2. Web Engineering Fundamentals

1. Introduction
2. Requirements Analysis
3. Web Modeling
4. Web Architectures
5. Web Accessibility

Resources

- Book
  - Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). **Web Engineering, 1st ed.** Hoboken, NJ: Wiley & Sons. ISBN: 04700-1554-3.
- Online material
  - INFSCI 2955: Web Engineering
  - Department of Information Science and Telecommunications, University of Pittsburgh
  - Website: http://www.sis.pitt.edu/~jgrady/

2.1 Introduction to Web Engineering

What is Web Engineering?

- "The application of systematic and quantifiable approaches to cost-effective analysis, design, implementation, testing, operation, and maintenance of high-quality Web applications." – Kappel *et al.*

- Extends *Software Engineering* to Web applications, but with Web-centric approaches.

Defining Web Applications

- A *Web application* is a system that utilizes W3C standards & technologies to deliver Web-specific resources to clients (typically) through a browser.
  - A strict definition that ensures software and UI aspects of the Web are examined carefully
- Technology + interaction.
  - Web site with no software components?
  - Web services?

The Case for Web Engineering

- Application development on the Web remains largely *ad hoc*.
  - Spontaneous, one-time events
  - Individual experience
  - Little or no documentation for code/design
- Short-term savings lead to long-term problems in operation, maintenance, usability, etc.
- Because Web apps are so interdependent, the problem is compounded.

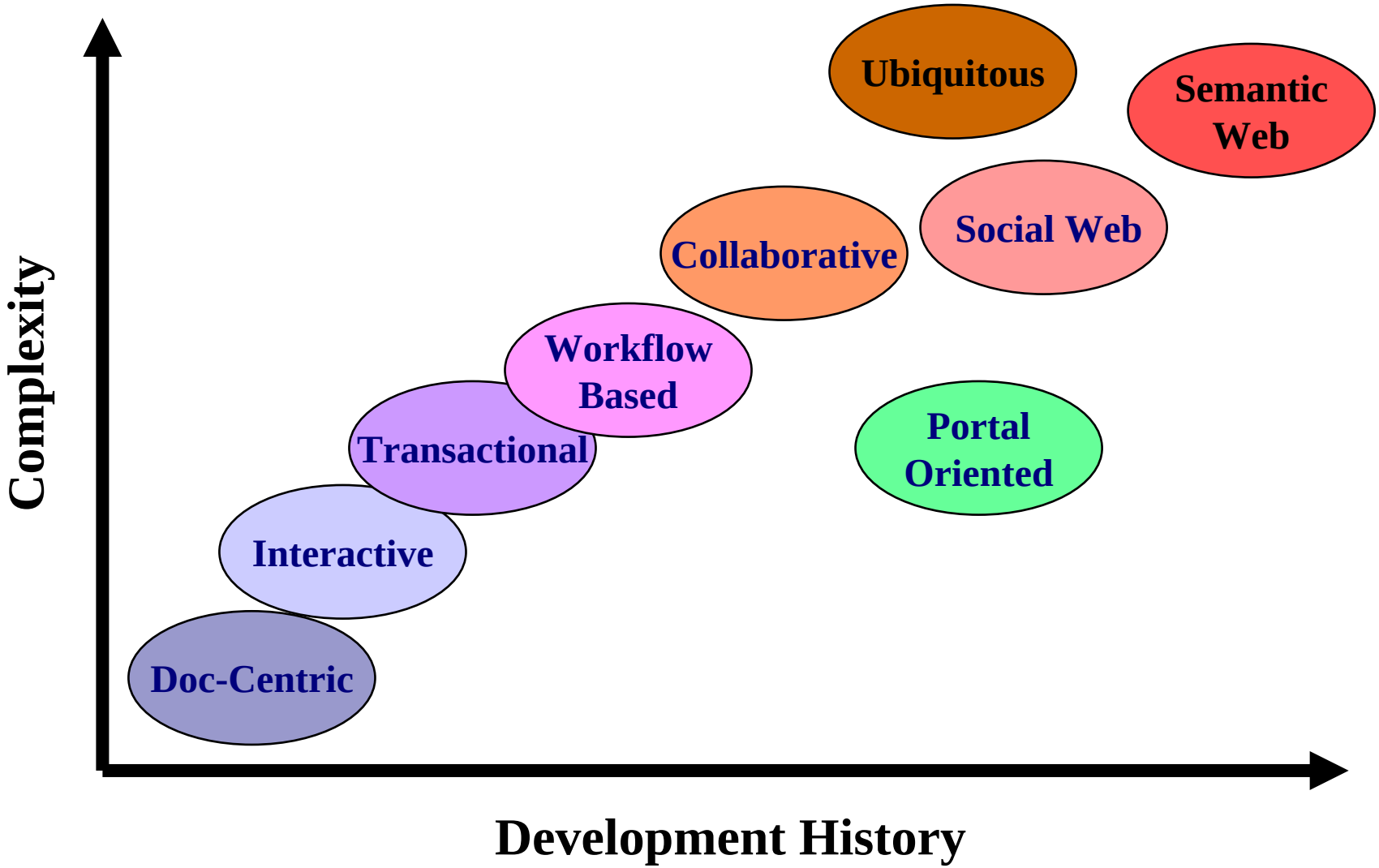
The Case for Web Engineering (cont.)

- Root Causes of poor design
  - Development as an authoring activity
  - Development is “easy”
  - Techniques that *should not* be used are misapplied.
  - Techniques that *should* be used are *not*.
- Particularly alarming given…
  - Most projects are now Web-based
  - More “mission-critical” apps moving to the Web

The Case for Web Engineering (cont.)

- Top project pitfalls (Cutter, 2000)
  - 84% - Failure to meet business objectives
  - 79% - Project schedule delays
  - 63% - Budget overrun
  - 53% - Lack of functionality
- Web Engineering's solution:
  - Clearly defined goals & objectives
  - Systematic, phased development
  - Careful planning
  - Iterative & continuous auditing of the entire process

Categories of Web Applications



Document-Centric Web sites

- Precursors to Web applications
- Static HTML documents
- Manual updates
- Pros
  - Simple, stable, short response times
- Cons
  - High management costs for frequent updates & large collections
  - More prone to inconsistent/redundant info
- Example: static home pages

Interactive & Transactional

- Come with the introduction of CGI and HTML forms
- Simple interactivity
- Dynamic page creation
  - Web pages and links to other pages generated dynamically based on user input
- Content updates -> Transactions
  - Decentralized
  - Database connectivity
  - Increased complexity
- Examples: news sites, booking systems, online banking

Workflow-Based Applications

- Designed to handle business processes across departments, organizations & enterprises
- Business logic defines the structure
- The role of Web services
  - Interoperability
  - Loosely-coupled
  - Standards-based
- Examples: B2B & e-Government
- High complexity; autonomous entities

Collaborative & Social Web

- Unstructured, cooperative environments
  - Support shared information workspaces to create, edit and manage shared information
- Interpersonal communication is paramount
- Classic example: Wikis
- The Social Web
  - Anonymity traditionally characterized WWW
  - Moving towards *communities of interest*
  - *Examples:* Blogs, collaborative filtering systems, social bookmarking (e.g., del.icio.us)
  - Integration with other forms of web applications (e..g, NetFlix)

Portal-Oriented

- Single points-of-entry to heterogenous information
  - Yahoo!, AOL.com, portal.kfupm.edu.sa
- Specialized portals
  - Business portals (e.g., employee intranet)
  - Marketplace portals (horizontal & vertical)
  - Community portals (targeted groups)

Ubiquitous

- Customized services delivered anywhere via multiple devices
- HCI is critical
  - Limitations of devices (screen size, bandwidth?)
  - Context of use
- Still an emerging field; most devices have single focus:
  - Personalization
  - Location-aware
  - Multi-platform delivery

Semantic Web

- Berners-Lee: Information on the Web should be readable to machines, as well as humans.
- Using metadata and ontologies to facilitate knowledge management across the WWW.
- Content syndication (RSS, Atom) promotes re-use of knowledge
- Is the Semantic Web even possible?

Characteristics of Web Apps

- How do Web applications differ from traditional applications?
- Or, another way, what Software Engineering methods & techniques can be adapted to Web Engineering?
- 3 dimensions of the ISO/IEC 9126-1 standard
  - Product
  - Usage
  - Development

Characteristics - Product

- Product-related characteristics constitute the “building blocks” of a Web application
- Content
  - Document character & multimedia (# of dimensions?)
  - Quality demands: current, exact, consistent, reliable
- Navigation Structure (Hypertext)
  - Non-linearity
  - Potential problems: Disorientation & cognitive overload
- User interface (Presentation)
  - Aesthetics
  - Self-explanation

Characteristics - Usage

- Much greater diversity compared to traditional non-Web applications
  - Users vary in numbers, cultural background, devices, h/w, s/w, location etc
- Social Context (Users)
  - Spontaneity - scalability
  - Heterogeneous groups
- Technical Context (Network & Devices)
  - Quality-of-Service
  - Multi-platform delivery
- Natural Context (Place & Time)
  - Globality
  - Availability

Characteristics - Development

- The Development Team
  - Multidisciplinary – print publishing, s/w devt, marketing & computing, art & technology
  - Community (including Open Source)
- Technical Infrastructure
  - Lack of control on the client side
  - Immaturity
- Process
  - Flexibility
  - Parallelism
- Integration
  - Internal – with existing legacy systems
  - External – with Web services
  - Integration issues: correct interaction, guaranteed QoS

2.2 Requirements Engineering

Overview

- Introduction to Requirements Engineering
- Fundamentals
- Specifics in Web Engineering
- Principles
- Adapting traditional Requirements Engineering to Web applications

Introduction

- *Requirements Engineering (RE)* – the principles, methods, & tools for eliciting, describing, validating, and managing project goals and needs.
- Given the complexity of Web apps, RE is a critical initial stage, but often poorly executed.
- What are the consequences?
  - Inadequate software architectures
  - "Unforeseen" problems
    - Budget overruns
    - Production delays
    - "That's not what I asked for"
  - Low user acceptance

Why Define Requirements?

- The authors build their case:
  - Bell & Thayer (1976) – Requirements don't define themselves.
  - Boehm (1981) – Removal of mistakes post hoc is up to *200 times* more costly.
  - The Standish Group (1994) – 30% of project fail before completion & almost half do not meet customer requirements
    - Unclear objectives, unrealistic schedules & expectations, poor user participation

Fundamentals of RE - 1

- Identify and involve (if possible) 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.
- Already, one can see RE as more of an art than a science.

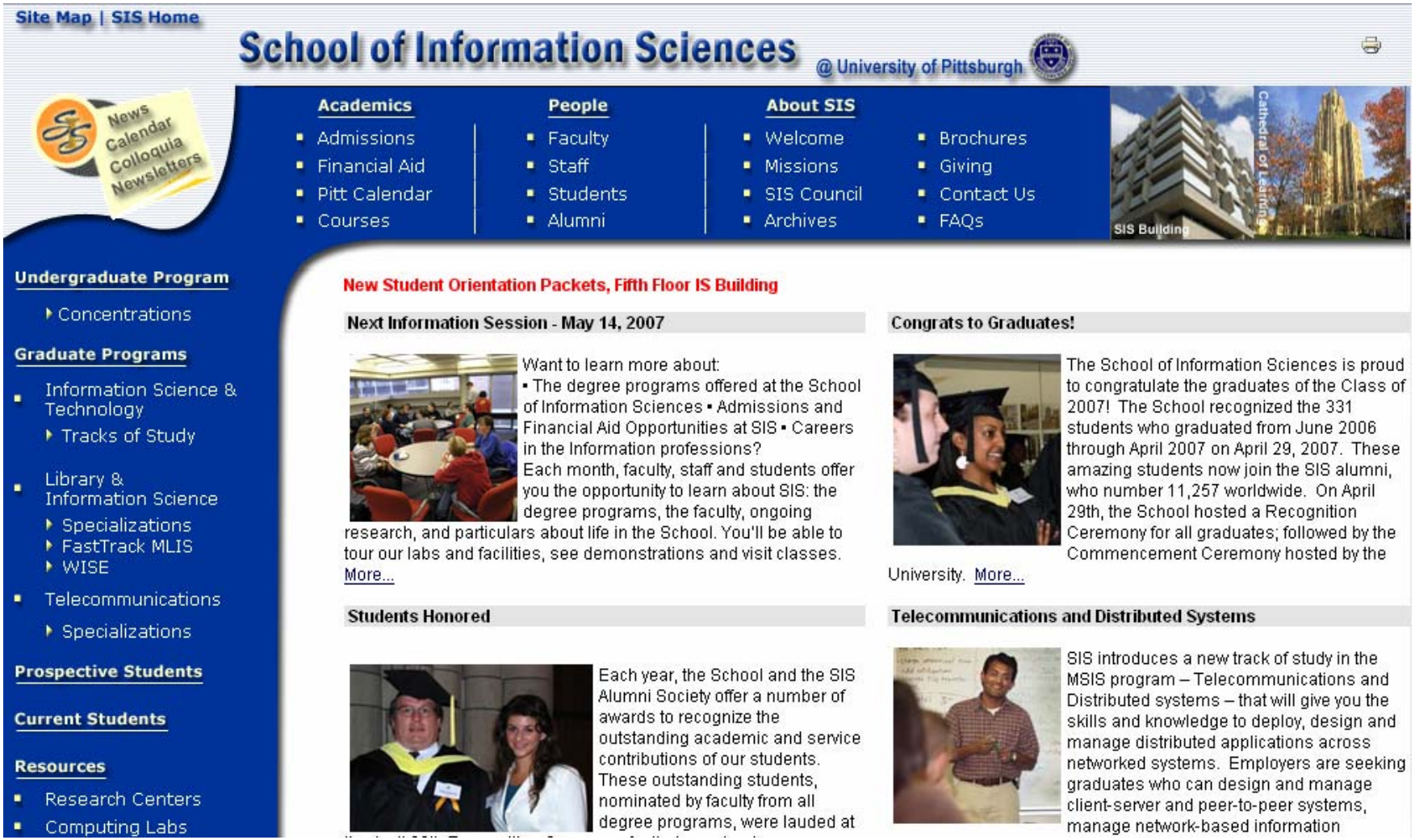
Fundamentals of RE - 2

- 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
- Keys to requirement definition:
  - Negotiation
  - Scenario-based discovery
  - Clear definition of context and constraints

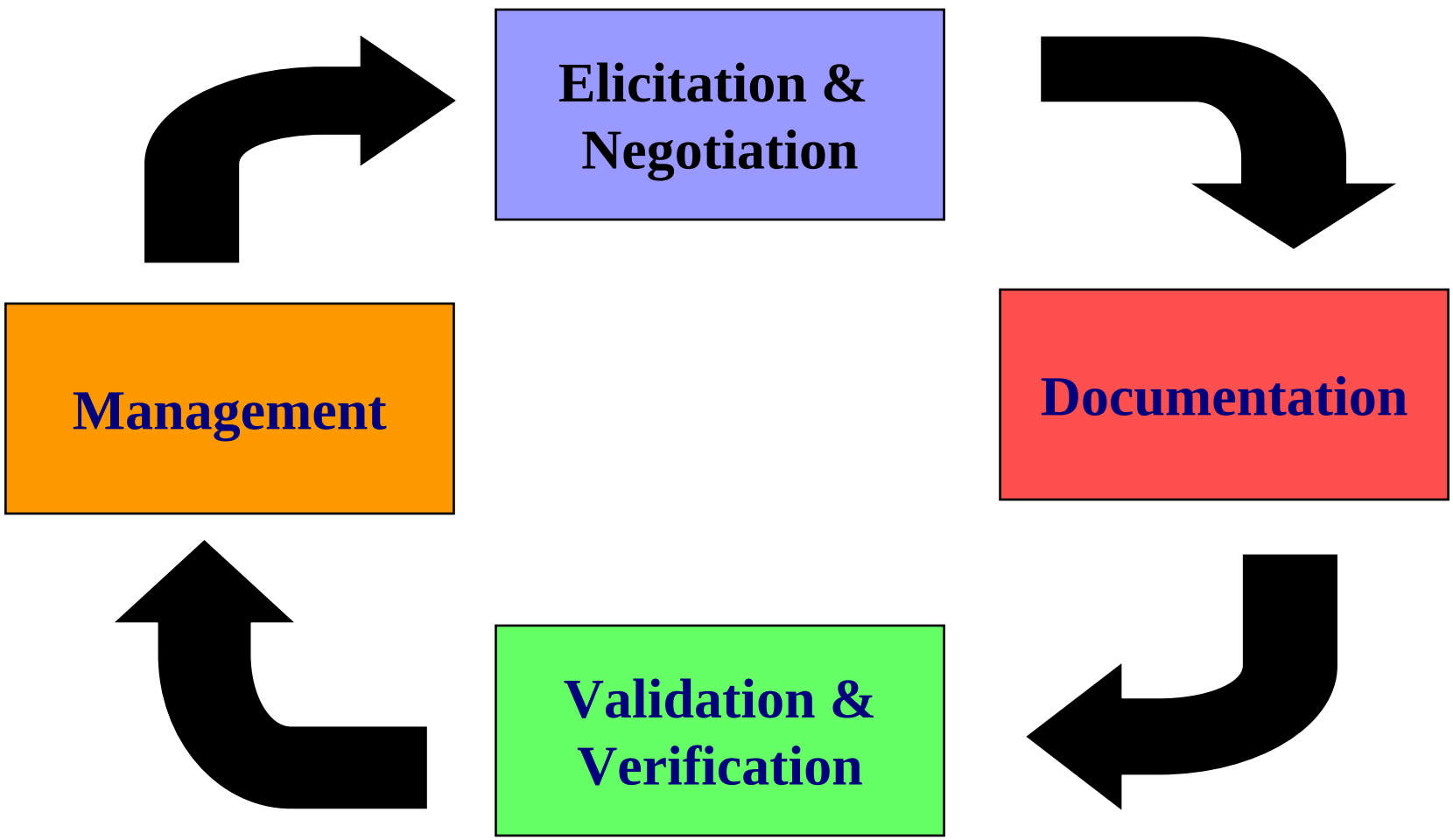
Fundamentals of RE - 3

- Objectives, objectives, objectives
  - Advertising
  - Customer service
  - Business transactions
- Audience, audience, audience
  - The designer is not the audience
  - Audience segmentation
  - User interviews and testing
- What about the Competition?
  - Other web sites
  - Other forms of advertising and transactions

Example: SIS Website



Summary - RE Activities



Specifics in Web Engineering

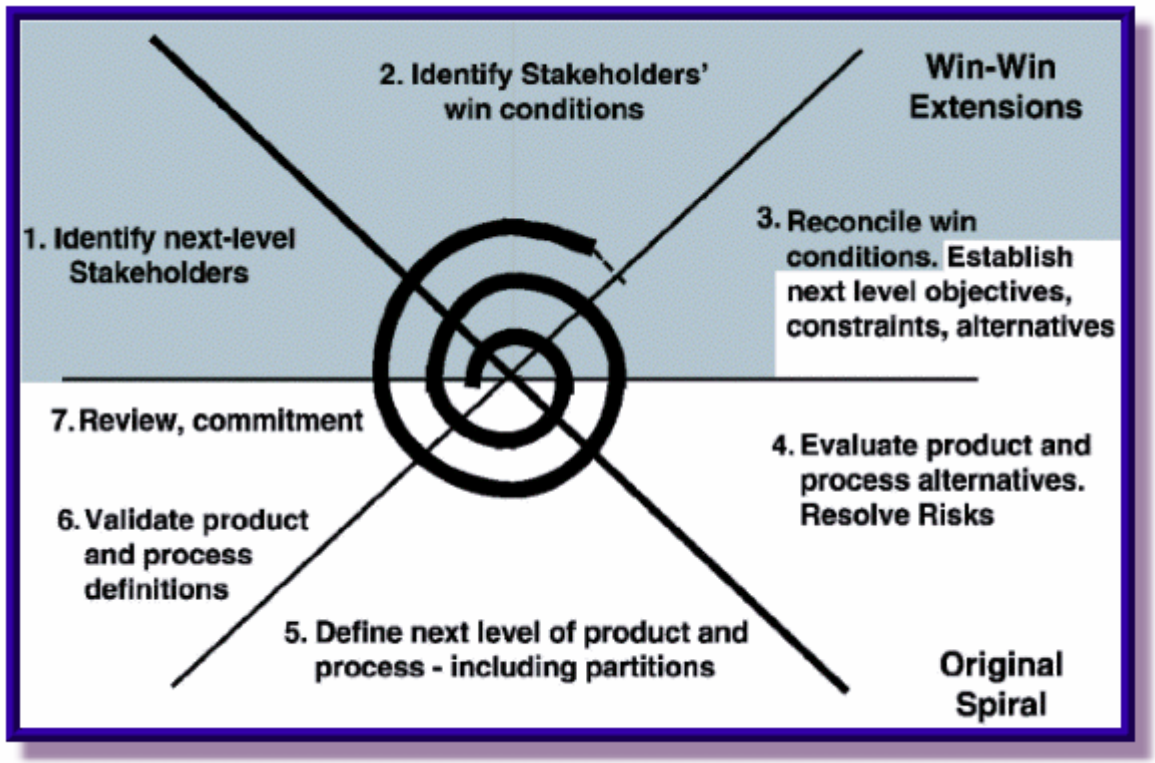
- Is RE for the Web really that different than RE for conventional software?
- Some would argue “no”, but many aspects of Web applications suggest otherwise
- 10 distinguishing characteristics
  - Multidisciplinary
  - Unavailability of stakeholders
  - Rapidly changing requirements & constraints

Specifics in Web Engineering - 2

- 10 distinguishing characteristics (cont.)
  - Unpredictable operational environment
  - Integration of legacy systems
    - Constrained by existing system
    - Constrained by $$$
  - Quality aspects
  - User interface quality
  - Content quality
  - Developer inexperience
  - Firm delivery dates

Principles for RE

- Inspired by the win-win spiral model (Boehm, 1996)



Source: http://www.stsc.hill.af.mil/Crosstalk/2001/12/boehm3.gif

Principles for RE - 2

- 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.

Principles for RE - 3

- 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
  - As the project progresses, requirements can become more concrete.

Principles for RE - 4

- Focusing on the System Architecture
  - The “solution space” – existing technologies & legacy systems – defines the “problem space.”
  - The architecture *must* be considered in the elicitation stage.
  - Refine requirements and architecture iteratively with increasing level of detail.

Principles for RE - 5

- Risk Orientation
  - Risk management is at the heart of the analysis process.
  - What are the greatest risks?
    - Integration issues w/ legacy systems
    - Expected vs. actual system quality
    - Inexperience of developers
  - How to mitigate risks?
    - Prototyping (avoid IKIWISI)
    - Show changes to customer iteratively
    - Integrate existing systems sooner than later

Adapting RE to Web Applications

- There isn't one single "right way" to RE among the many methods, techniques, tools, etc. available.

- For your Web application project, ask the following questions:
  - What are the critical requirements?
  - How should requirements be documented?
  - What tools should be use, if any?

Adapting – Requirement Types

- Taxonomies (e.g. IEEE 830) exist that describe *functional* and *non-functional* requirements.
  - *Functional* – describes the capability's purpose (e.g., the ability to transfer money between user accounts.)
  - *Non-functional* – describes the capability's properties (e.g., the system can handle 1,000 concurrent users)

Adapting – Requirement Types

- Non-functional requirement types
  - Content
  - Quality (6 Types)
    - Functionality
    - Reliability
    - Usability
    - Efficiency
    - Maintainability
    - Portability

Adapting – Requirement Types

- Non-functional requirement types (continued)
  - System Environment
  - User Interface
    - Self-explanatory & intuitive
    - Usage-centered design
  - Evolution
  - Project Constraints

Adapting – Documentation

- 4 categories of notation
  - *Stories* – Plain-language scenarios; understandable to non-technical persons.
  - *Itemized Requirements* – Plain-language lists of requirements
  - *Formatted Requirements* – Accurately-defined, but allow for plain-language descriptions
    - Ex. Use case scenarios in UML
  - *Formal Specifications* – Expressed in formal syntax & semantics; rarely used in Web applications.

Adapting – Documentation

- So, what's best for a Web development project?
  - Low to medium accuracy is suitable for Web apps; formal specifications very rarely required.
  - Keep elicitation and management of requirements low.
  - Scalability is (most likely) important.
  - Formatted requirements (i.e. use cases) are heavily used.

Adapting – Tools

- Requirements Elicitation
  - EasyWinWin (the author's software)
  - Book: *Getting to Yes: Negotiating an Agreement Without Giving in* by Fisher, Ury, Patton (1994)
- Requirements Validation
  - Online feedback (Web surveys)
- Requirements Management
  - Database system – traceability, versioning

Challenges with Stakeholders

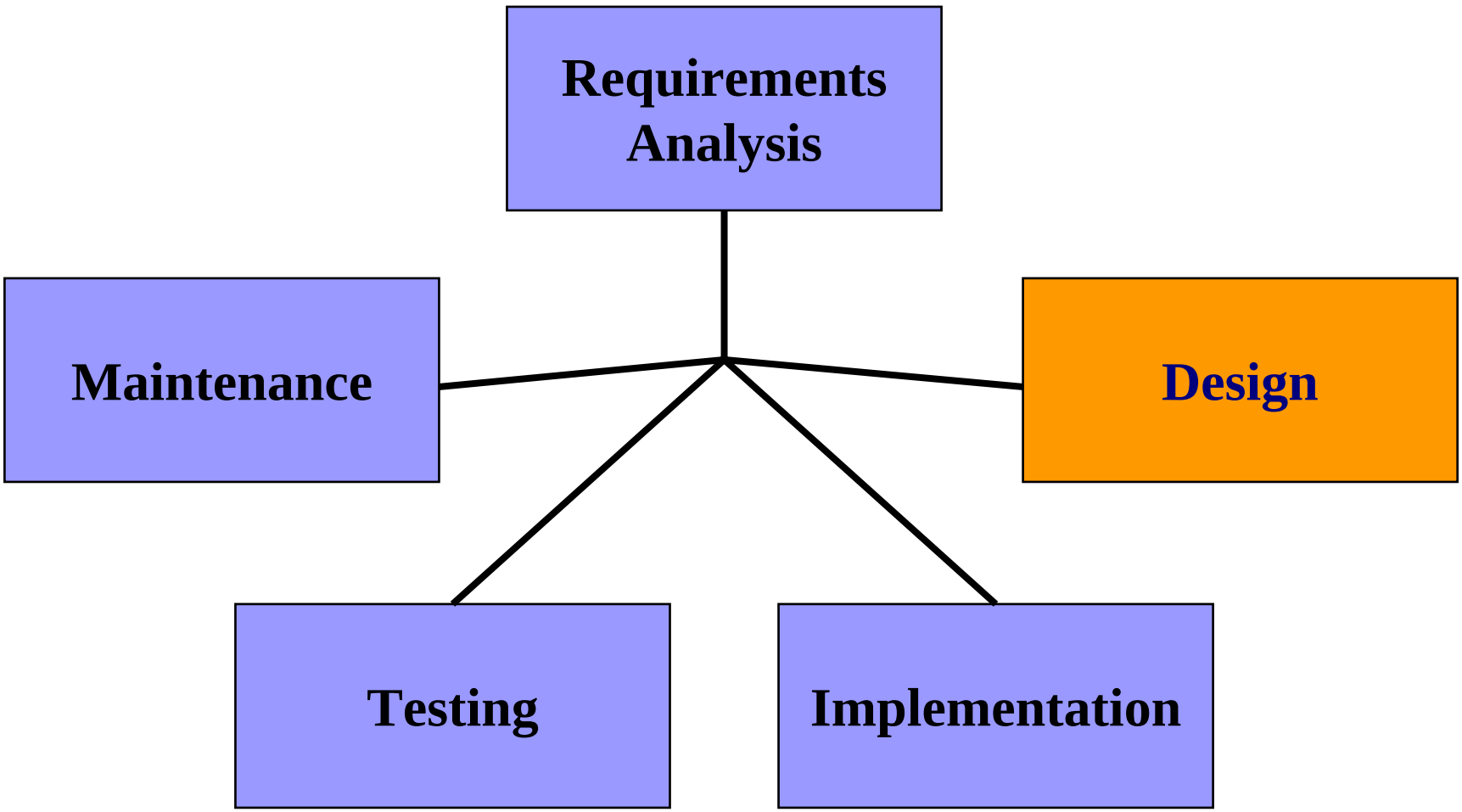
- McConnell (1996)
  - Users don’t know what they want.
  - Lack of commitment.
  - Ever-expanding requirements.
  - Communication delays.
  - Users don’t take part in reviews.
  - Users don’t understand the technology.
  - Users don’t understand the process.

Challenges with Developers

- Users and engineers/developers speak different “languages”.
- The tendency to “shoe-horn” the requirements into an existing model
  - Saves time for developers, but results may not meet user’s needs.
- Engineers & developers are also asked to do RE, but sometimes lack people skills and domain knowledge

2.3 Modeling Web Application

Summary – Web Engineering



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

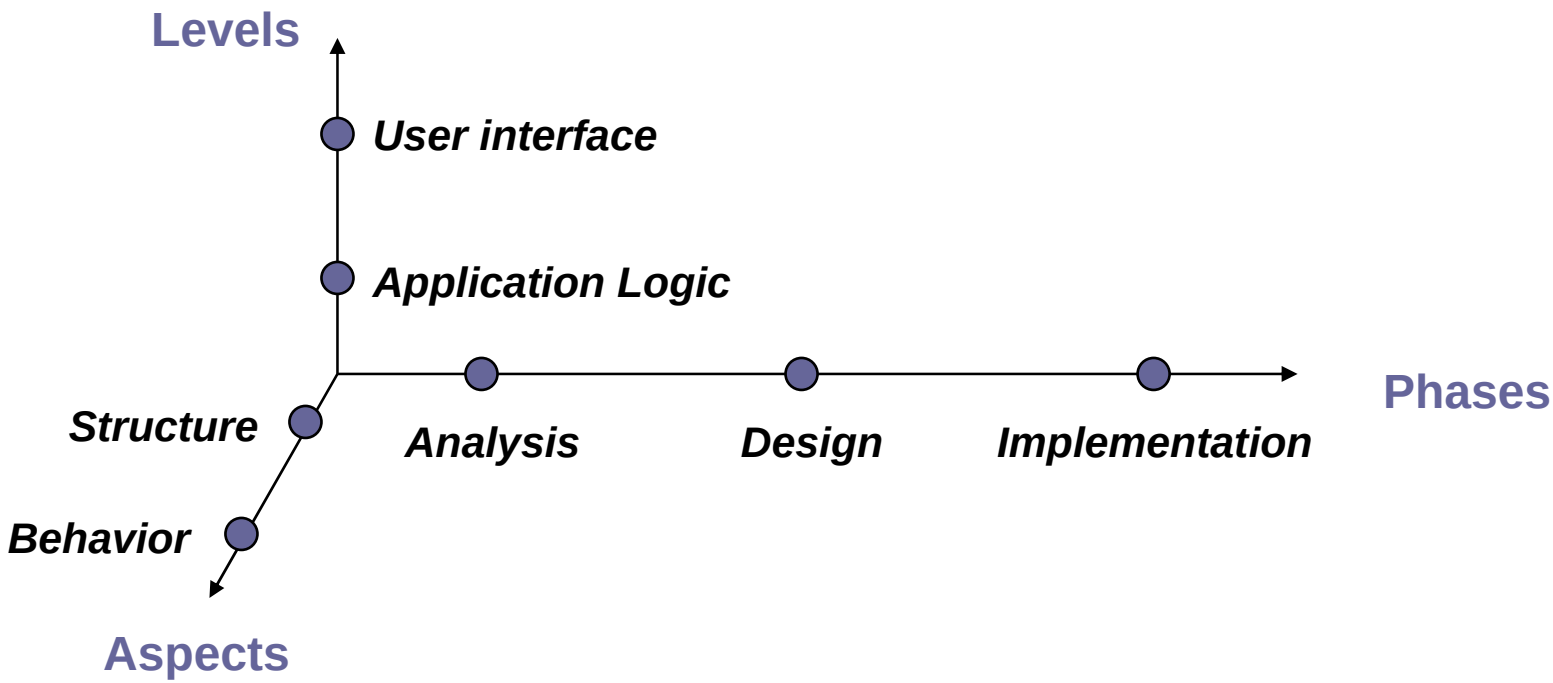
Web Modeling

- Modeling static & dynamic aspects of content, hypertext, and presentation.
- We focus on *object-oriented* analysis & design
  - *Analysis*: Finding & discovering objects/ concepts in a domain
  - *Design*: Defining 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



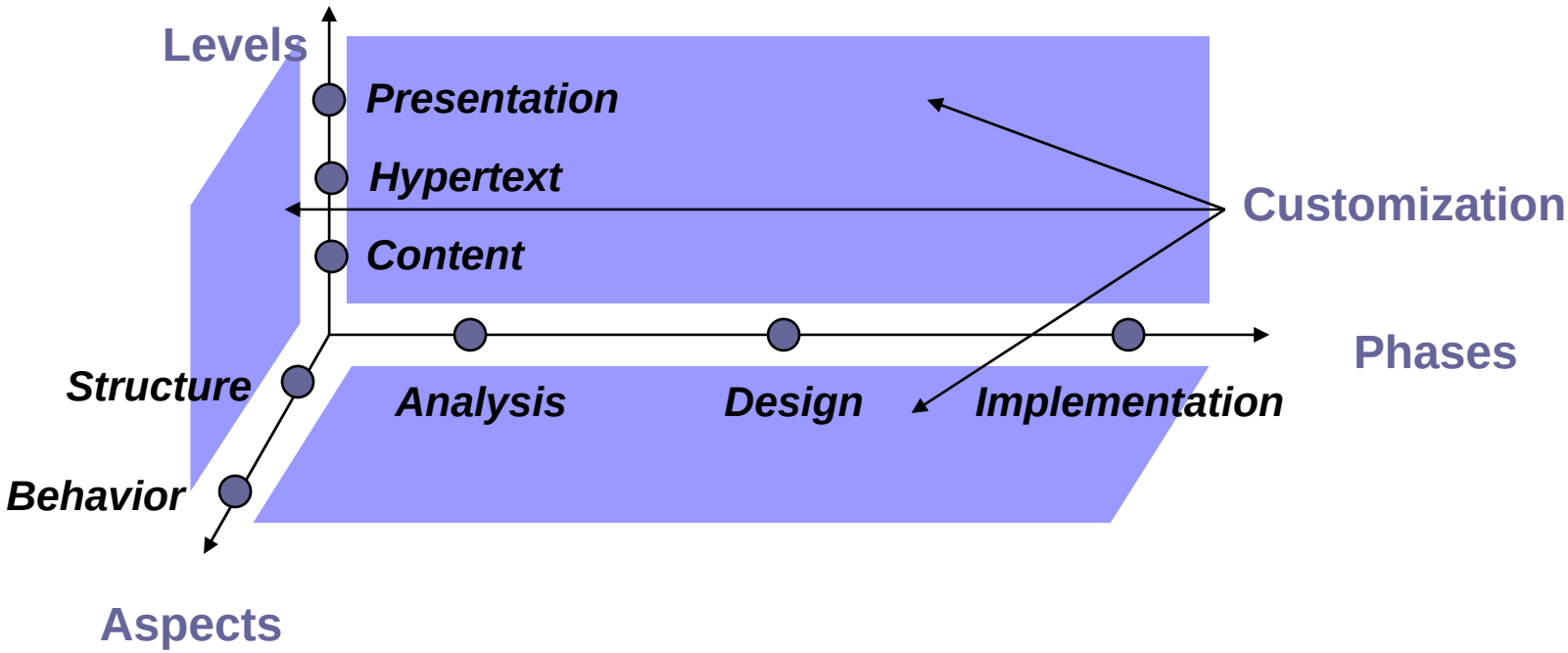
- *Levels* – the “how” & “what” of an application
- *Aspects* – objects, attributes, and relationships; function & processes
- *Phases* – Development cycle

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.
  - *Structural* – Class diagrams
  - *Behavioral* – Use Case diagrams, State machine diagrams

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

- 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)
  - 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

- 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 all 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

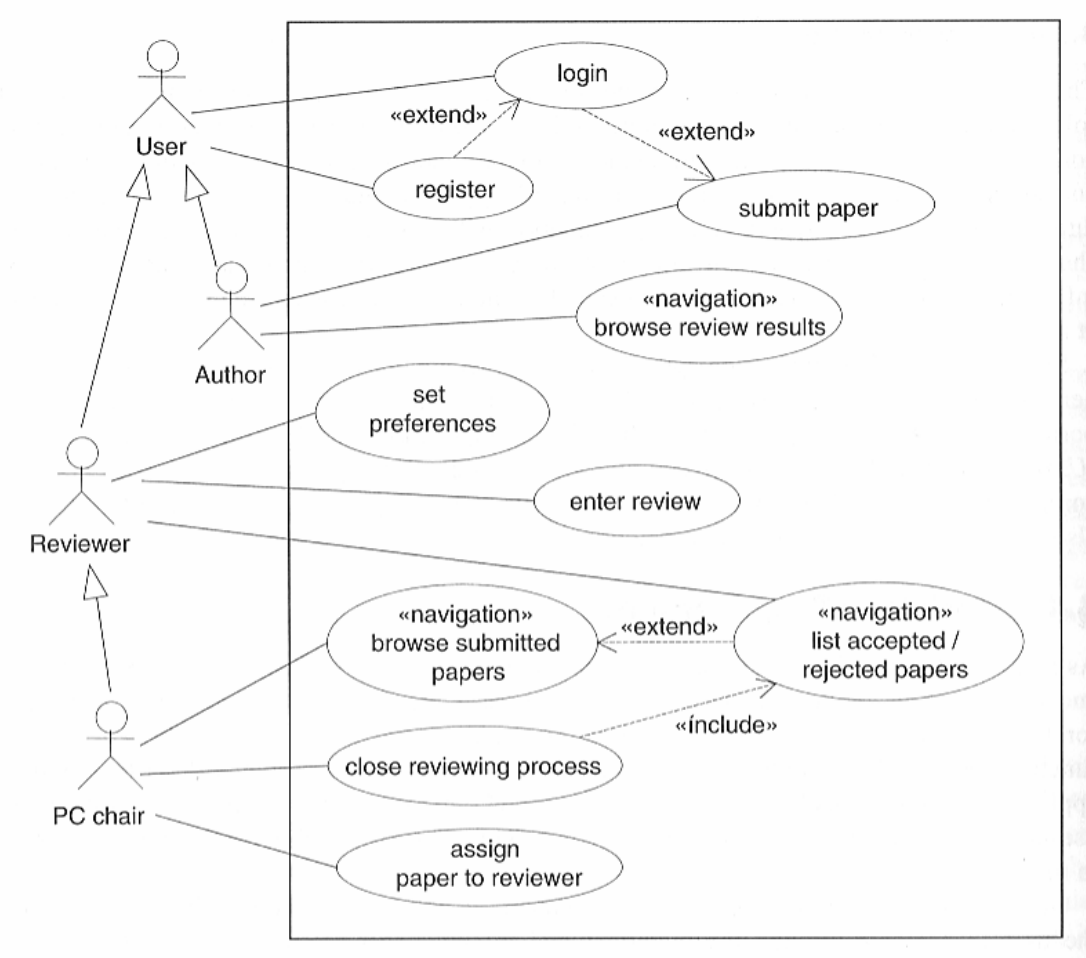
- Use shorts sentences
- Delete “noise” words
  - NO : “*The* System authenticates…”
  - YES: “System authenticates…”
- Avoid technology-specific terms (initially, at least)
  - NO : “Cashier swipes Product ID across scanner.”
  - YES: “Cashier enters Product ID.”

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



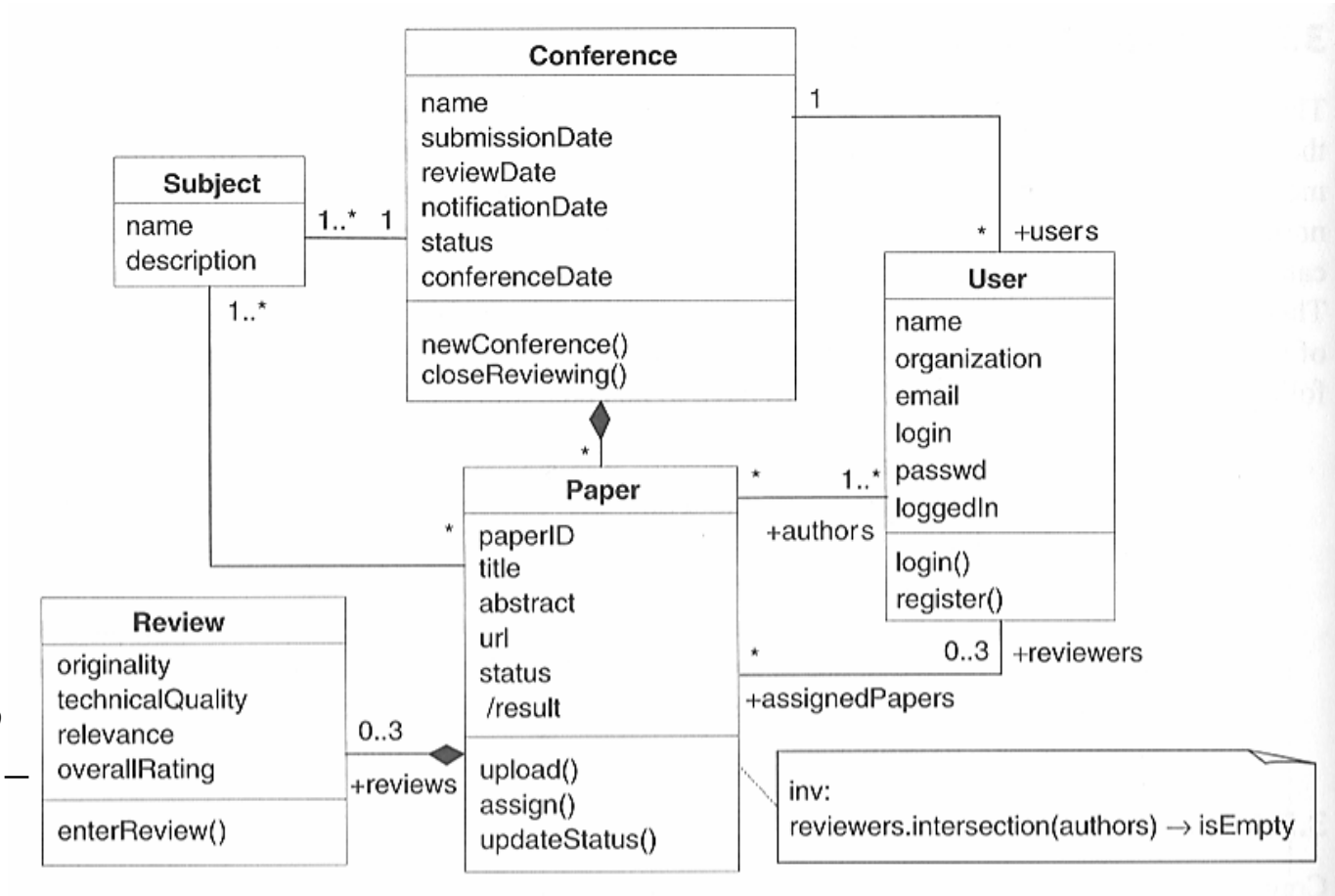
Source: Web Engineering – Kappel et al.

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

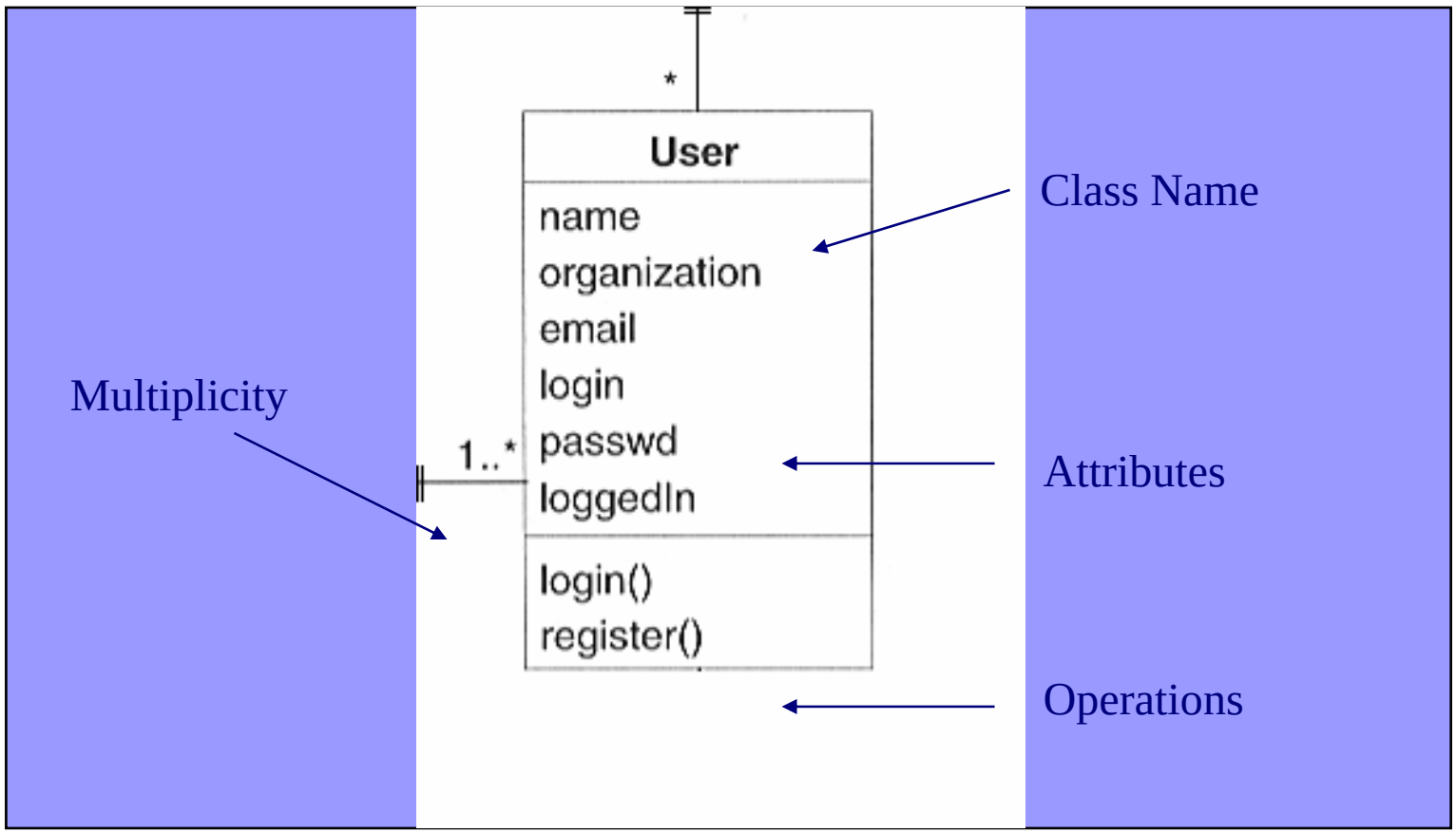
- Conference Paper Submission System



Source: Web Engineering – Kappel et al.

Class Diagrams

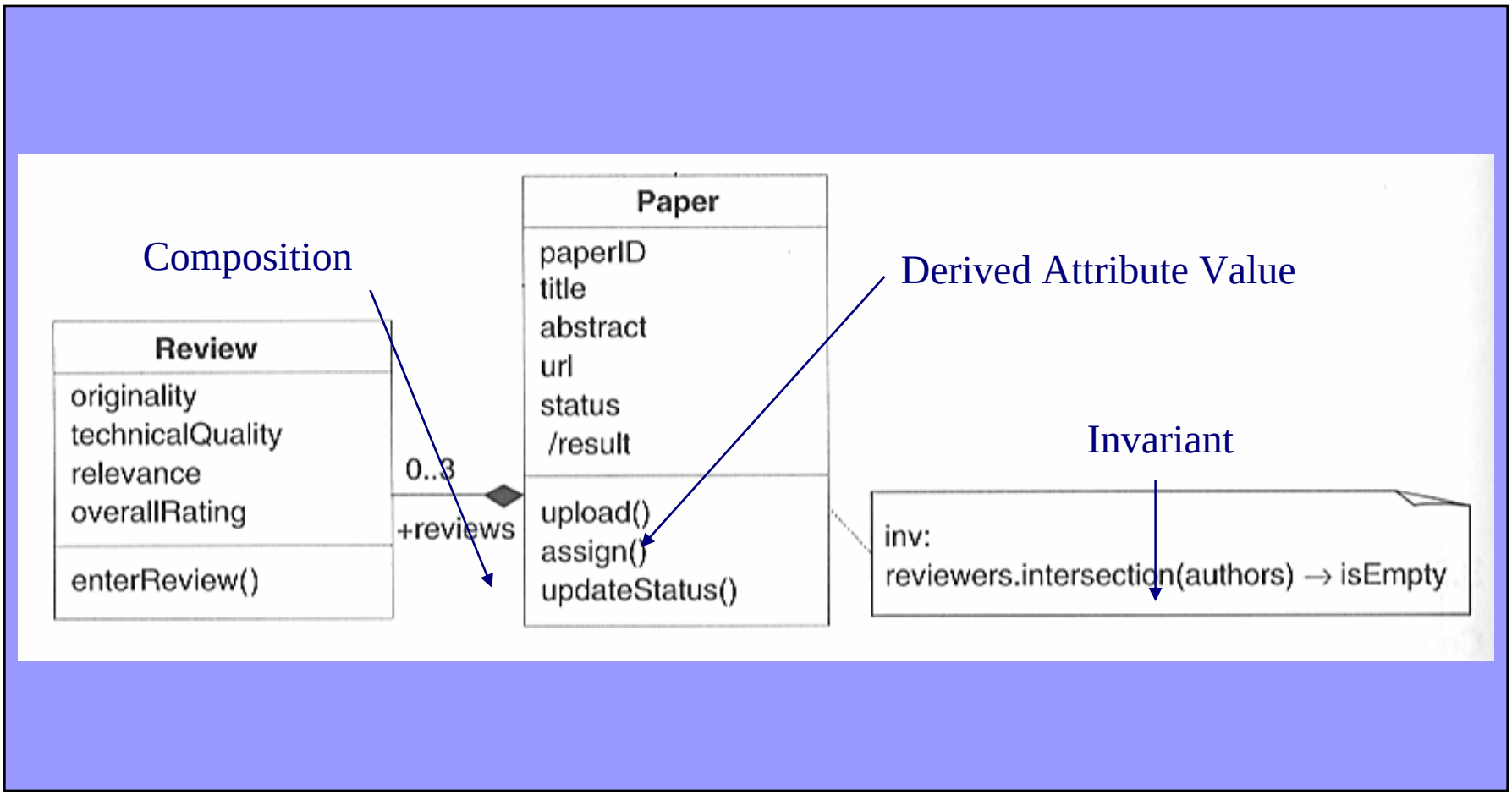
- Notations



Source: Web Engineering – Kappel et al.

Class Diagrams

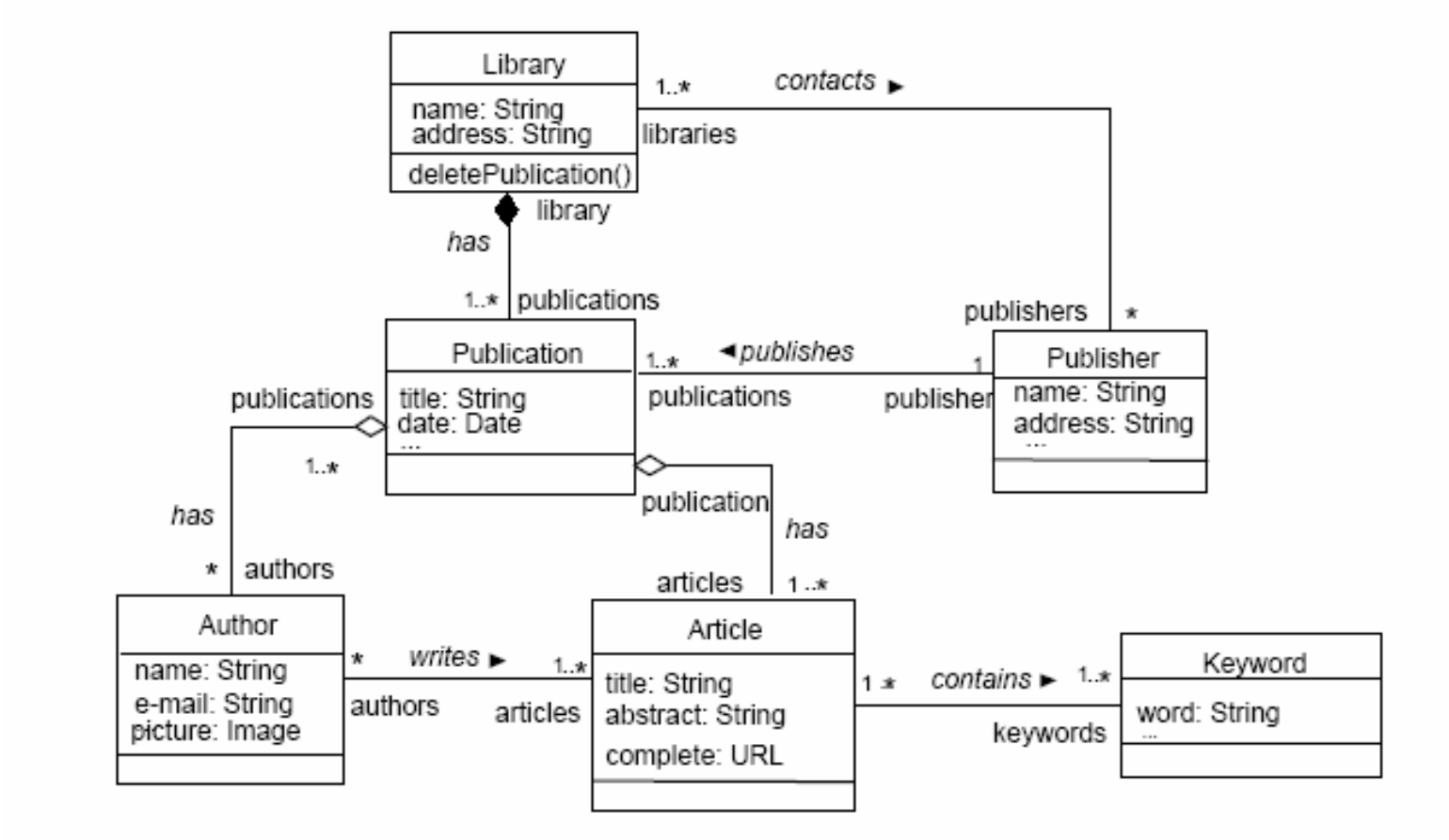
- Notations (continued)



Source: Web Engineering – Kappel et al.

Class Diagram – Example 2

- Online Library Application

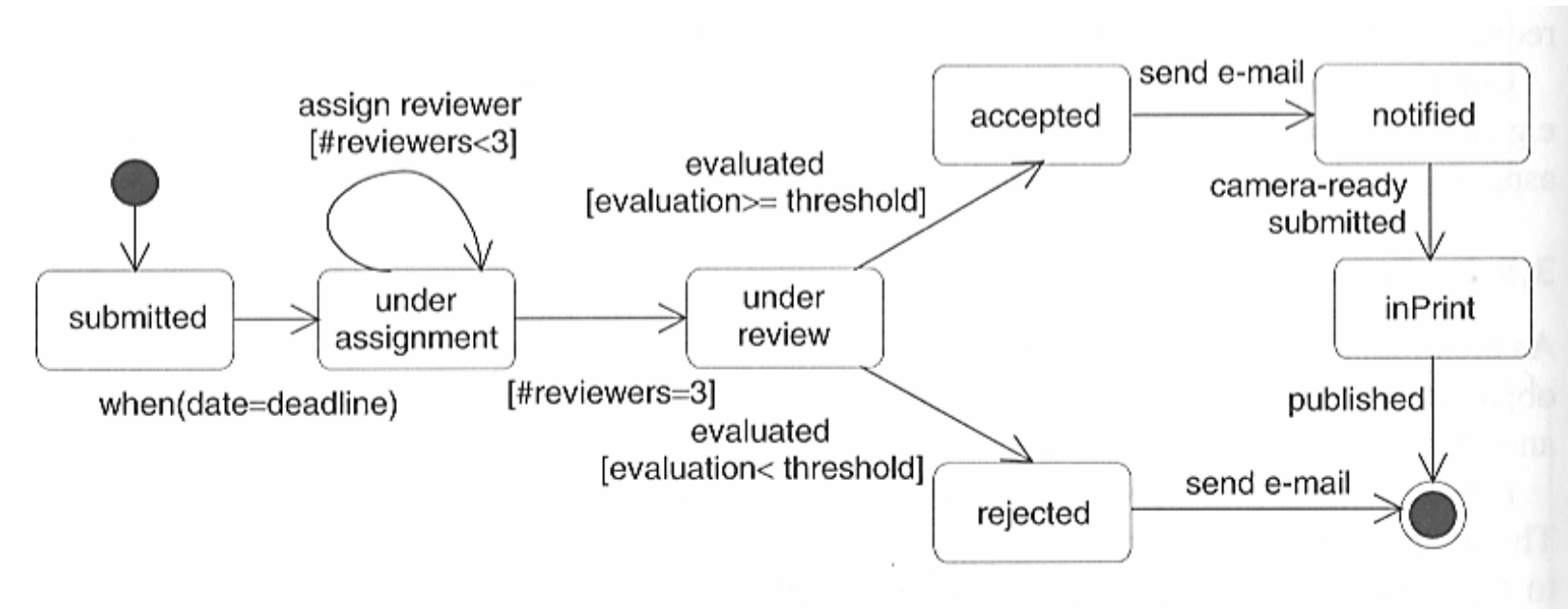


Source: Web Engineering – Kappel et al.

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



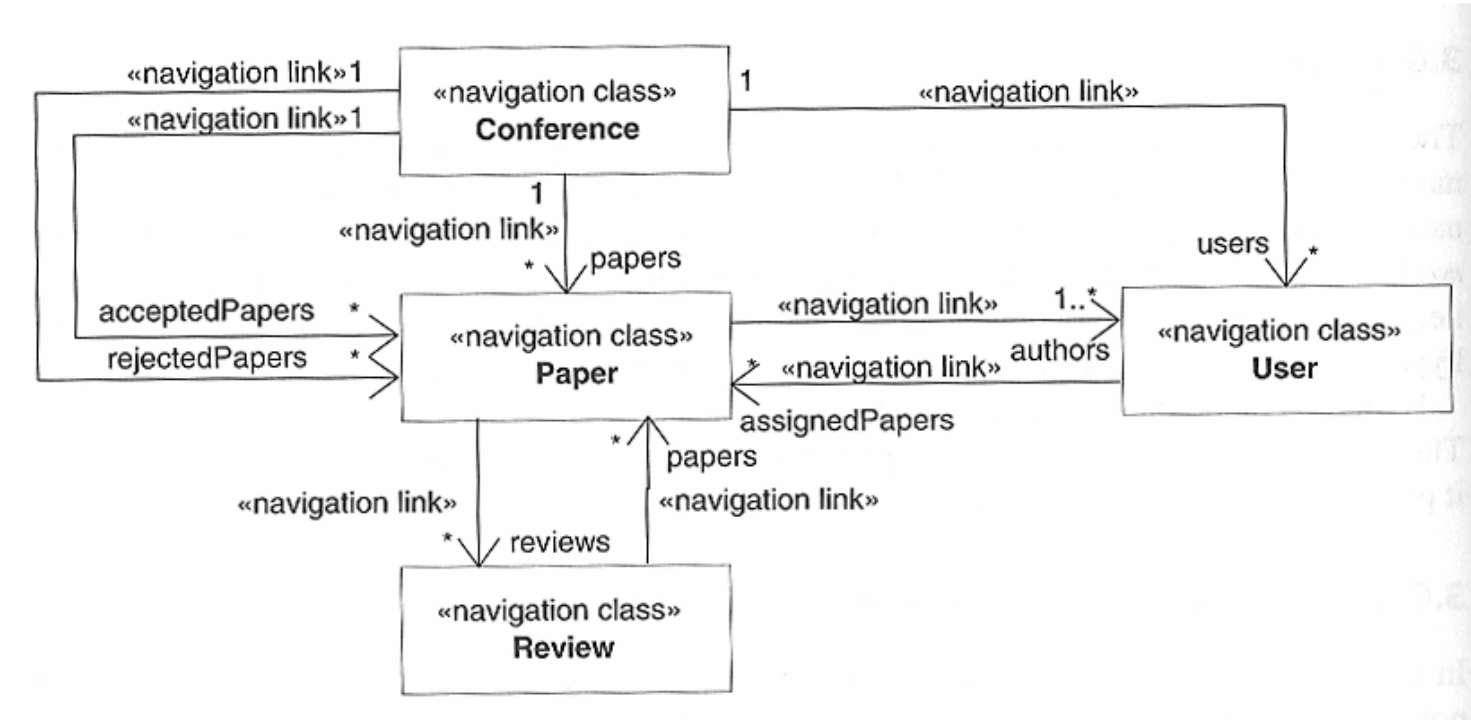
Source: Web Engineering – Kappel et al.

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

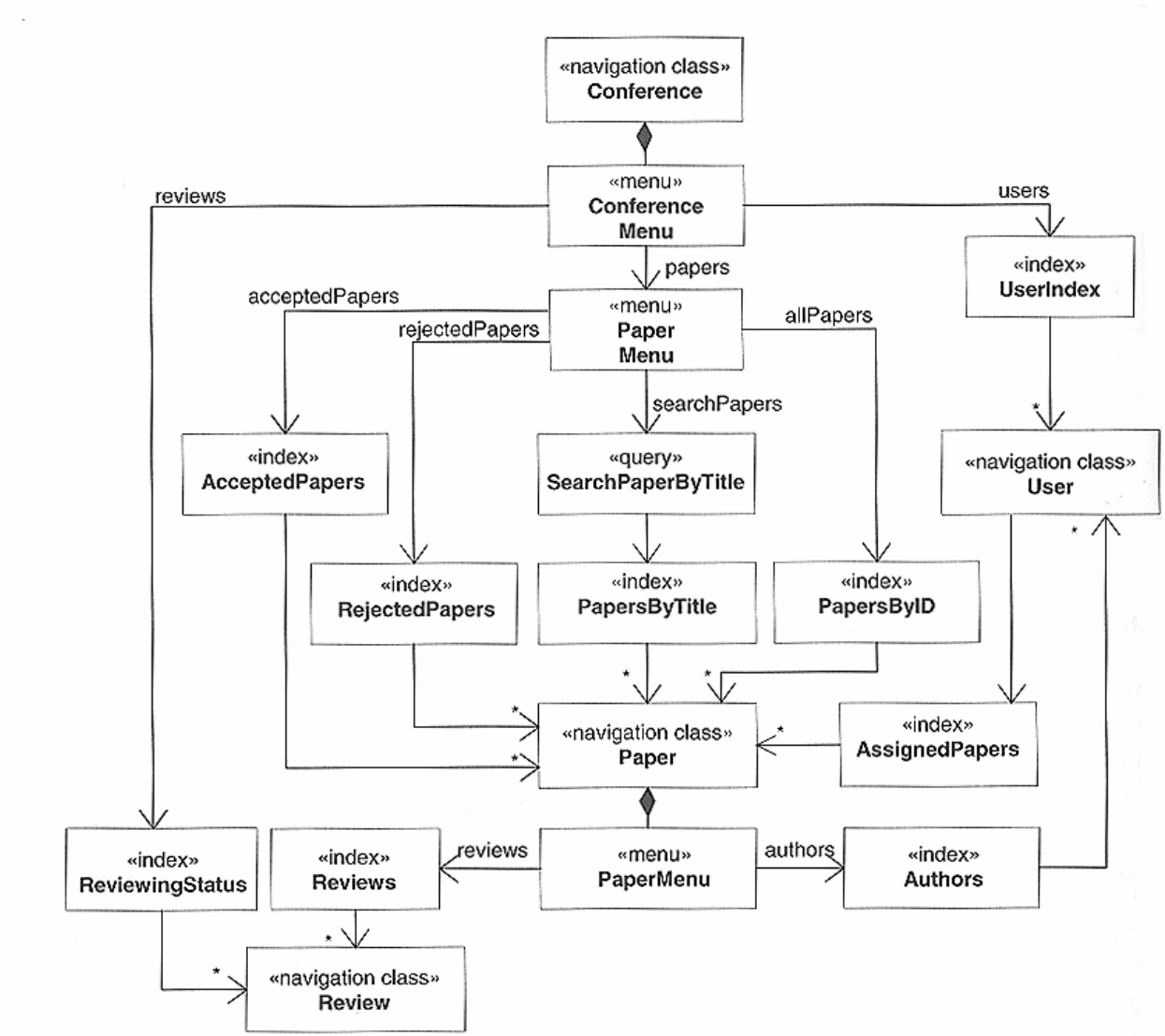
- UWE
  - Navigation vs. Process vs. External
- HDM
  - Structural vs. Perspective vs. Application
- WebML
  - Contextual vs. Non-contextual
  - Intra-page vs. Inter-page
- OO-H
  - I, T, R, X, S-links

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

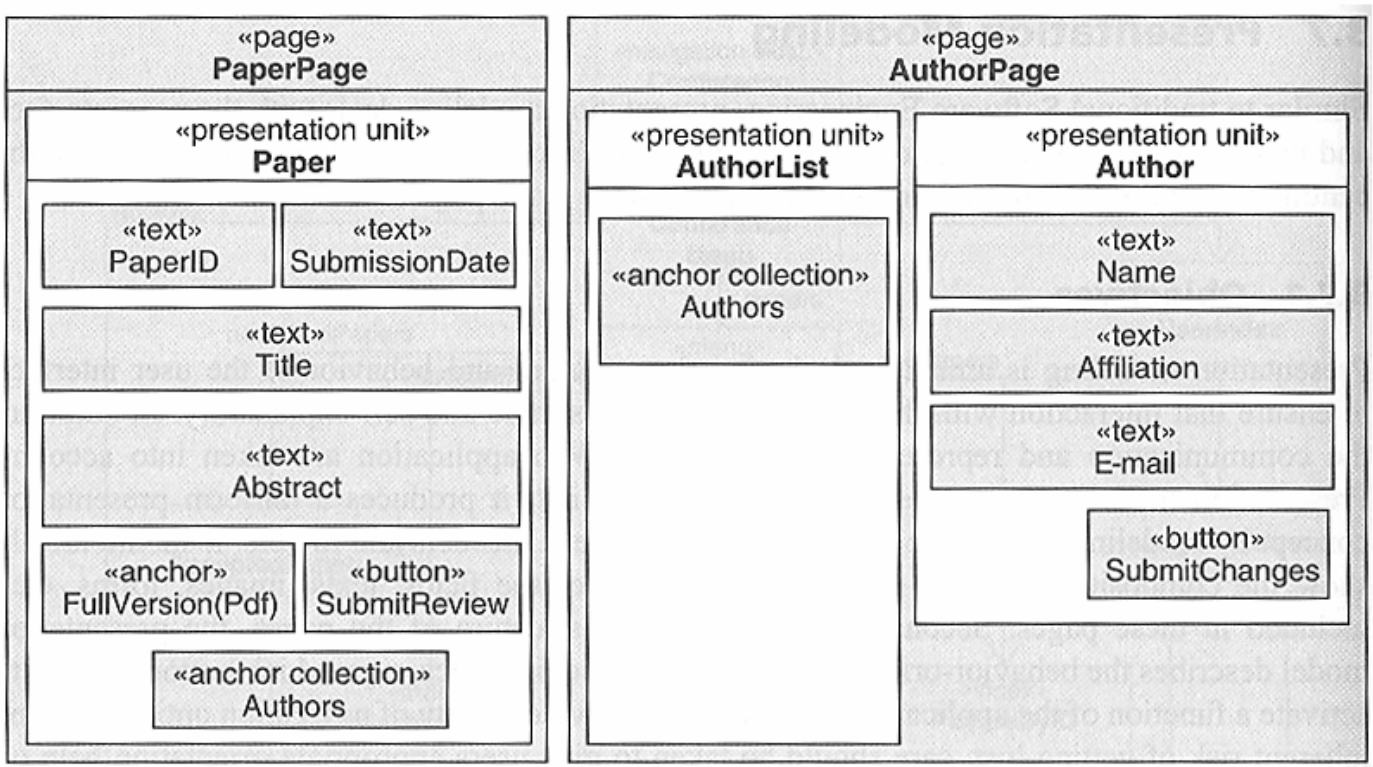
- *Purpose*: To model the look & feel of the Web application at the page level.
- The design should aim for *simplicity* and *self-explanation*.
- Describes presentation structure:
  - Composition & design of each page
  - Identify recurring elements (headers/footers)
- Describes presentation behavior: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

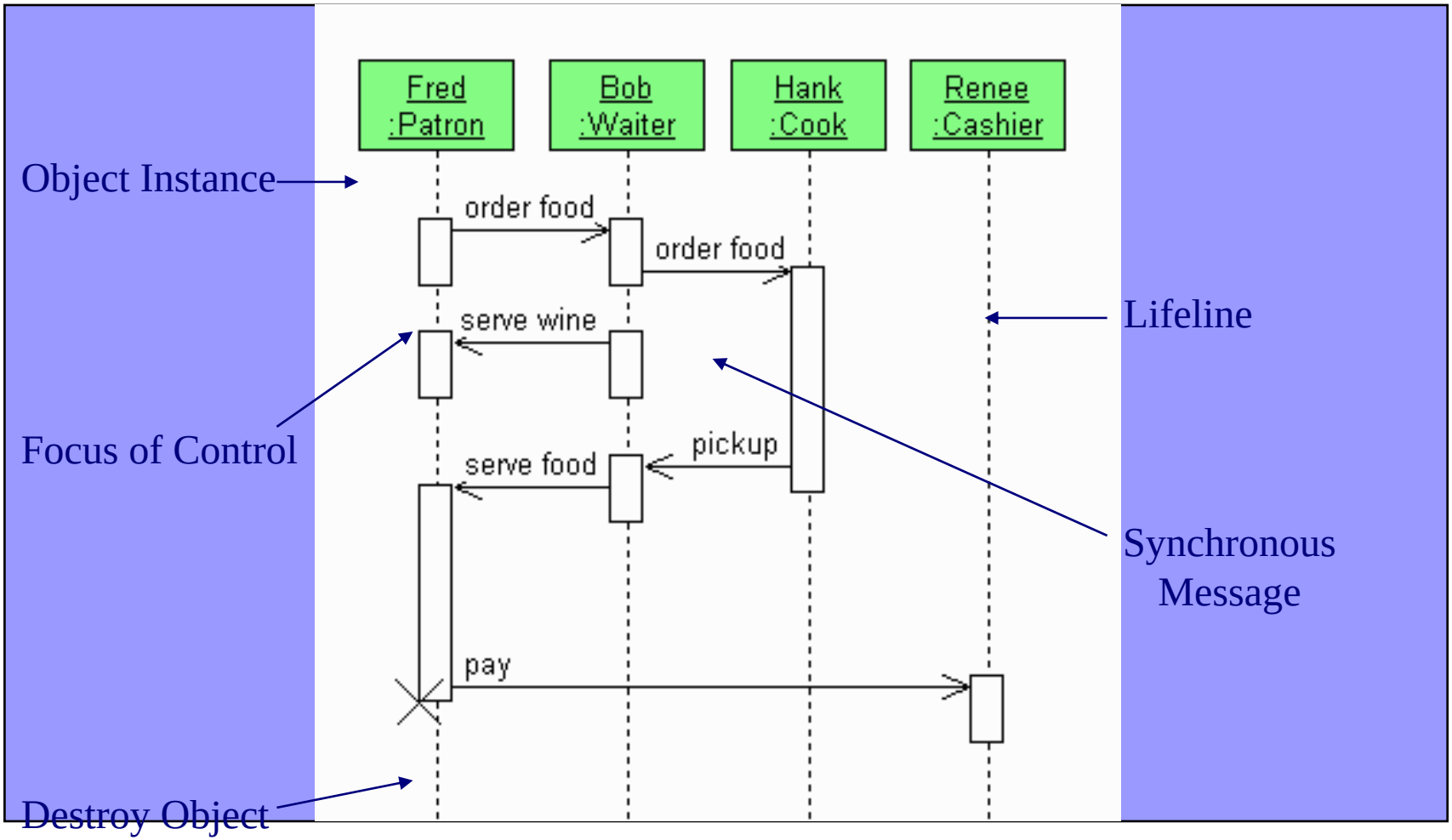


Source: Web Engineering – Kappel et al.

Sequence Diagrams

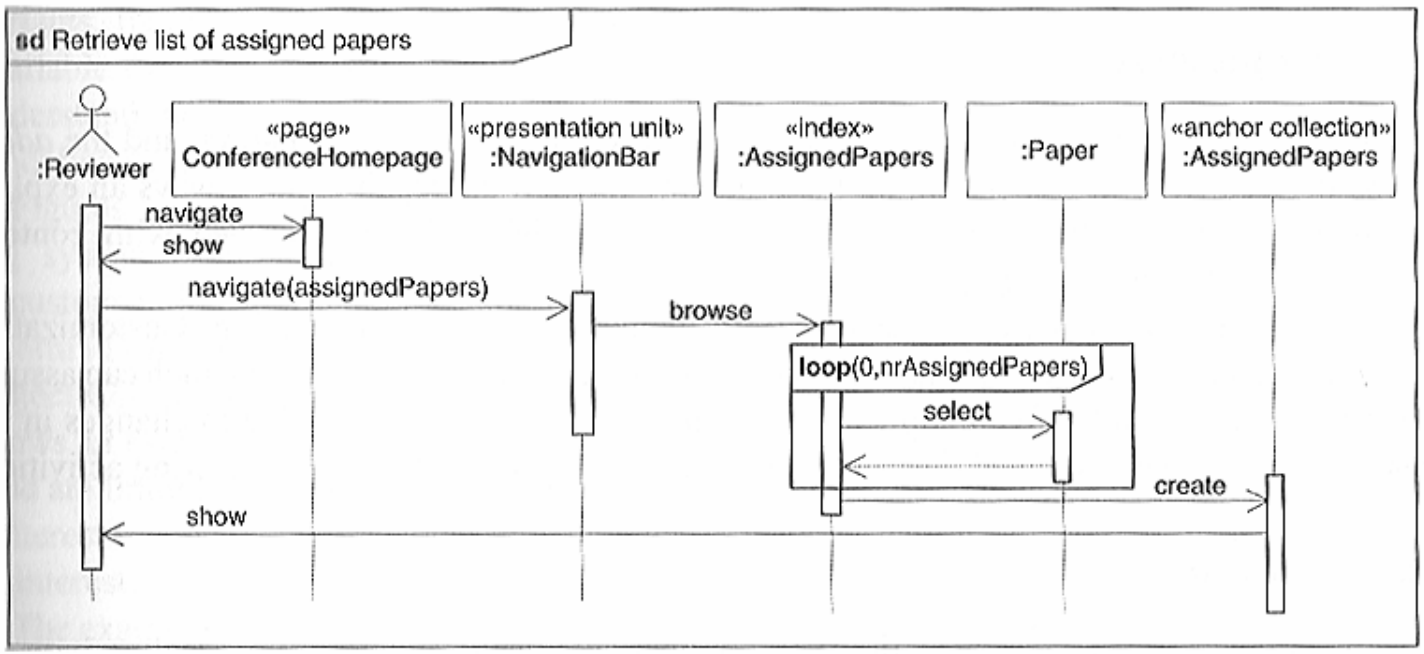
- *Purpose*: Depicts sequential interactions (i.e., the flow of logic) between objects in an application over time.
  - What messages, what order, & to whom.
  - Ex.: Object A calls method of Object B
  - Ex.: Object B passes method call from Object A to Object C.
- *Result:* Dynamic system interactions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

Sequence Diagram - Notation



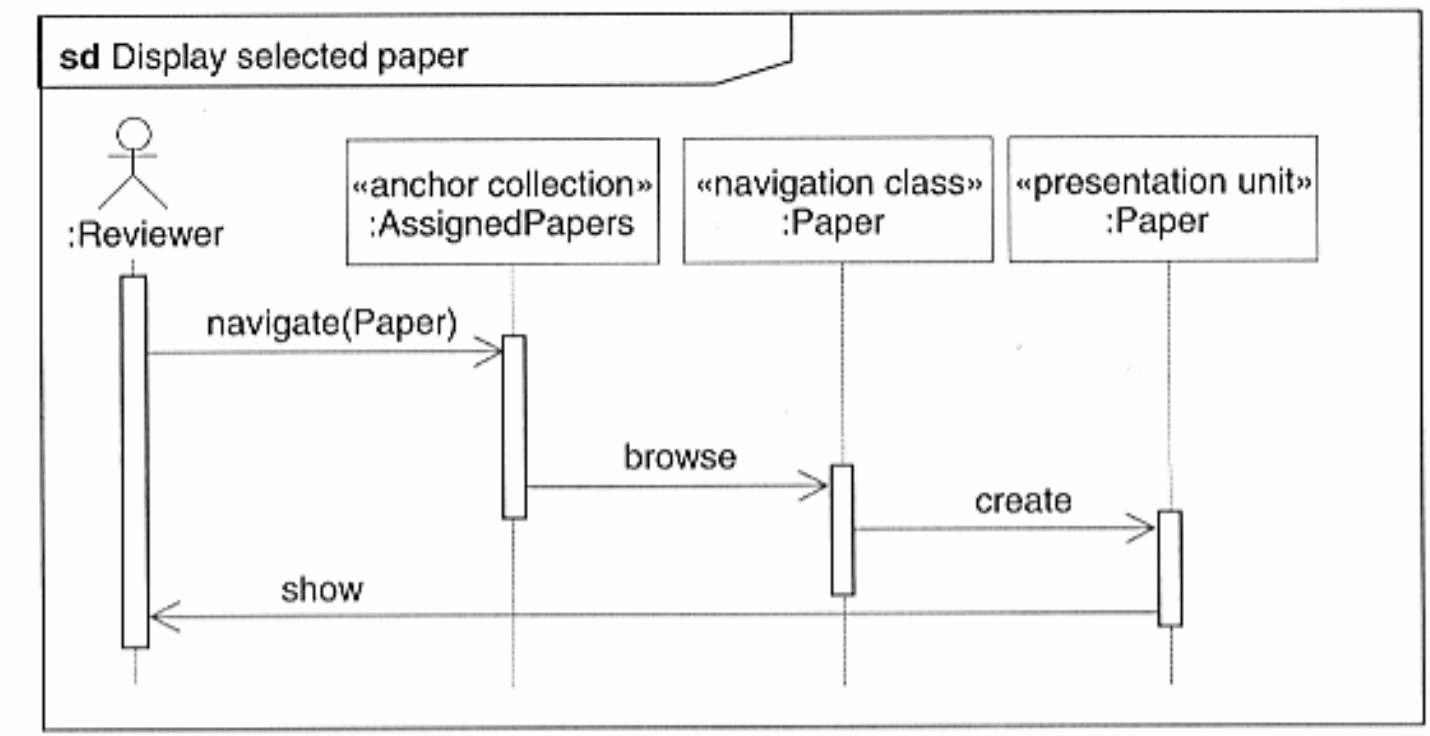
Source: Wikipedia – Sequence Diagram

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

2.4 Web Architectures

Overview

- Architecture defined
- Developing architectures
- Types of architectures
- Generic Web Architecture
- Layered-aspect architectures
- Data-aspect architectures

Architecture Defined

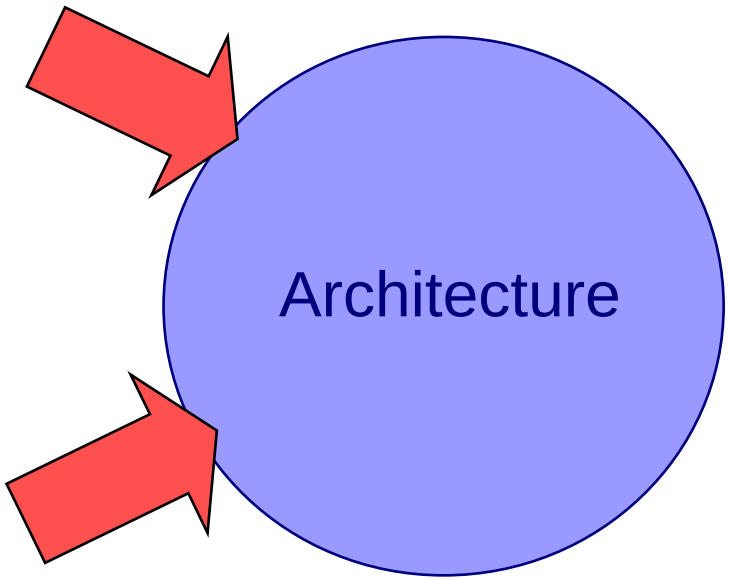
- Define “software architecture”
  - http://www.sei.cmu.edu/architecture/definitions.html
  - *“Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.”* – Eoin Woods
- Authors focus on 5 key attributes of software architectures
  - Structure, Elements, Relationships
  - Analysis => Implementation
  - Multiple viewpoints (conceptual, runtime, process & implementation)
  - Understandable
  - Framework for flexibility

Developing Architectures

- Influences on Architectures

Functional Requirements

- Clients
- Users
- Other Stakeholders

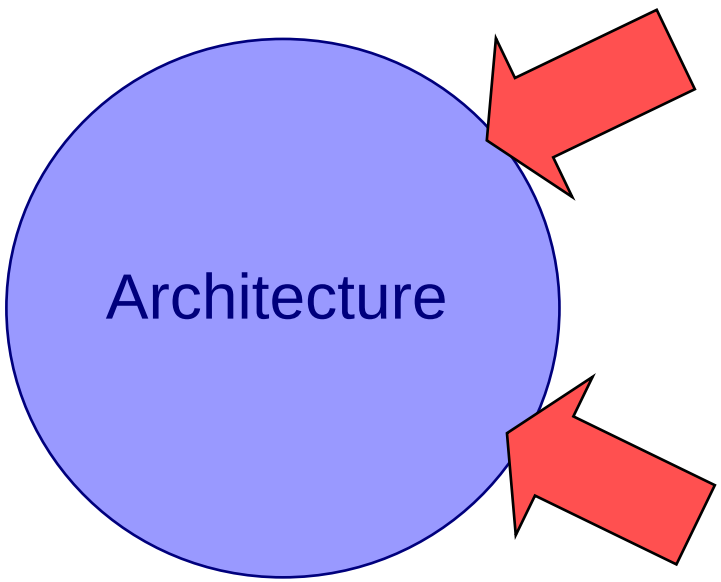


Experience with

- Existing Architecture
- Patterns
- Project Management
- Other?

Developing Architectures

- Influences on Architectures (continued)



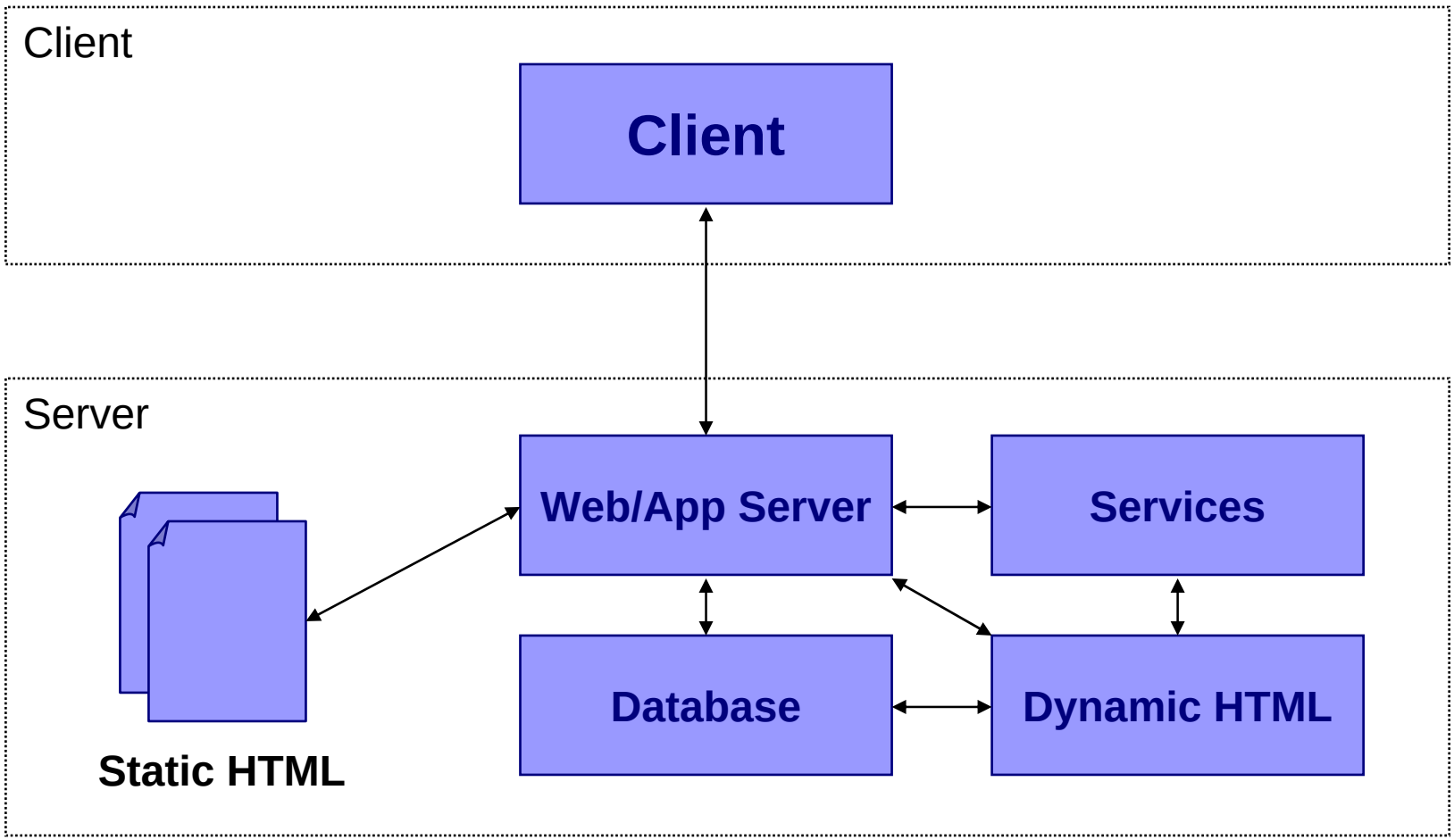
Quality considerations with

- Performance
- Scalability
- Reusability
- Other?

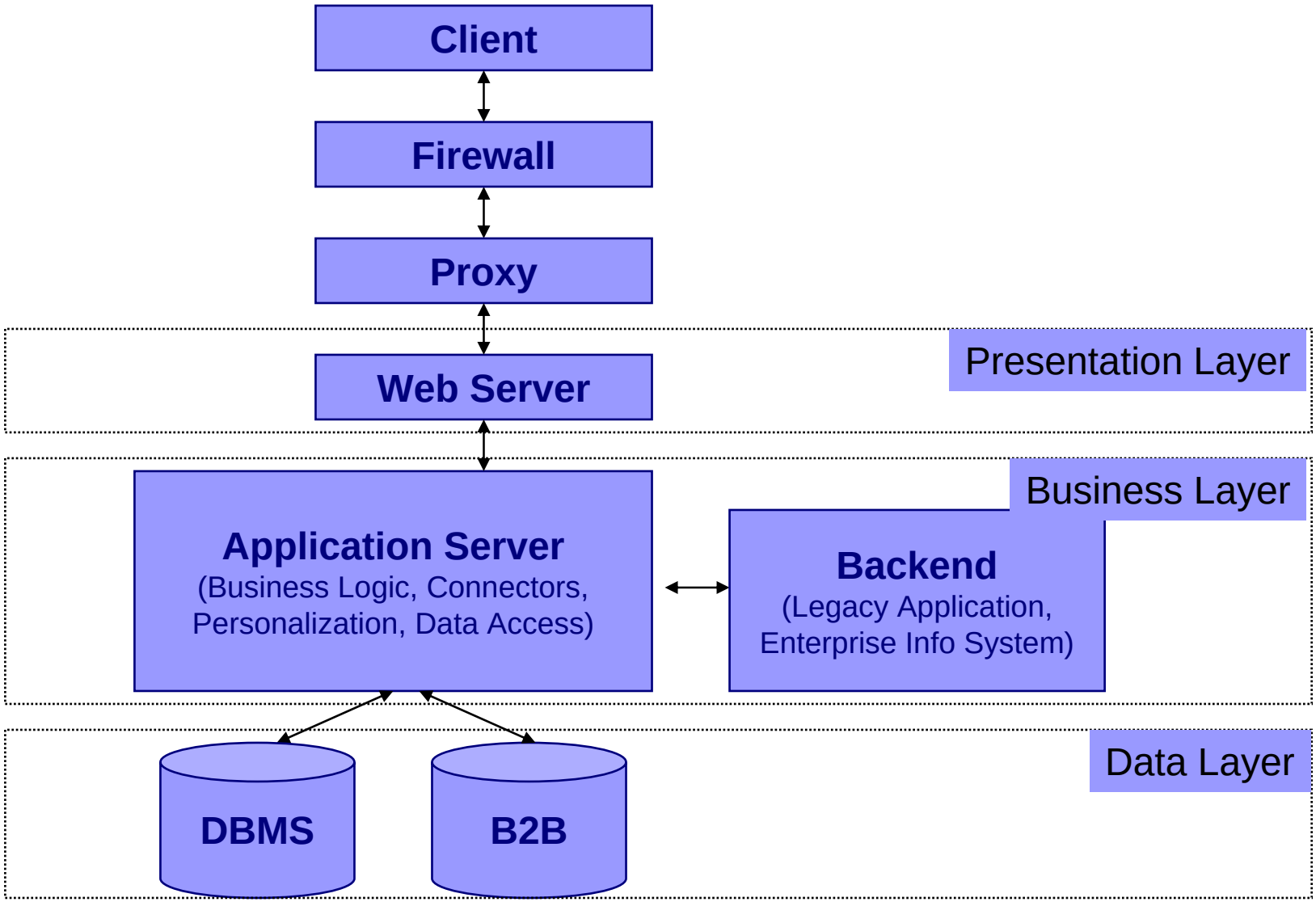
Technical Aspects

- Operating System
- Middleware
- Legacy Systems
- Other?

Client/Server (2-Layer)



N-Layer Architectures



Why an N-Layer Architecture?

- Separating services in business layer promotes re-use different applications
  - Loose-coupling – changes reduce impact on overall system.
  - More maintainable (in terms of code)
  - More extensible (modular)
- Trade-offs
  - Needless complexity
  - More points of failure

More on Proxies

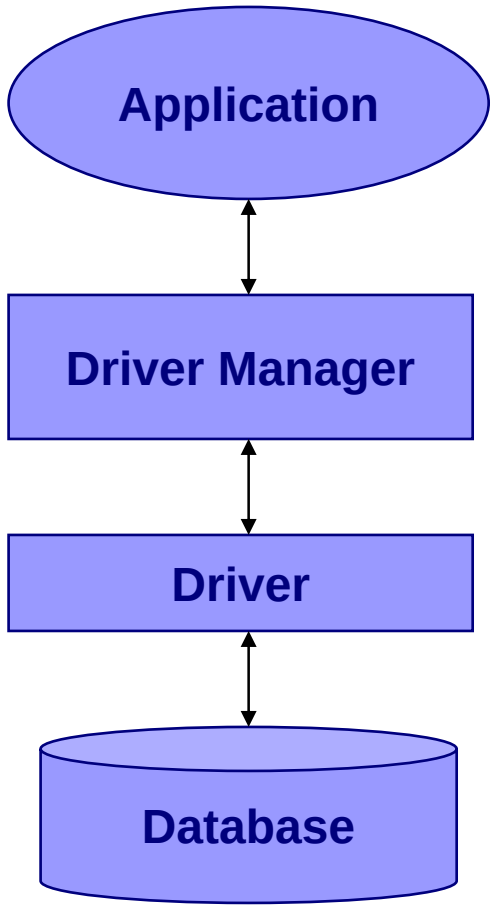
- Originally for *caching* data
- Can also server other roles:
  - Link Proxy
    - *Persistent URL's* – maps the URL the client sees to the actual URL.
    - *AJAX* – allows data from a 2nd server to be accessed via a client script.
  - History Proxy
    - HTTP is stateless - navigation history cannot be shared across multiple websites.
    - Multiple companies can access a server-side cookie (e.g. DoubleClick)

Integration Architectures

- Enterprise Application Integration (EAI)
- Web Services
- Portals/Portlets
- Challenges/Pitfalls
  - Cannot separate logic & data in legacy systems
  - Incompatible schemes
  - Poor documentation
  - Measuring performance/scalability

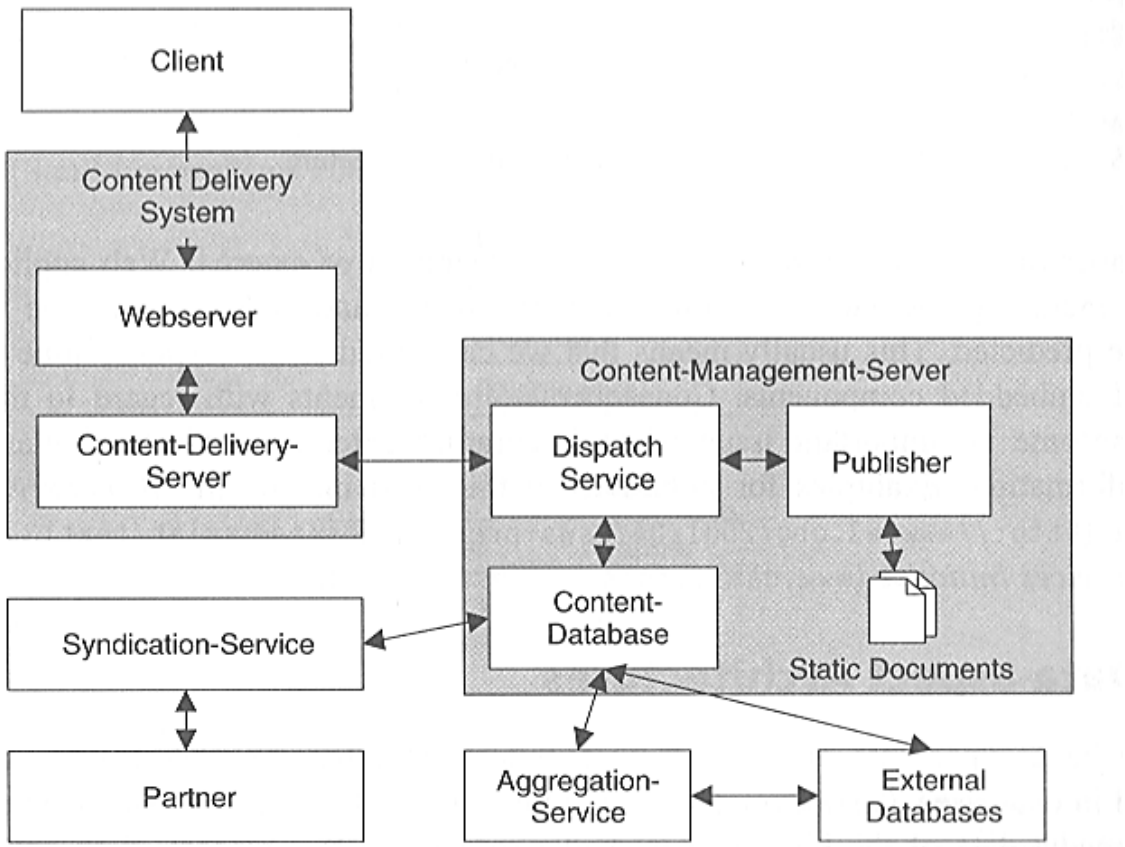
Data-Aspect Architectures

- Data can be grouped into either of 3 architectural categories:
  1. Structured data of the kind stored in DBs
  2. Documents of the kind stored in document management systems
  3. Multimedia data of the kind stored in media servers
- Structured data (JDBC/ODBC)
  - Accessed either directly via a web extension (for 2-tier) or over app server (for n-tier).
  - Since DB technology are highly mature, they are easy to integrate
  - Easy to implement
    - APIs are available to access DBs (e.g., JDBC, ODBC)



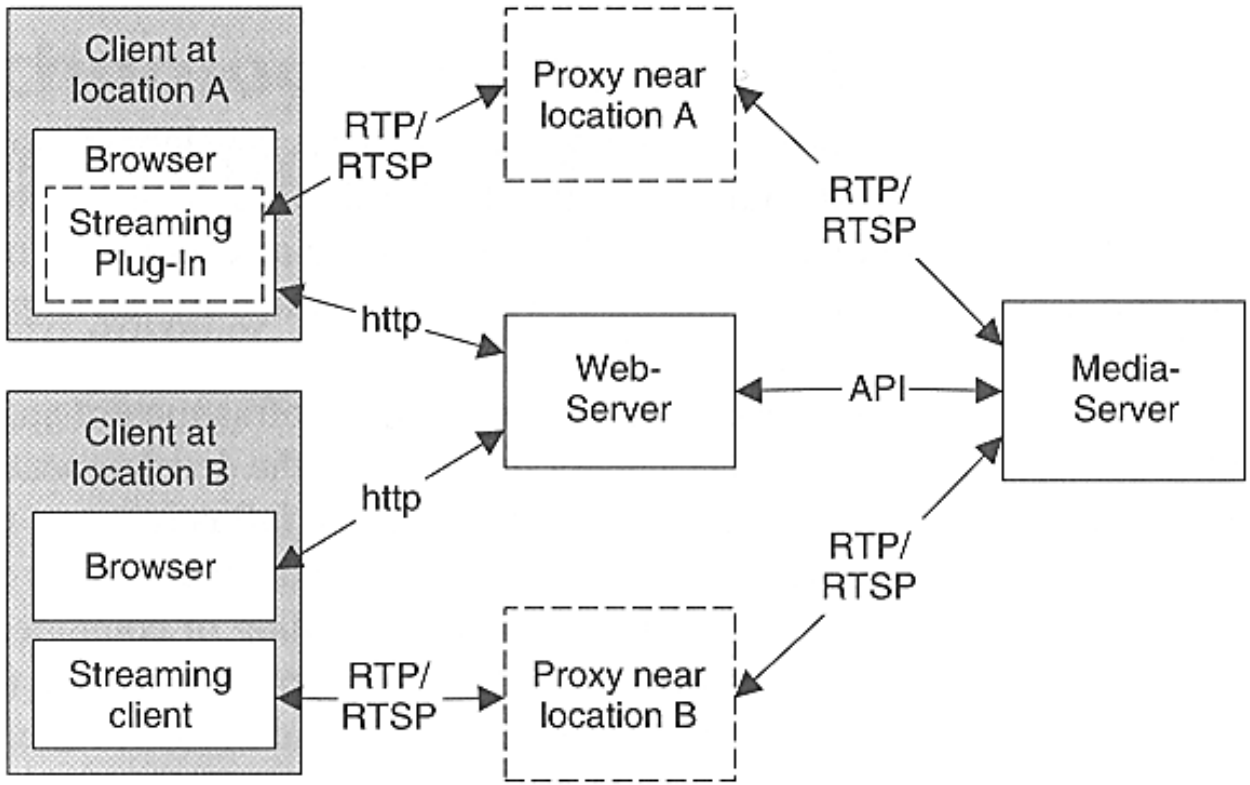
Data-Aspect Architectures

- Web Document Management



Data-Aspect Architectures

- Web Multimedia Management: Point-to-point



2.5 Web Accessibility

Overview

- The Case for Usability
- Defining Web Usability
- General Design Guidelines
- Usability Engineering
- Web Accessibility in Depth

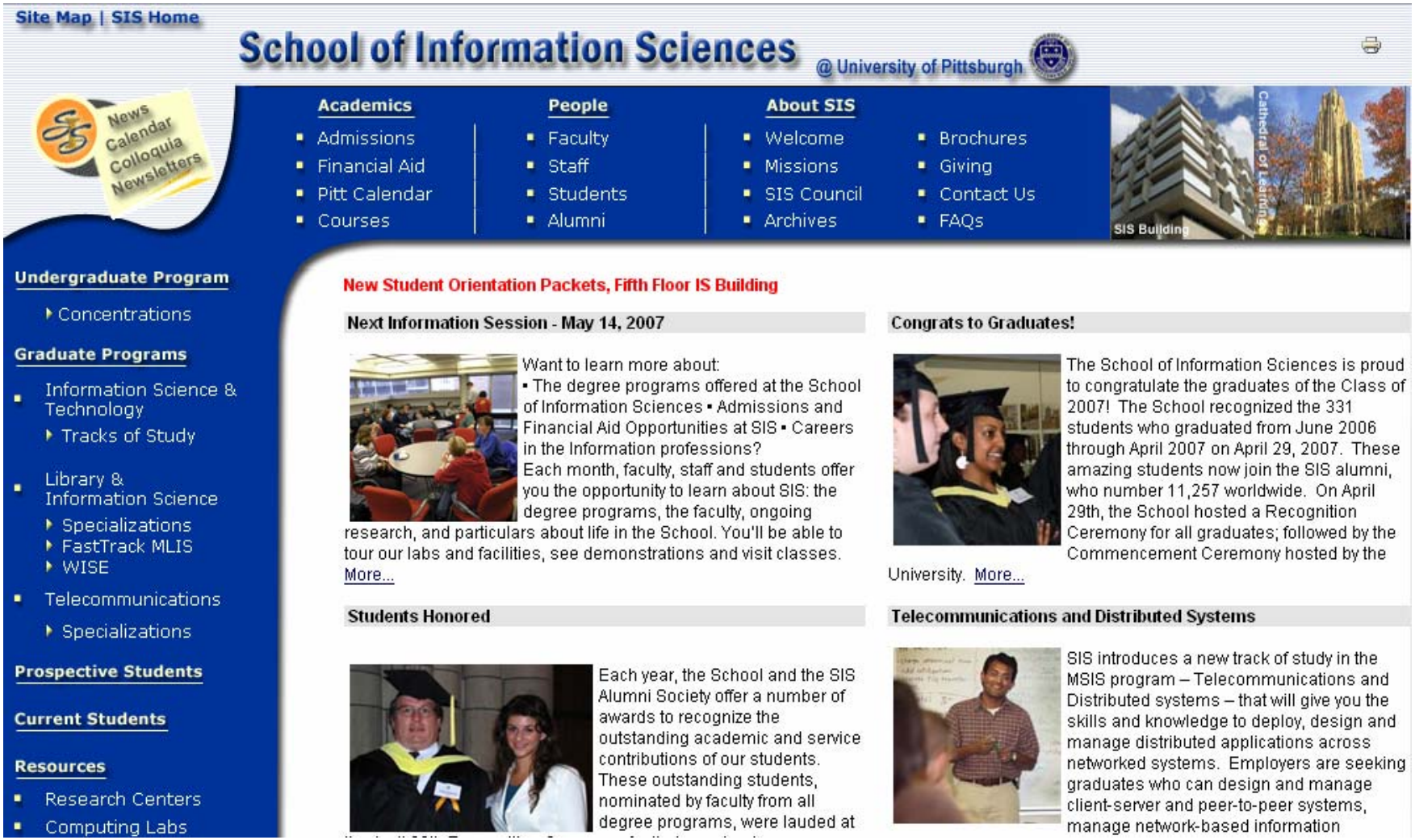
Why is Usability Important?

- "Mission critical" Web applications
- Poor design leads to lost time, productivity
- Your website speaks for your organization
  - Customers have choices
  - Easy come, easy go
- Diverse contexts
  - Proliferation of web-enabled devices
  - Increasing adoption by special needs groups – ex. seniors

Top 7 Gripes

- Contact information – address or phone number is buried
- Search function is not visible or unclear as to functionality
- No easy way to get back to critical points
- Pages that should load fast don't (e.g. Main page or key link page)
- Slow page loads are not incremental
- "What's new" is old
- Back button requires a repost of data

Example: SIS Website



Usability Defined

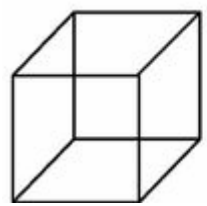
- ISO/IEC standard definition (1998):
  - "[T]he extent to which a product can be used by specified *users* within a specified usage *context* to achieve specified *goals effectively*, *efficiently*, and *satisfactorily*."
- Usability engineering is an ongoing, but critical process
  - Define user and task models
  - Iteratively test and reevaluate
  - User-based vs. expert methods

Defining Usability in Web Apps

- Traditional software usability specifics do not necessarily carry over to the Web:
  - People use your application *immediately*.
  - No manual or trainers
  - No salespeople
- How to categorize users?
  - First-time or returning?
  - Expert or novice?
  - Broadband or dial-up?
  - Desktop or mobile?

Human Information Processing

- Human cognition places a critical role in user interface design.
  - Perception
    - Positioning, grouping, arranging
    - Perceiving shapes and relationships
  - Memory
    - Limitations of working memory
    - Chunking, 7 ± 2 (Miller)
  - Attention
    - Focusing on one aspect
    - Movement, color schemes



General Design Guidelines

- Design guidelines represent best practices
- OK for “general” users
  - Normal cognitive ability
  - Normal audiovisual abilities
- Some guidelines may be inappropriate for audience members with special needs.
  - Ex. Navigation elements for schizophrenics
- More rigorous usability engineering techniques should be employed (later in lecture.)

Guidelines – Response Times

- As response times increase, user satisfaction decreases
  - Anything greater than 3 seconds, and the user becomes aware she's waiting
  - After 10 seconds, user gives up
- Optimize, or minimize graphics
- Consider breaking up large pages.
- <img> - use "width" & "height" attributes
- Don't forget your dial-up audience!
  - Home page size should be < 50Kb
  - Provide warnings (MPG – 2.5Mb)
- http://www.websiteoptimization.com/services/analyze/wso.php

Guidelines – Efficiency

- Minimize distance between clickable elements (while keeping effective sizing)
- Avoid frequent changes between mice and keyboards
- Tab-friendly for text-based browsers
- Minimize clicks to accomplish tasks (rule of thumb: no more than 4 clicks)
- Not so good: http://www.brown.edu

Guidelines – Colors

- Colors have different meaning depending on your audience
  - Cultural differences
  - Domain-specific meanings
  - Warm vs. cool colors
- Minimize the number of colors
- Avoid extreme hues, highly saturated colors
- How does your site look on a CRT? LCD?
- Supplement colors with other visual aids for those with limited color vision.

Guidelines – Text Layout

- Screen vs. Paper
- Consider different window sizes
  - Avoid multiple columns
  - Avoid fixed width
- Readability
  - Sans-serif for screen, serif for print
  - Avoid patterns, low-contrast background
  - Short paragraphs
- Allow for user-selected font-sizes

Guidelines – Page Structure

- Display considerations
- Use relative positioning over absolute.
- Vertical scrolling is fine; horizontal scrolling is NOT.
- Important elements should ALWAYS be visible.
- Make page print-friendly or provide alternative style and print button.
- Not-so-good: http//www.arngren.net

Guidelines – Navigation

- Provide your user with a mental model of the site ASAP.
  - Intuitive navigation elements
  - Site map
  - Breadcrumbs
- Pulldown menus?
  - Pros: Efficient use of space
  - Cons: Key information is hidden
- Not-so-good: Brown Univ. (circa 2005)

Guidelines – Multicultural

- Location is typically not a constraint on the Web.
- "Lowest common denominator" applies:
  - Avoid over-expressive colors
  - Symbols
  - Language
  - Information representation (date/time formats)
- Present form elements consistently
- Self-selection?

Guidelines – Establishing Trust

- Loyalty is fleeting, but instilling confidence during a transaction is highly critical

- Ways to build trust:
  - About us
  - Easy-to-access Contact Information
  - Interaction mechanisms (FAQ, chat rooms)
  - Security & privacy policies
  - Exchange and warranty policies
  - Customer relations management

Guidelines – Animations & Icons

- Remember human attention – animations are typically distracting
  - Draw attention to an important function
  - Explain something
- Iconography should be used to support navigation understanding
  - Map to commonly-known metaphors
  - Use redundant text and “alt” text!
  - Not appropriate for (some) cognitive-impaired users
- Not-so-good: http://www.globalaigs.org/

Guidelines – Consistency

- Consistency keeps learning to a minimum; users don't want to have to think!
- Identity can be set by consistent components
  - Header: home, logo, navigation, search, help
  - Footer: author, modification, contact
- Consistent design helps users avoid getting lost, especially when jumping to different sub-units of an organization.

Usability Engineering

- Consists of 4 phases that are essentially parallel to the Web Engineering process
  - Requirements Analysis
  - Design
  - Implementation
  - Operation

User-Centered vs. Usage-Centered

Phase	Focal Points	
	User-Centered *(Traditional)*	*Usage-Centered* *(Web)*
Requirements	Meetings, interviews, focus groups	Competitive analysis; Task analysis & models
Design & Implementation	User requirements Direct user participation	Models Inspection & remote testing
Operation	Training, evaluation of help-desk logs	Log file analysis; server stats; user feedback analysis

Requirements Analysis

- Systems Analyst & Usability Expert take the lead:
  - Competitive Analysis
  - Define qualitative/quantitative goals
    - Information, Entertainment, Exchange (Siegel)
    - Make them concrete and testable!
  - User-centered: build user profiles
  - Usage-centered
    - Task analysis
    - Ease-of-use or Ease-of-learning?

Interaction and Design

- Initially, the Interface Designer builds a *conceptual* model
  - Based on core use cases
  - Shows the basic structure
- Getting feedback from potential users
  - Storyboards & Paper Mock-ups
  - Card-sorting (Navigation)
- Usability expert provides input after this first round.

Interaction and Design

- Designer and coders can then elaborate on the details
- Additional user testing:
  - *Prototypes* – exhibit some functionality
  - *Usability Tests* – real context, real tasks.
- Remote usability testing
  - Sample of representative users
  - Client-Logging software
  - Web-cams if possible
  - Better external validity & lower costs(?)

Coding and Post-Deployment

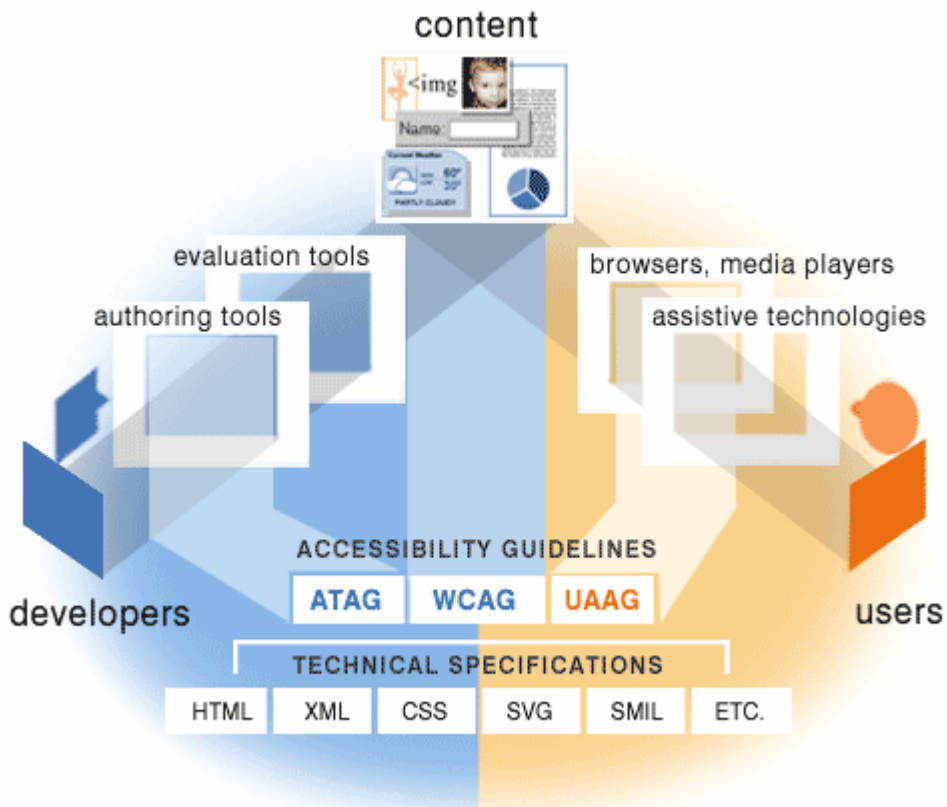
- Usability Expert assumes the role of the Quality Assurance manager.
  - Consistency?
  - Observed guidelines & standards?
  - Adhered to (current) requirements?
- Bring same users back in for testing, if possible.
- Document, document, document!

More on Web Accessibility

- People with disabilities are adopting the Web in greater numbers.
- Tim Berners-Lee stressed universal access to the Web as essential.
- 20% of the world's population have disabilities in at least one of the senses.
- *Key take-aways*:
  - Designing for special needs doesn't necessarily require reinventing your application.
  - Doing so can also help "general" users

Web Accessibility Initiative (WAI)

- Web Content Accessibility Guidelines 1.0 (WCAG, 1999) published by the W3C's WAI
- 3 Priorities
  - 1) Must
  - 2) Should
  - 3) May
- Defines Groups
- WCAG 2.0?



Special Needs Groups

- WAI identifies the following special needs groups:
  - Visual
  - Hearing
  - Physical (Motor)
  - Speech
  - Cognitive
  - Age-related

Visual Considerations

- High-contrast color schemes
- Large font sizes; ability to change fonts
- Use alt attributes!
- <label-for> tags in forms
- Avoid frames
- Access key attributes, and rapid tabbing
- Many software packages for text-to-speech
  - Some integrate with browsers
  - OK Firefox plug-in: FireVox
- Good example: http://www.afb.org

Aural Considerations

- Captioning audio and video
  - Synchronized Multimedia Integration (SMIL)
  - Good QuickTime, RealAudio Support
  - W3C standard
- Complement text with simple images
- Clear, simple language

Physical (Motor) Considerations

- May require specialized hardware
  - Mice
  - Keyboards
  - Voice Recognition
- Avoid elements that require time-dependent responses or precise mouse movements.
- Access key attributes
- Consistent tab ordering in forms.

Cognitive Considerations

- Most neglected of the groups
  - Little research in terms of Web usability
  - “Reinvent the wheel” mentality
- Typically have trouble dealing with abstractions – keep things concrete
- Still a relatively new research field
  - Approaches may vary.
  - No distracting elements
  - Emphasis on consistent navigation
  - High-contrast; large font sizes

Helpful Tools & Resources

- Development
  - Firefox Developer Toolbar (http://chrispederick.com/work/web-developer/)
- Testing
  - http://webxact.watchfire.com (Bobby)
  - http://www.webaim.org (WAVE tool)
- Section 508 of the Rehabilitation Act
  - http://www.section508.gov