditions: User is identified, authenticated, and has opened administration tool
- Success Guarantee: New user account is created and saved. Username and password grant the new user access to network.

Use Case – Example 1 [cont.]

- **Main Success Scenario**:
  1. System requests input for username & password
  2. User enters username & password
  3. System requests other identifiable user information (ex. real name, SSN#, address)
  4. User enters other identifiable user information
  5. System verifies username & password
  6. System stores new user information
  7. System displays success message
  8. System presents user options

Use Case Guidelines

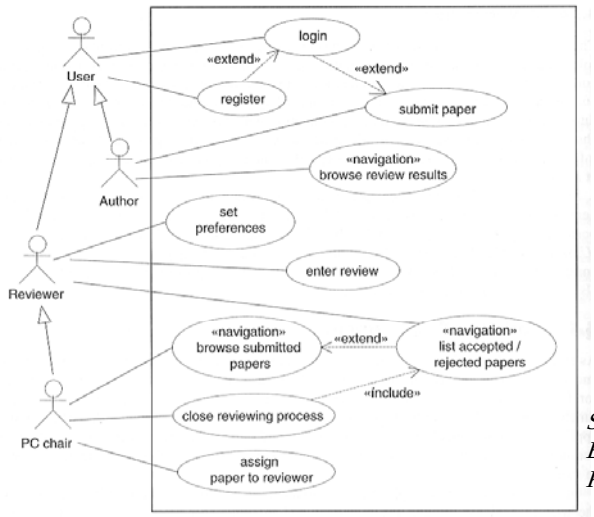
- Use short sentences
- Delete "noise" words
  - NO :"*The* System authenticates…"
  - YES: "System authenticates…"
- Avoid technology-specific terms (initially, at least)
  - NO : "Cashier swipes Product ID across scanner."
  - YES: "Cashier enters Product ID."

Use Case Diagrams

- Provide a graphical overview of a system's use cases, its external actors, and their relationships
- Use case diagrams are NOT requirements!
- Can be used for functional & hypertext requirements
  - Same model (UWE/authors' approach)
  - Use "<<navigation>>" annotation to distinguish hypertext from functional

Use Case Diagram - Example

- Conference Paper Submission System



Source: Web Engineering – Kappel et al.

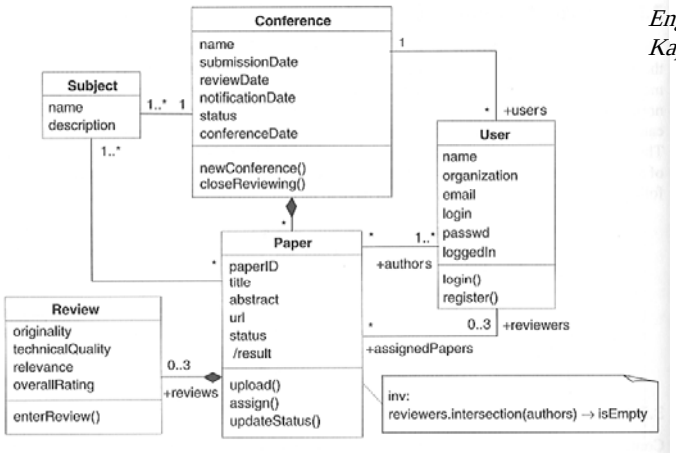
Content Modeling

- *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 Diagram – Example 1

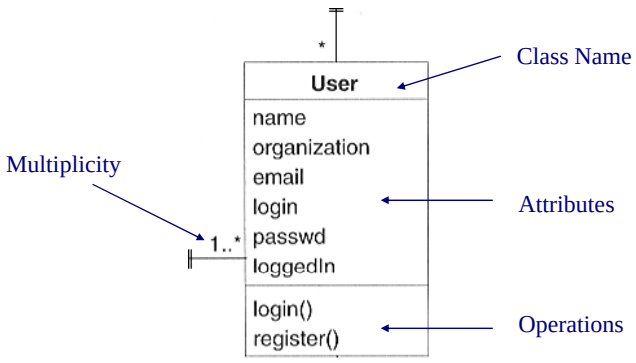
- Conference Paper Submission System

Source: Web Engineering – Kappel et al.



Class Diagrams (cont.)

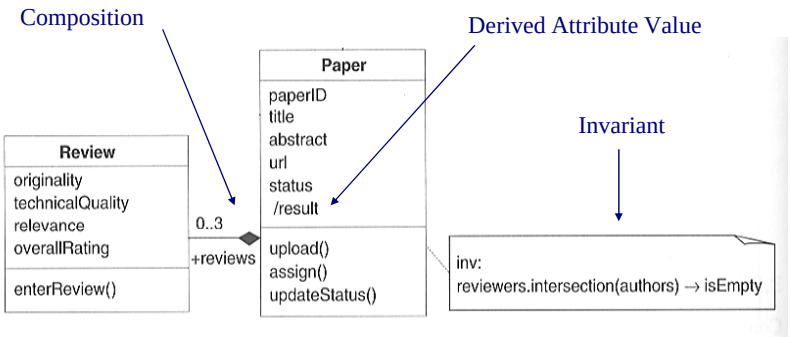
- Notations



Source: Web Engineering – Kappel et al.

Class Diagrams (cont.)

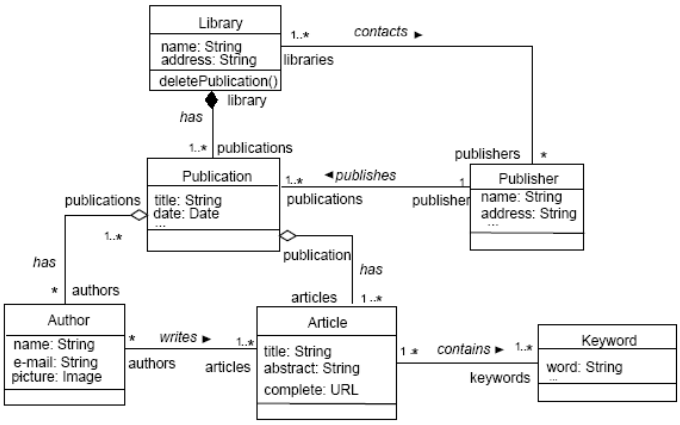
- Notations (continued)



Source: Web Engineering – Kappel et al.

Class Diagram – Example 2

- Online Library Application

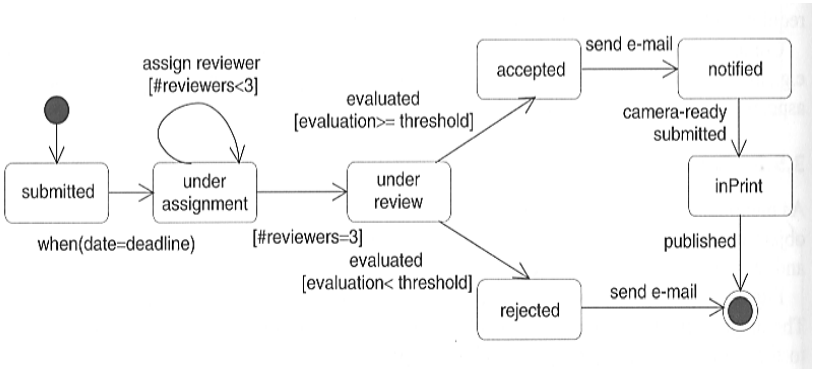


Source: Web Engineering – Kappel et al.

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
- For pure UML modeling, can be very useful for hypertext models (next section).

State Machine Diagram - Example



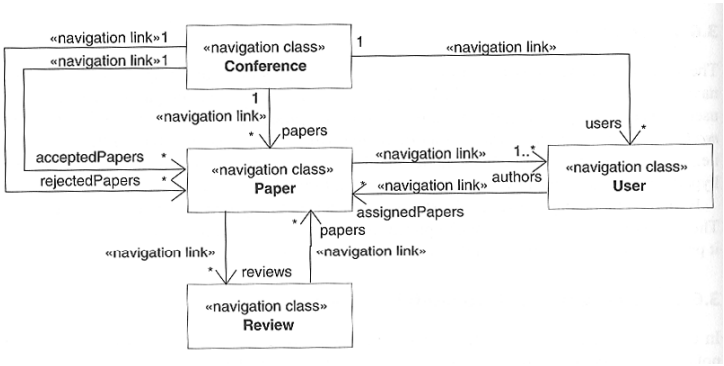
Source: Web Engineering – Kappel et al.

Hypertext Modeling

- Purpose: To model the navigation paths available to users.
- Artifacts
  - Hypertext Structure Model – navigating among classes
  - Access Model – UML-compliant site map
- Focuses on the structure of the hypertext & access elements.
- Use "<<navigation class>>" annotation to distinguish from content classes.

Hypertext Structure Model

- Conference Paper Submission System



Source: Web Engineering – Kappel et al.

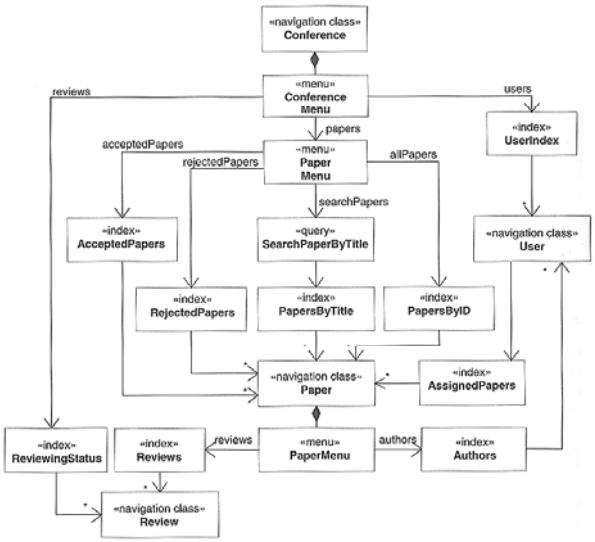
Link Classification Types

- UWE
  - Navigation vs. Process vs. External
- HDM
  - Structural vs. Perspective vs. Application
- WebML
  - Contextual vs. Non-contextual
  - Intra-page vs. Inter-page
- OO-H
  - I, T, R, X, S-links

Access Model

- Hypertext structure models describe navigation, but not orientation.
- Access models describe both through Navigation patterns, used to consistently describe conventional elements.
  - <<index>> (list)
  - <<guided-tour>> (sequential links)
  - <<menu>>, <<query>>

Access Model - Example



Source: Web Engineering – Kappel et al.

Presentation Modeling

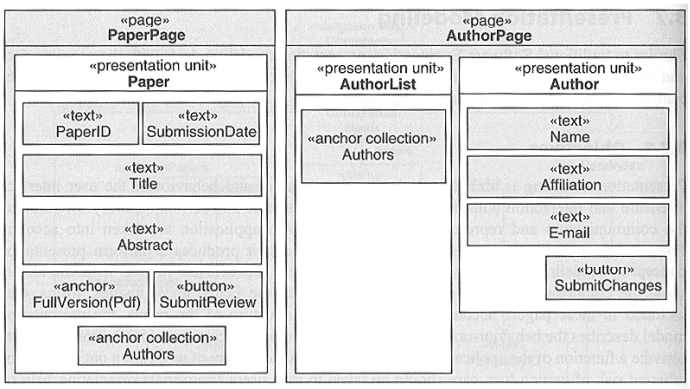
- Purpose: To model the look & feel of the Web application at the page level.
- The design should aim for simplicity and self-explanation.
- Describes presentation structure:
  - Composition & design of each page
  - Identify recurring elements (headers/footers)
- Describes presentation behavior:
  -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

Composition Model - Example

- Paper and Author Page Templates

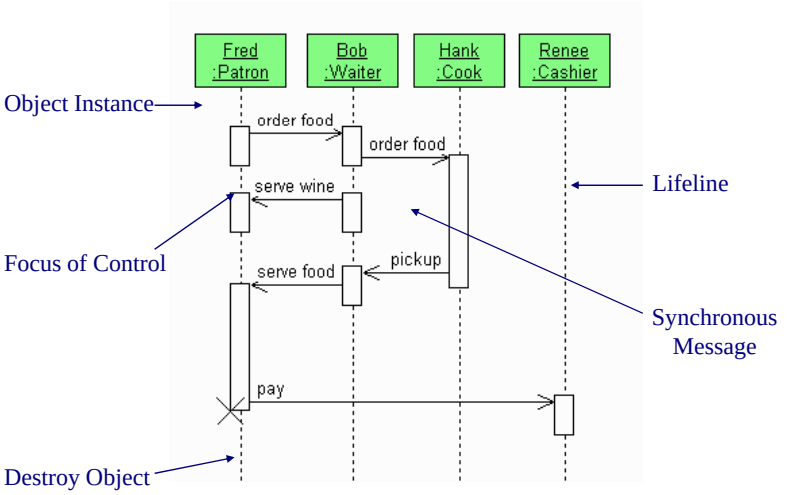


Source: Web Engineering – Kappel et al.

Sequence Diagrams

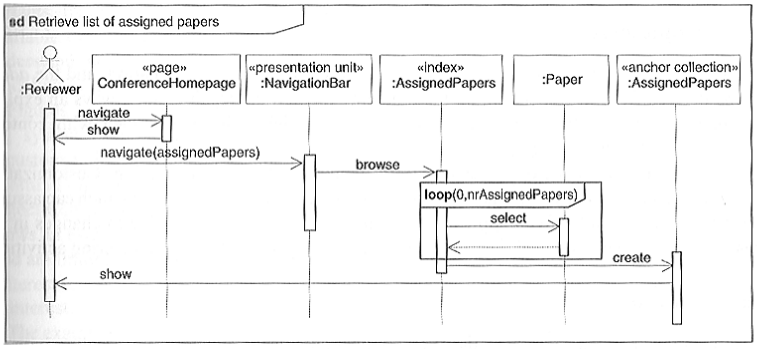
- Purpose: Depicts sequential interactions (i.e., the flow of logic) between objects in an application over time.
  - What messages, what order, & to whom.
  - Ex.: Object A calls method of Object B
  - Ex.: Object B passes method call from Object A to Object C.
- Result: Dynamic system interactions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

Sequence Diagram - Notation



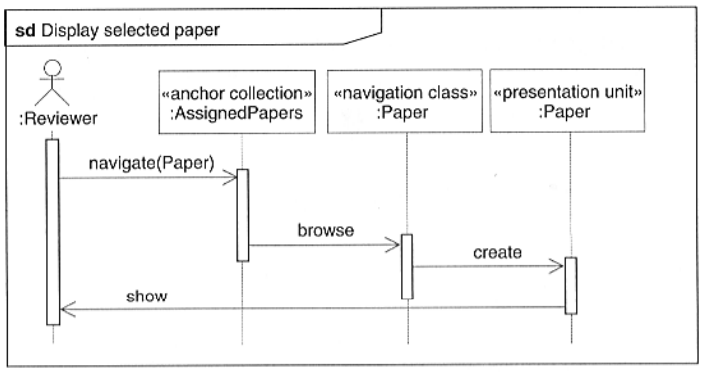
Source: Wikipedia – Sequence Diagram

Sequence Diagram – Example 1



Source: Web Engineering – Kappel et al.

Sequence Diagram – Example 2



Source: Web Engineering – Kappel et al.

Modeling Methods

- We've primarily discussed Object-Oriented Modeling (e.g., UML), but there are other methodologies:
  - Data-Oriented (Hera, WebML)
  - Hypertext-Oriented (HDM)
  - Software-Oriented (WAE)
- Choosing a method depends on system purpose, focus, and requirements

Q & A

?

Tools

- Unified Modeling Language (UML)
  - http://en.wikipedia.org/wiki/Unified_Modeling_Language
- List of UML Tools
  - http://en.wikipedia.org/wiki/List_of_UML_tools
- Use Cases Tutorial
  - http://www.parlezuml.com/tutorials/usecases.htm
- UWE – UML-based Web Engineering
  - http://www.pst.ifi.lmu.de/projekte/uwe/