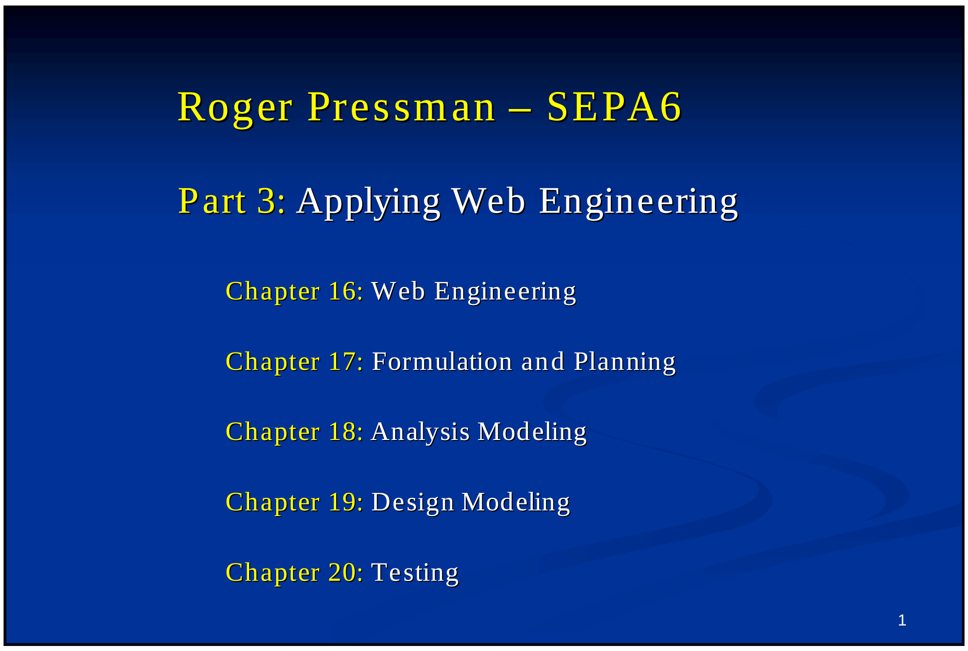


Roger Pressman – SEPA6

Part 3: Applying Web Engineering

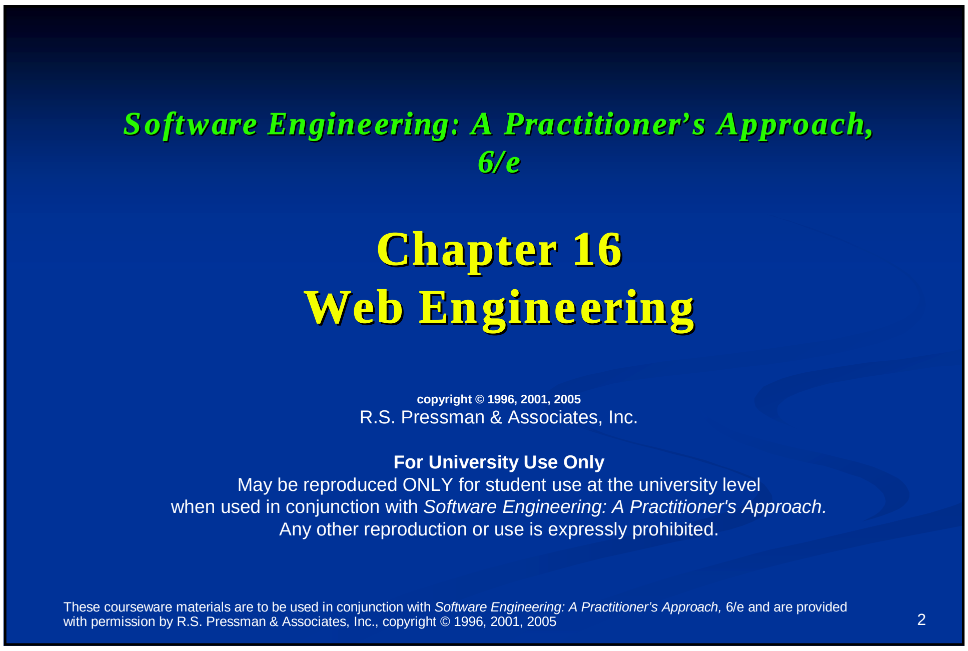
Chapter 16: Web Engineering

Chapter 17: Formulation and Planning

Chapter 18: Analysis Modeling

Chapter 19: Design Modeling

Chapter 20: Testing



Software Engineering: A Practitioner’s Approach, 6/e

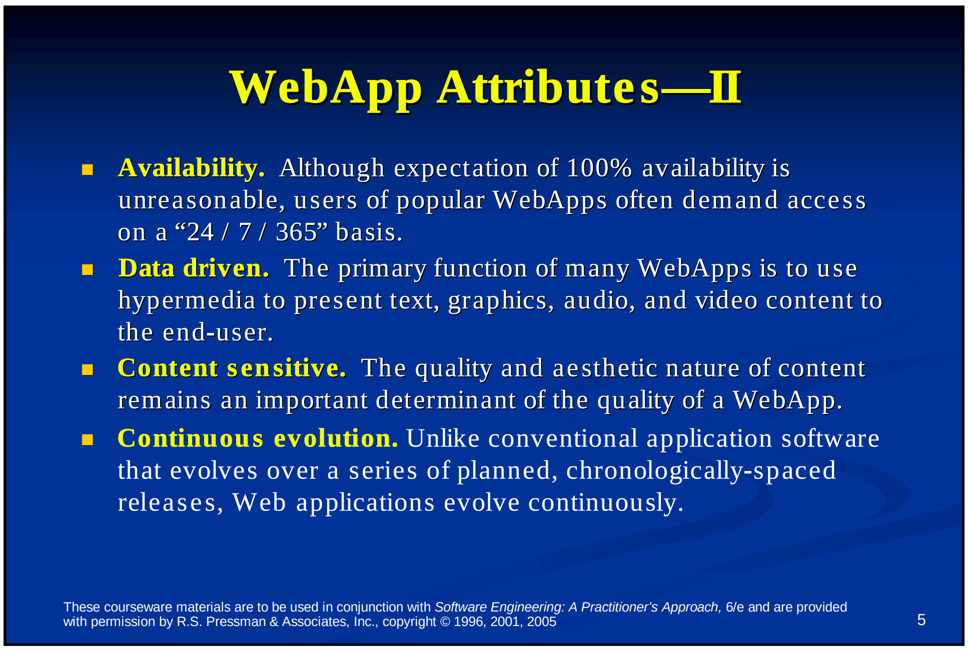
Chapter 16 Web Engineering

Web Applications

- WebApps encompass:
  - complete Web sites
    - Simple information Web sites
    - Complex e-Commerce sites with embedded functionality and data retrieval
    - Complex Web sites that are interoperable with other legacy software and systems
  - specialized functionality within Web sites
  - information processing applications that reside on the Internet or on an intranet

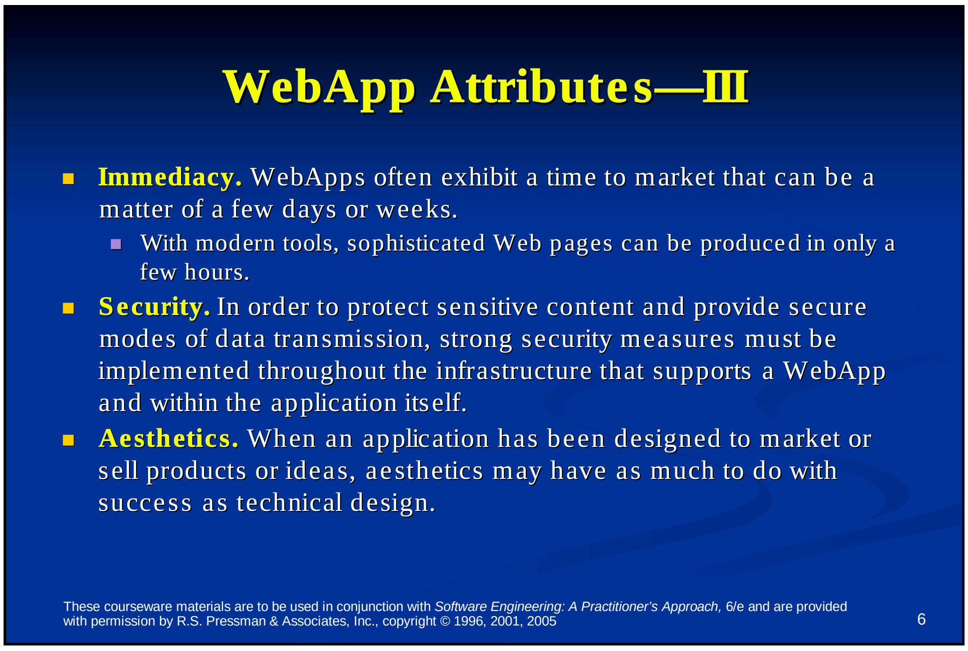
WebApp Attributes—I

- **Network intensiveness.** A WebApp resides on a network and must serve the needs of a diverse community of clients.
- **Concurrency.** A large number of users may access the WebApp at one time; patterns of usage among end-users will vary greatly.
- **Unpredictable load.** The number of users of the WebApp may vary by orders of magnitude from day to day.
- **Performance.** If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere.



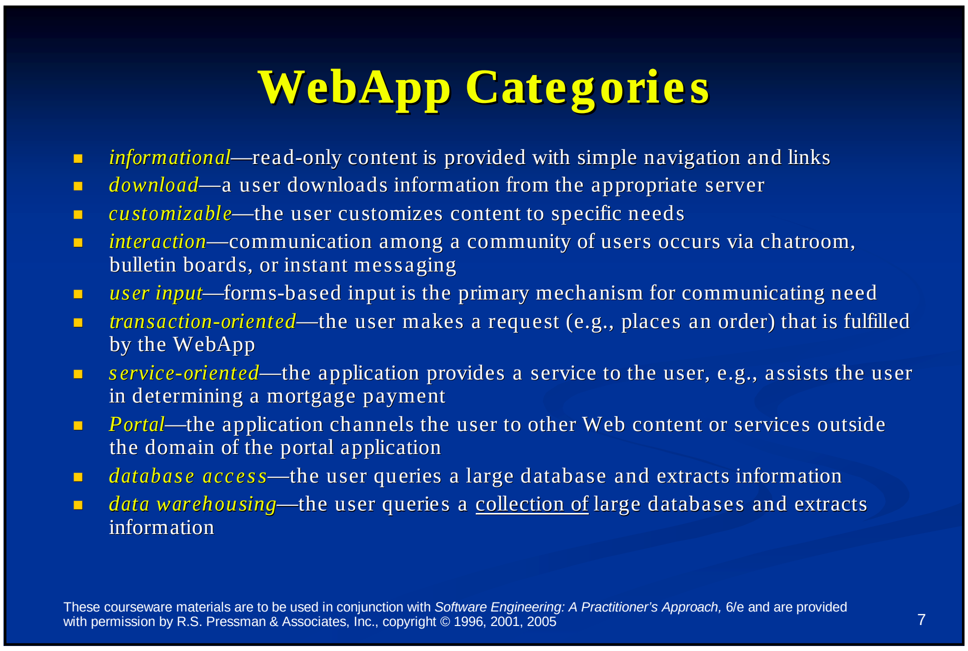
WebApp Attributes—II

- **Availability.** Although expectation of 100% availability is unreasonable, users of popular WebApps often demand access on a “24 / 7 / 365” basis.
- **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.
- **Content sensitive.** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.
- **Continuous evolution.** Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.



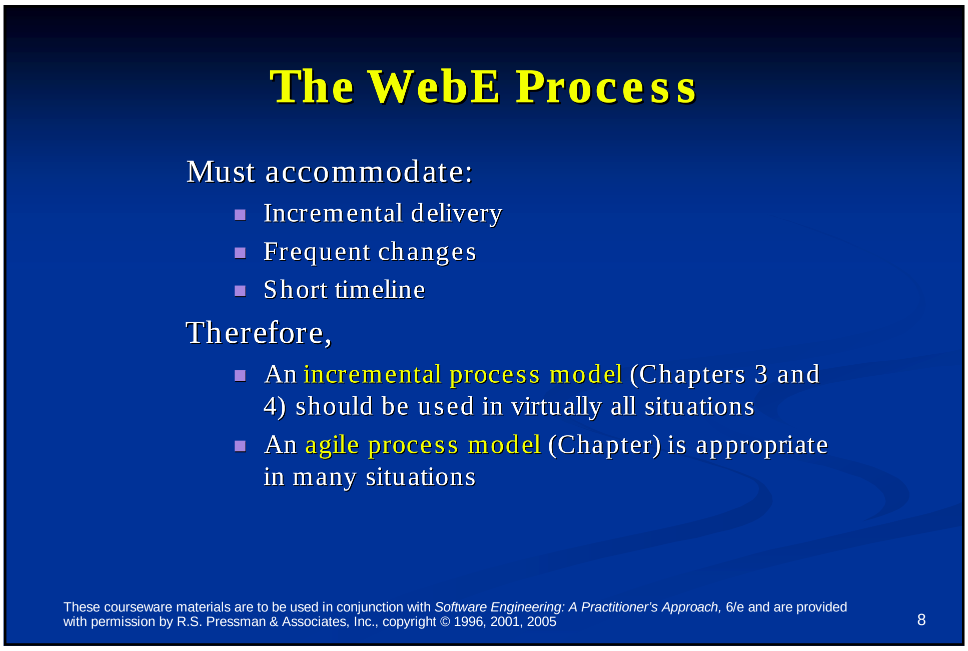
WebApp Attributes—III

- **Immediacy.** WebApps often exhibit a time to market that can be a matter of a few days or weeks.
  - With modern tools, sophisticated Web pages can be produced in only a few hours.
- **Security.** In order to protect sensitive content and provide secure modes of data transmission, strong security measures must be implemented throughout the infrastructure that supports a WebApp and within the application itself.
- **Aesthetics.** When an application has been designed to market or sell products or ideas, aesthetics may have as much to do with success as technical design.



WebApp Categories

- *informational*—read-only content is provided with simple navigation and links
- *download*—a user downloads information from the appropriate server
- *customizable*—the user customizes content to specific needs
- *interaction*—communication among a community of users occurs via chatroom, bulletin boards, or instant messaging
- *user input*—forms-based input is the primary mechanism for communicating need
- *transaction-oriented*—the user makes a request (e.g., places an order) that is fulfilled by the WebApp
- *service-oriented*—the application provides a service to the user, e.g., assists the user in determining a mortgage payment
- *Portal*—the application channels the user to other Web content or services outside the domain of the portal application
- *database access*—the user queries a large database and extracts information
- *data warehousing*—the user queries a collection of large databases and extracts information



The WebE Process

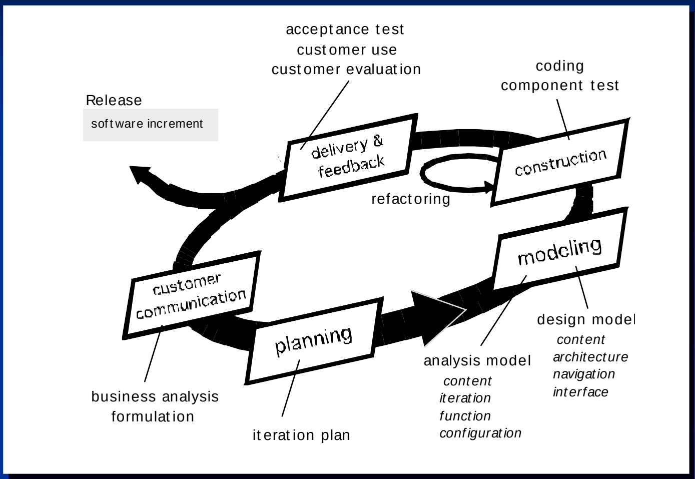
Must accommodate:

- Incremental delivery
- Frequent changes
- Short timeline

Therefore,

- An incremental process model (Chapters 3 and 4) should be used in virtually all situations
- An agile process model (Chapter) is appropriate in many situations

The WebE Process



The WebE Process Framework—I

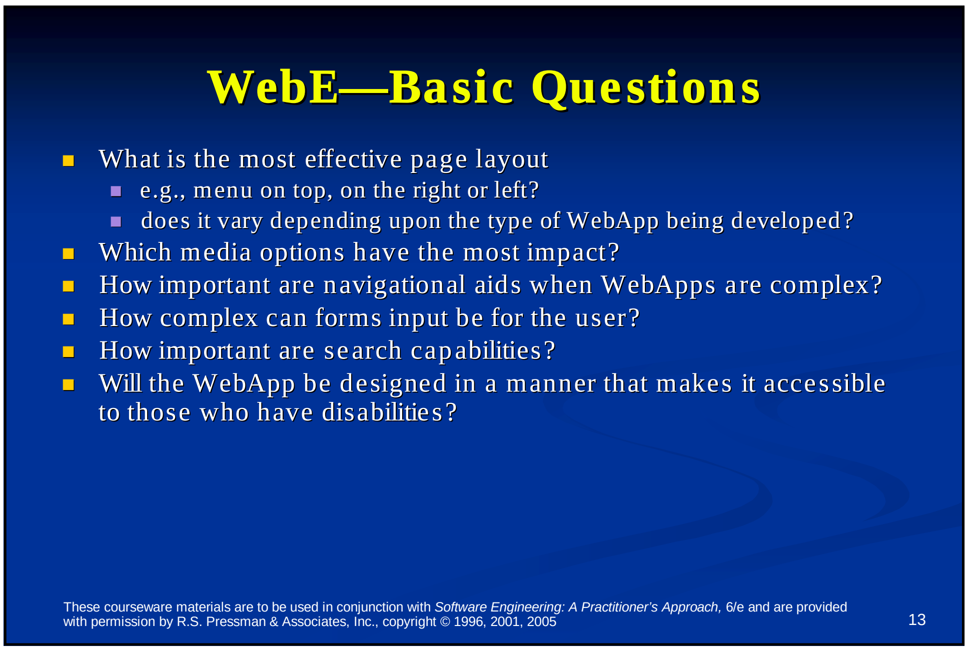
- Customer communication
  - *Business analysis* defines the business/organizational context for the WebApp.
  - *Formulation* is a requirements gathering activity involving all stakeholders. The intent is to describe the problem that the WebApp is to solve
- Planning
  - The "plan" consists of a task definition and a timeline schedule for the time period (usually measured in weeks) projected for the development of the WebApp increment.

The WebE Process Framework—II

- Modeling
  - Analysis model—establishes a basis for design
    - Content Analysis.
    - Interaction Analysis.
    - Functional Analysis.
    - Configuration Analysis.
  - Design model—represents key WebApp elements
    - Content design
    - Aesthetic design
    - Architectural design
    - Interface design
    - Navigation design
    - Component design

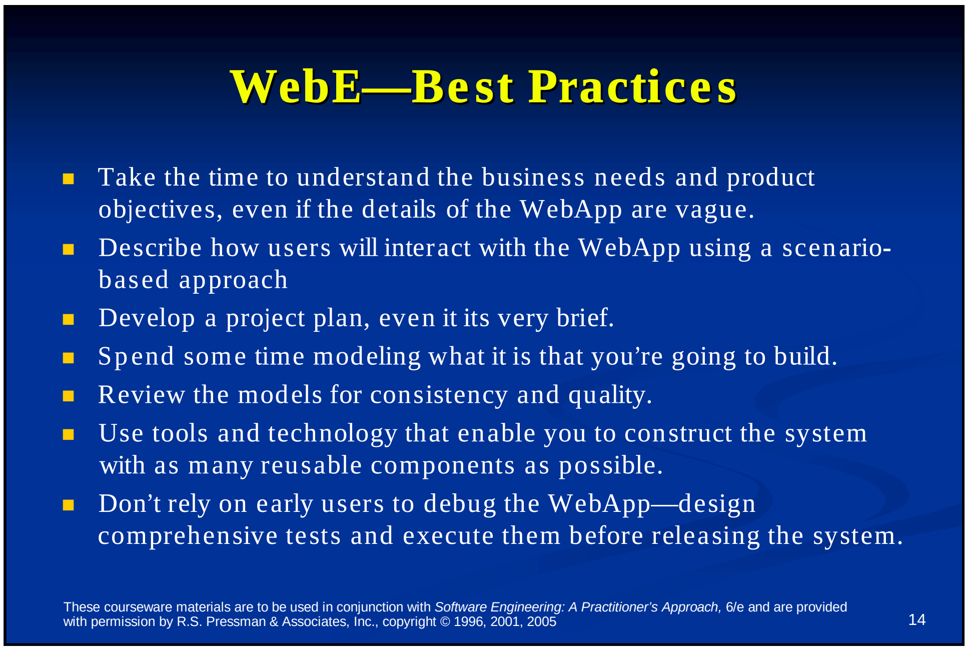
The WebE Process Framework—III

- Construction
  - WebE tools and technology (such as ASP.NET, etc.) are applied to construct the WebApp that has been modeled
  - Testing of all design elements
- Delivery and Evaluation (Deployment)
  - configure for its operational environment
  - deliver to end-users, and
  - Evaluation feedback is presented to the WebE team
  - the increment is modified as required (the beginning of the next incremental cycle)



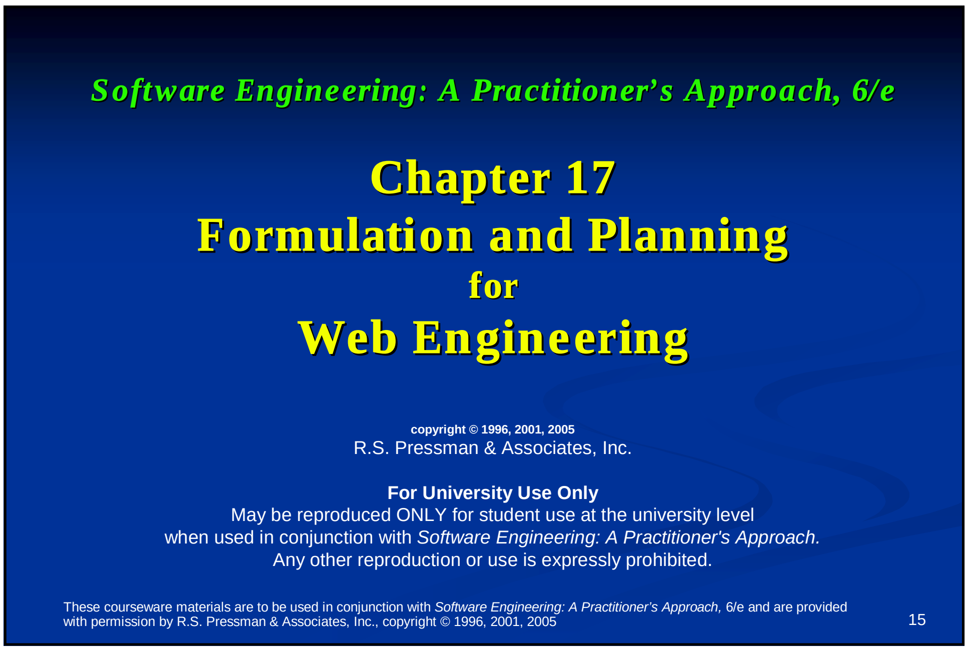
WebE—Basic Questions

- What is the most effective page layout
  - e.g., menu on top, on the right or left?
  - does it vary depending upon the type of WebApp being developed?
- Which media options have the most impact?
- How important are navigational aids when WebApps are complex?
- How complex can forms input be for the user?
- How important are search capabilities?
- Will the WebApp be designed in a manner that makes it accessible to those who have disabilities?



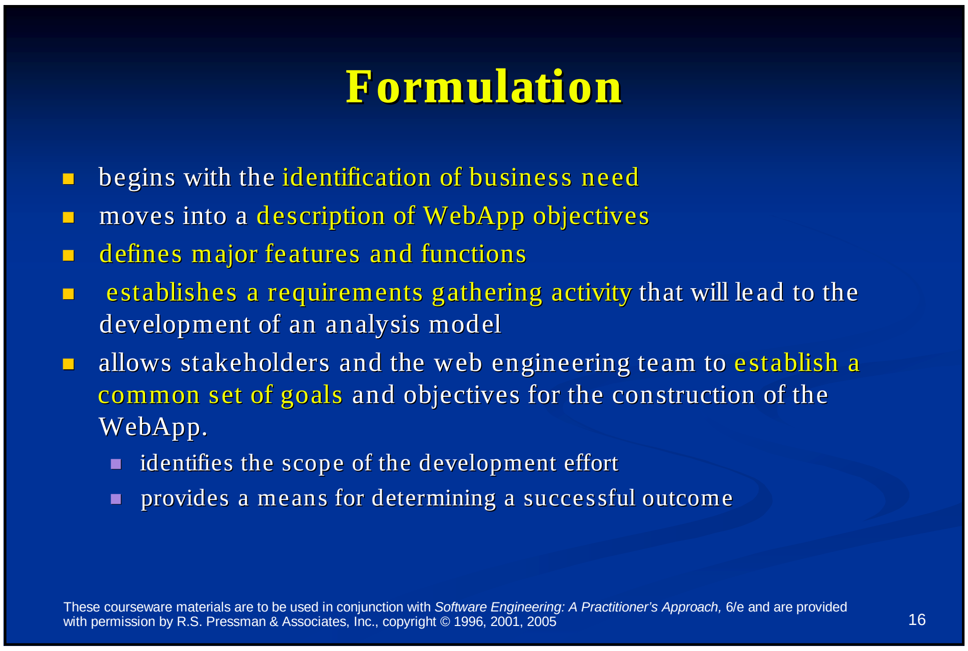
WebE—Best Practices

- Take the time to understand the business needs and product objectives, even if the details of the WebApp are vague.
- Describe how users will interact with the WebApp using a scenario-based approach
- Develop a project plan, even it its very brief.
- Spend some time modeling what it is that you're going to build.
- Review the models for consistency and quality.
- Use tools and technology that enable you to construct the system with as many reusable components as possible.
- Don't rely on early users to debug the WebApp—design comprehensive tests and execute them before releasing the system.



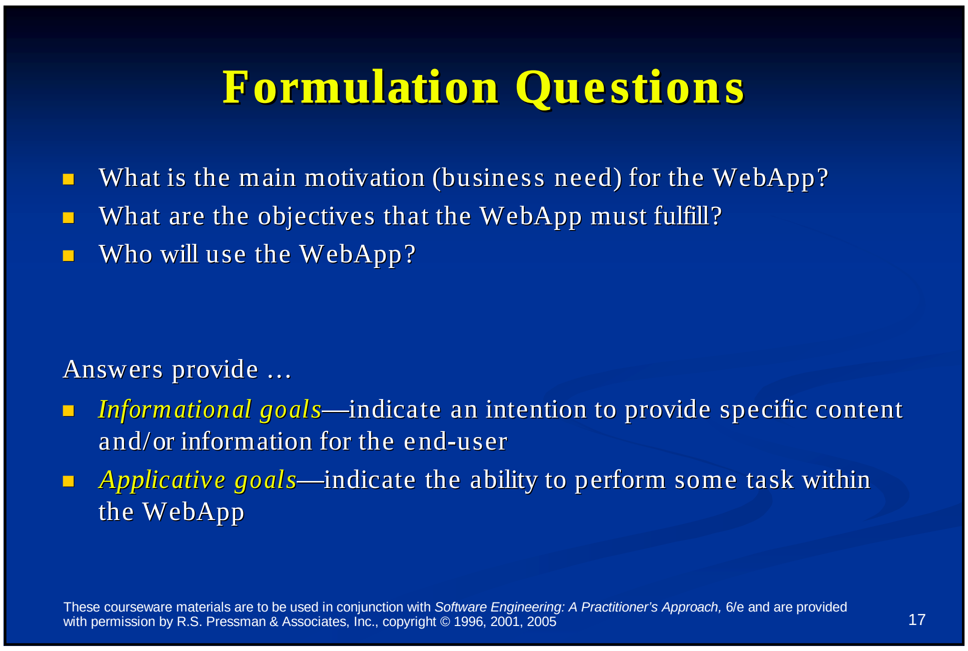
Software Engineering: A Practitioner's Approach, 6/e

Chapter 17 Formulation and Planning for Web Engineering



Formulation

- begins with the identification of business need
- moves into a description of WebApp objectives
- defines major features and functions
- establishes a requirements gathering activity that will lead to the development of an analysis model
- allows stakeholders and the web engineering team to establish a common set of goals and objectives for the construction of the WebApp.
  - identifies the scope of the development effort
  - provides a means for determining a successful outcome

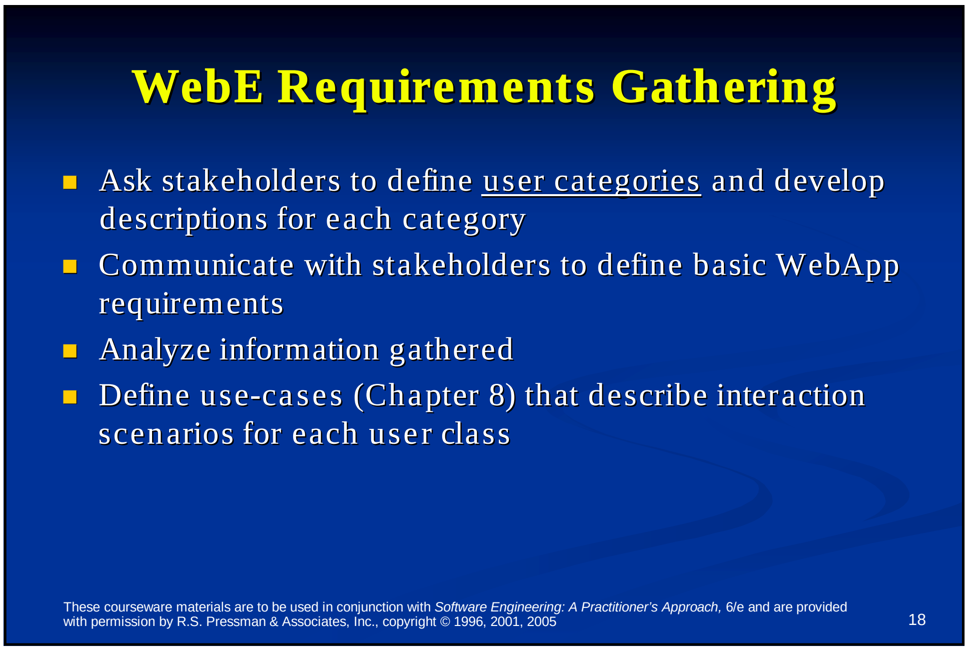


Formulation Questions

- What is the main motivation (business need) for the WebApp?
- What are the objectives that the WebApp must fulfill?
- Who will use the WebApp?

Answers provide …

- *Informational goals*—indicate an intention to provide specific content and/or information for the end-user
- *Applicative goals*—indicate the ability to perform some task within the WebApp



WebE Requirements Gathering

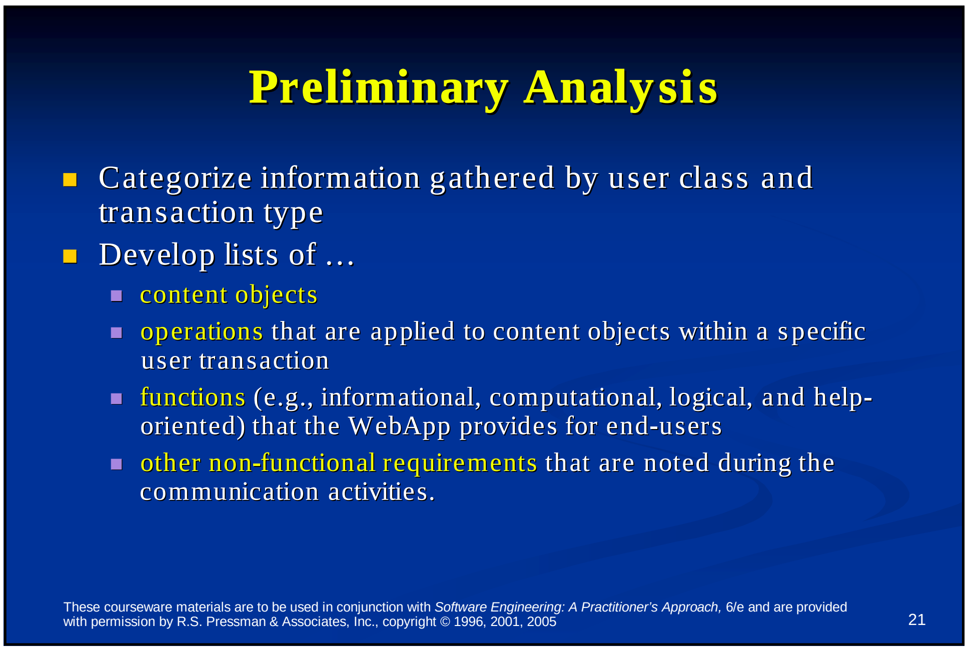
- Ask stakeholders to define user categories and develop descriptions for each category
- Communicate with stakeholders to define basic WebApp requirements
- Analyze information gathered
- Define use-cases (Chapter 8) that describe interaction scenarios for each user class

Defining User Categories

- What is the user's overall objective when using the WebApp?
- What is the user's background and sophistication relative to the content and functionality of the WebApp?
- How will the use arrive at the WebApp?
- What generic WebApp characteristics does the user like/dislike?

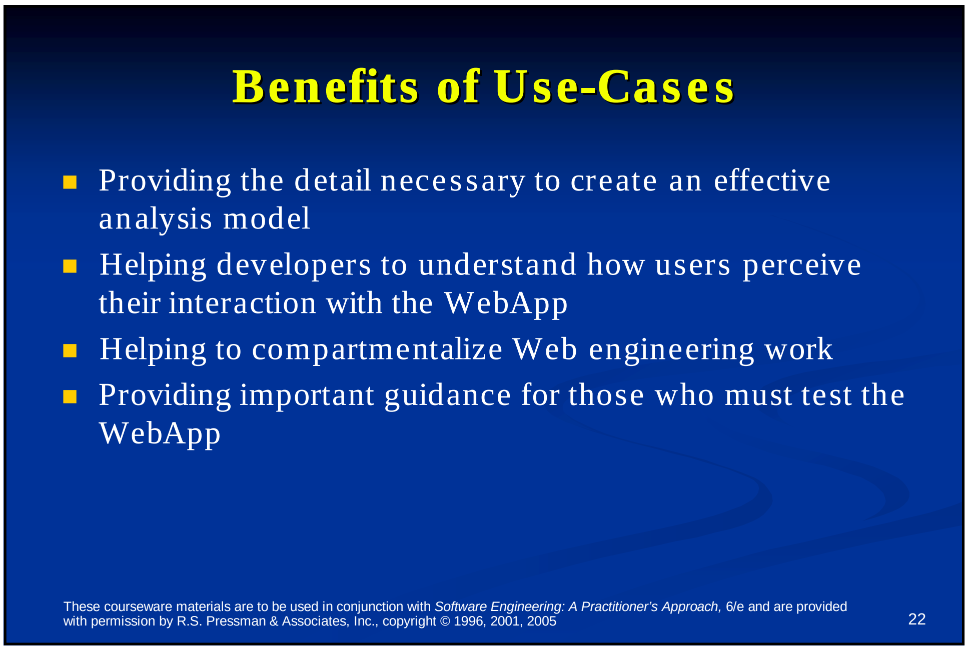
Communicating with Stakeholders

- *Traditional focus groups*—a trained moderator meets with a small group of representative end-users (or internal stakeholders playing the role of end-users).
- *Electronic focus groups*—a moderated electronic discussion conducted with a group of representative end-users and stakeholders.
- *Iterative surveys*—a series of brief surveys, addressed to representative users and requesting answers to specific questions about the WebApp
- *Exploratory surveys*—a Web-based survey that is tied to one or more WebApps that have users who are similar to the ones that will use the WebApp to be developed.
- *Scenario-building*—selected user are asked to create informal use-cases that describe specific interactions with the WebApp.



Preliminary Analysis

- Categorize information gathered by user class and transaction type
- Develop lists of …
  - content objects
  - operations that are applied to content objects within a specific user transaction
  - functions (e.g., informational, computational, logical, and help-oriented) that the WebApp provides for end-users
  - other non-functional requirements that are noted during the communication activities.



Benefits of Use-Cases

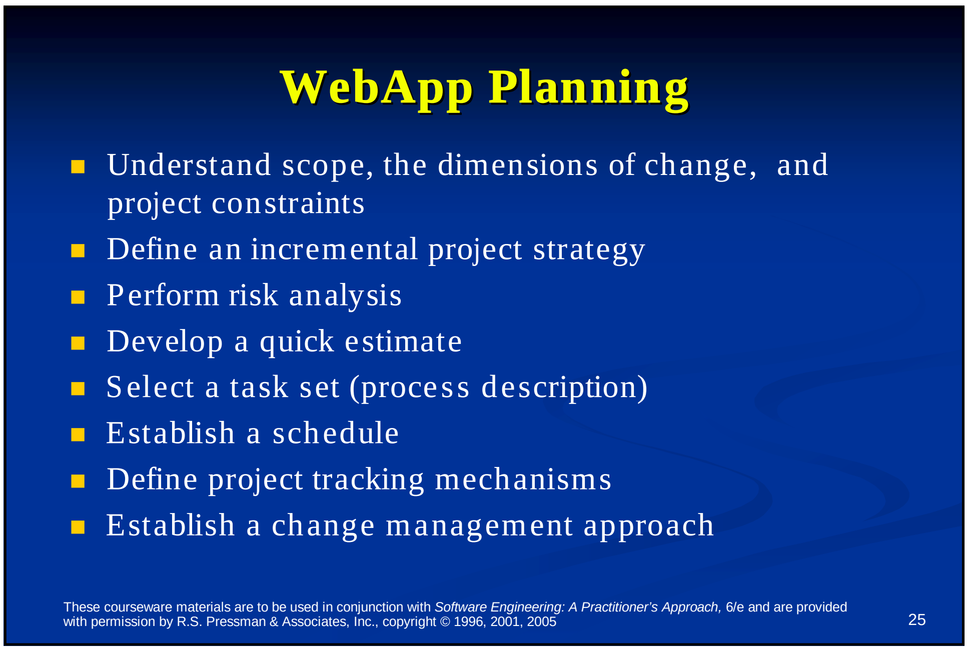
- Providing the detail necessary to create an effective analysis model
- Helping developers to understand how users perceive their interaction with the WebApp
- Helping to compartmentalize Web engineering work
- Providing important guidance for those who must test the WebApp

The WebE Team

- WebE team roles
  - Content Developer/Providers
  - Web Publisher (hosting)
  - Web Engineer
    - Architectural designer
    - Graphics designer
    - Programmer
  - Business domain experts
  - Support Specialist
  - Administrator (a.k.a. “Web Master”)

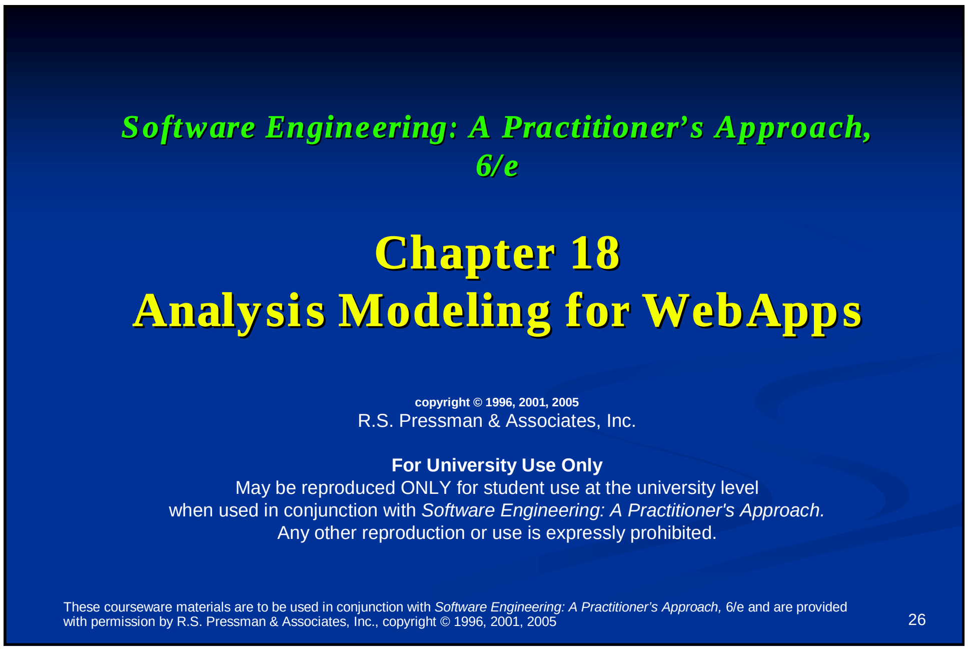
Project Differences

	Traditional Projects	**small e-Projects**	**major e-Projects**
Requirements Gathering	Rigorous	Limited	Rigorous
Technical Specifications	Robust: models, spec	Descriptive overview	robust: UML models, spec
Project Duration	Measured in months or years	Measured in days, weeks or months	Measured in months or years
Testing and QA	Focused on achieving quality tar-gets	Focused on risk control	SQAas described in Chapter 26
Risk Management	Explicit	Inherent	Explicit
Half-life of deliverables	18 months or longer	3 to 6 months or shorter	6 to 12 months or shorter
Release Process	Rigorous	Expedited	Rigorous
Post-release customer feedback	Requires proactive effort	Automatically obtained from user interaction	Obtained both auto-matically and via solicited feedback



WebApp Planning

- Understand scope, the dimensions of change, and project constraints
- Define an incremental project strategy
- Perform risk analysis
- Develop a quick estimate
- Select a task set (process description)
- Establish a schedule
- Define project tracking mechanisms
- Establish a change management approach



Software Engineering: A Practitioner's Approach, 6/e

Chapter 18
Analysis Modeling for WebApps

Analysis

Content Analysis. The full spectrum of content to be provided by the WebApp is identified, including text, graphics and images, video, and audio data. Data modeling can be used to identify and describe each of the data objects.

Interaction Analysis. The manner in which the user interacts with the WebApp is described in detail. Use-cases can be developed to provide detailed descriptions of this interaction.

Functional Analysis. The usage scenarios (use-cases) created as part of interaction analysis define the operations that will be applied to WebApp content and imply other processing functions. All operations and functions are described in detail.

Configuration Analysis. The environment and infrastructure (ie. Web server, database server, etc.) in which the WebApp resides are described in detail.

When Do We Perform Analysis?

- In some WebE situations, analysis and design merge. However, an explicit analysis activity occurs when …
  - the WebApp to be built is large and/or complex
  - the number of stakeholders is large
  - the number of Web engineers and other contributors is large
  - the goals and objectives (determined during formulation) for the WebApp will effect the business' bottom line
  - the success of the WebApp will have a strong bearing on the success of the business

The User Hierarchy

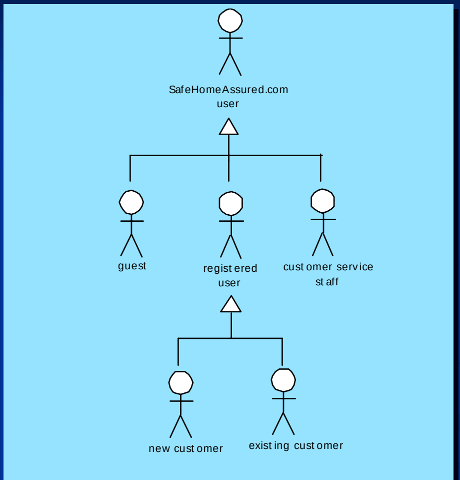


Figure 18.1 User hierarchy for SafeHomeAssured.com

Use-Case Diagram

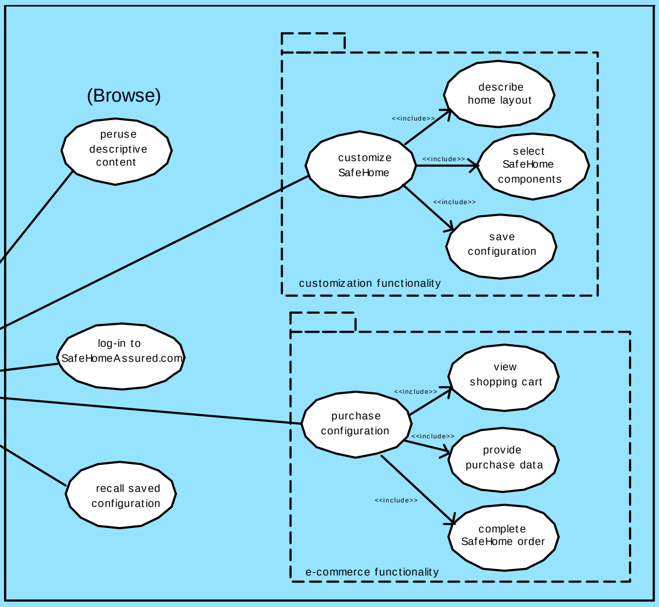
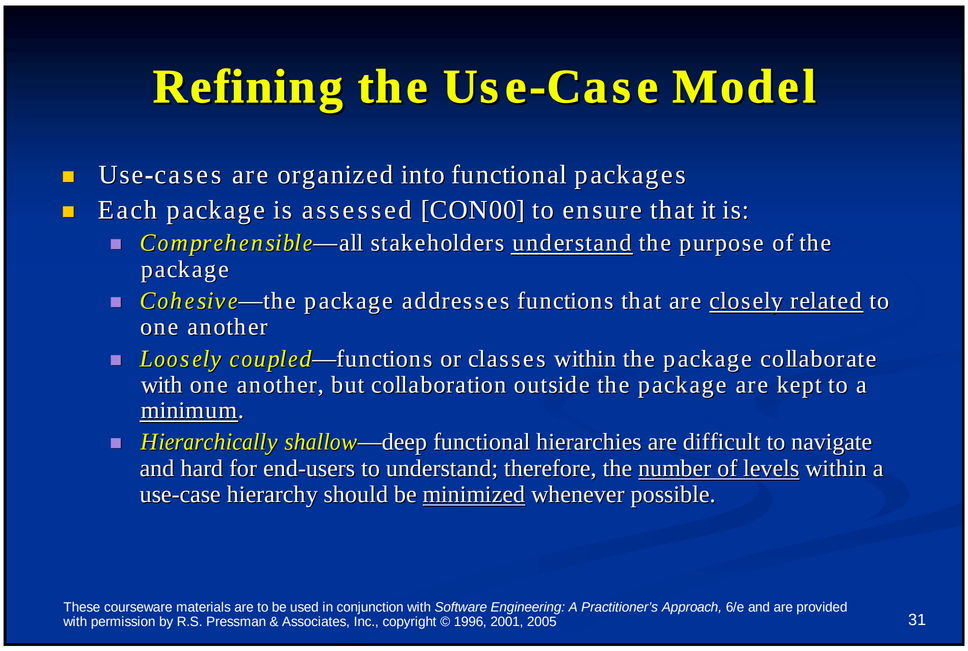
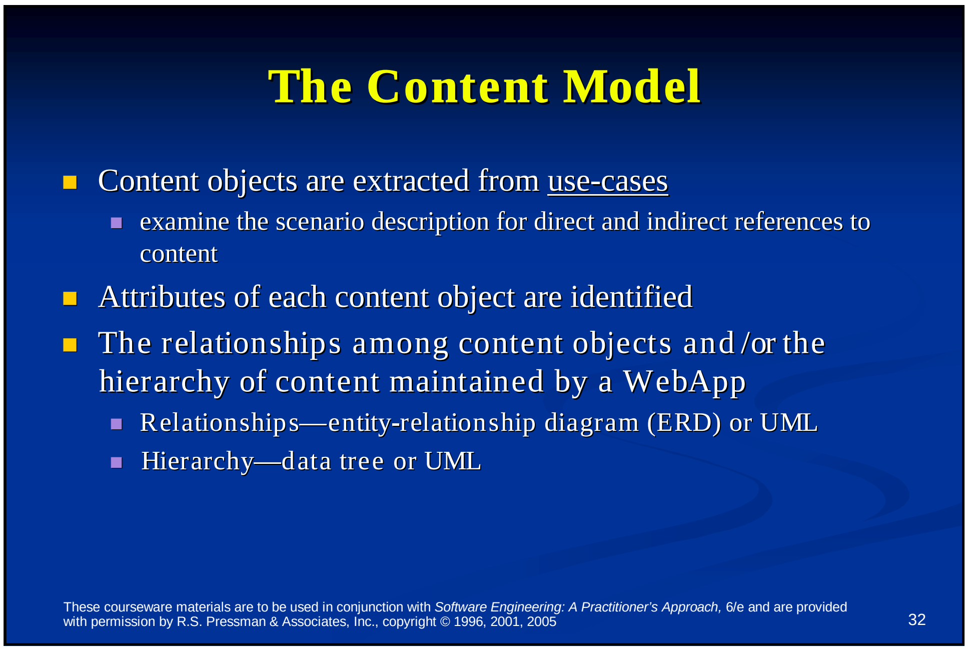


Figure 18.2 Use-case diagram for new-customer



Refining the Use-Case Model

- Use-cases are organized into functional packages
- Each package is assessed [CON00] to ensure that it is:
  - *Comprehensible*—all stakeholders understand the purpose of the package
  - *Cohesive*—the package addresses functions that are closely related to one another
  - *Loosely coupled*—functions or classes within the package collaborate with one another, but collaboration outside the package are kept to a minimum.
  - *Hierarchically shallow*—deep functional hierarchies are difficult to navigate and hard for end-users to understand; therefore, the number of levels within a use-case hierarchy should be minimized whenever possible.



The Content Model

- Content objects are extracted from use-cases
  - examine the scenario description for direct and indirect references to content
- Attributes of each content object are identified
- The relationships among content objects and/or the hierarchy of content maintained by a WebApp
  - Relationships—entity-relationship diagram (ERD) or UML
  - Hierarchy—data tree or UML

Data Tree

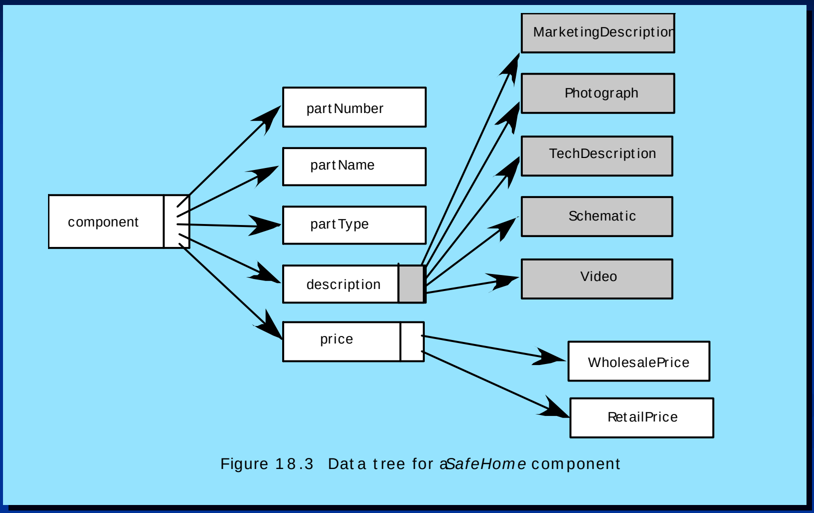


Figure 18.3 Data tree for a *SafeHome* component

Analysis Classes

- Analysis classes are derived by examining each use-case
- A grammatical parse is used to identify candidate classes
- A UML class diagram is developed for each analysis class

Analysis Class Example

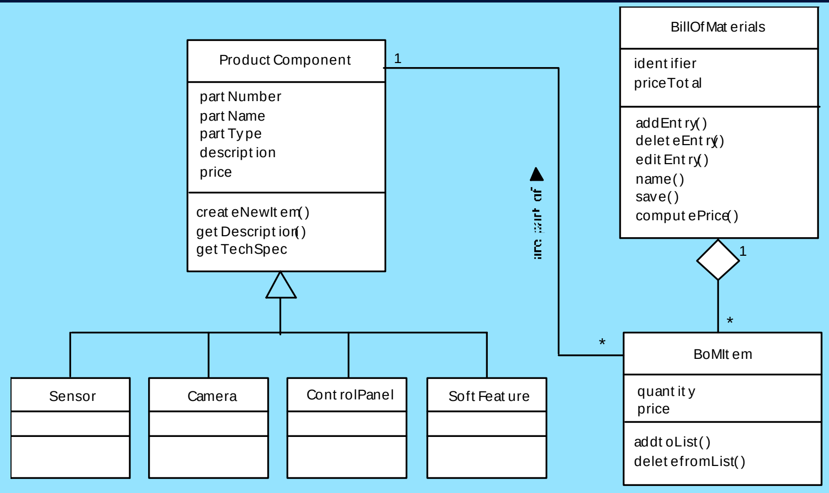


Figure 18.4 Analysis classes for use-case: *select SafeHome components*

The Interaction Model

- Composed of four elements:
  - use-cases
  - sequence diagrams
  - state diagrams
  - a user interface prototype
- Each of these is an important UML notation and has been described in Chapter 8

Sequence Diagram

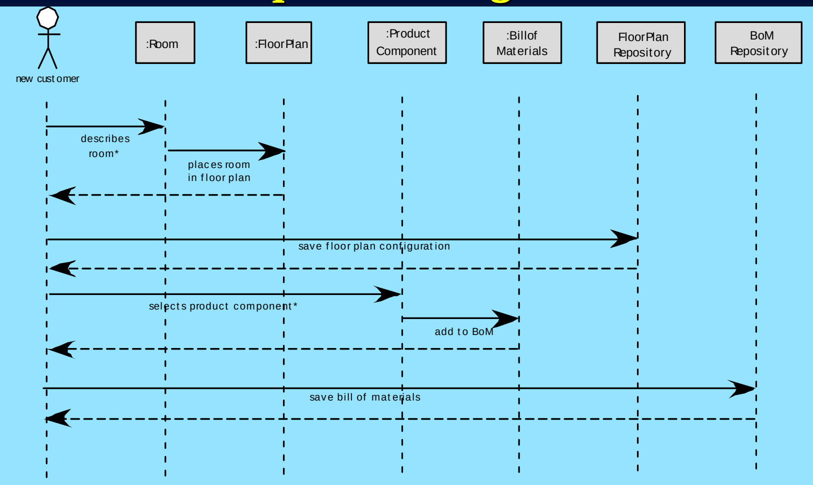
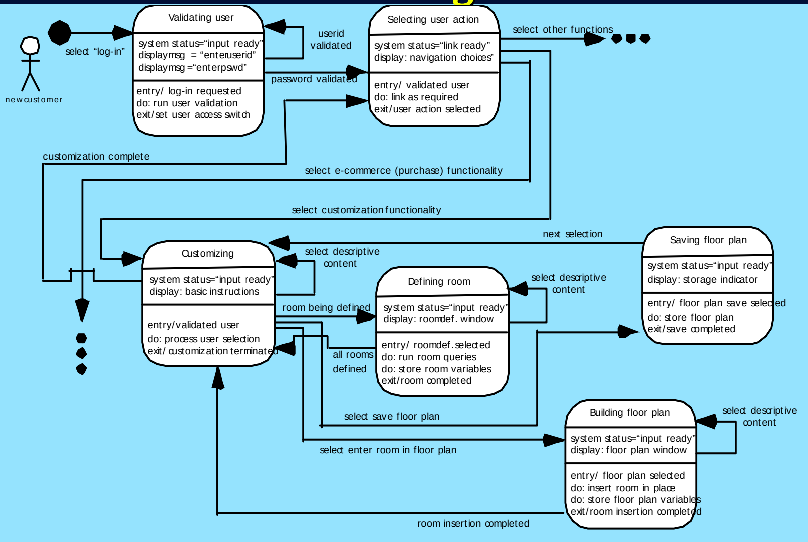


Figure 18.5 Sequence diagram for use-case: *select SafeHome components*

State Diagram



new customer
select "log-in"
Validating user
system status="input ready"
displaymsg = "enteruserid"
displaymsg ="enterpswd"
entry/ log-in requested
do: run user validation
exit/set user access switch
userid validated
password validated
Selecting user action
system status="link ready"
display: navigation choices"
entry/ validated user
do: link as required
exit/user action selected
select other functions
customization complete
select e-commerce (purchase) functionality
select customization functionality
next selection
Saving floor plan
system status="input ready"
display: storage indicator
entry/ floor plan save selected
do: store floor plan
exit/save completed
Customizing
system status="input ready"
display: basic instructions
entry/validated user
do: process user selection
exit/ customization terminated
select descriptive content
room being defined
Defining room
system status="input ready"
display: roomdef. window
entry/ roomdef. selected
do: run room queries
do: store room variables
exit/room completed
select descriptive content
all rooms defined
select save floor plan
select enter room in floor plan
Building floor plan
system status="input ready"
display: floor plan window
entry/ floor plan selected
do: insert room in place
do: store floor plan variables
exit/room insertion completed
select descriptive content
room insertion completed

Figure 18.6 Partial state diagram for **new customer** interaction

The Functional Model

- The functional model addresses two processing elements of the WebApp
  - user observable functionality that is delivered by the WebApp to end-users
  - the operations contained within analysis classes that implement behaviors associated with the class.
- An activity diagram can be used to represent processing flow

Activity Diagram

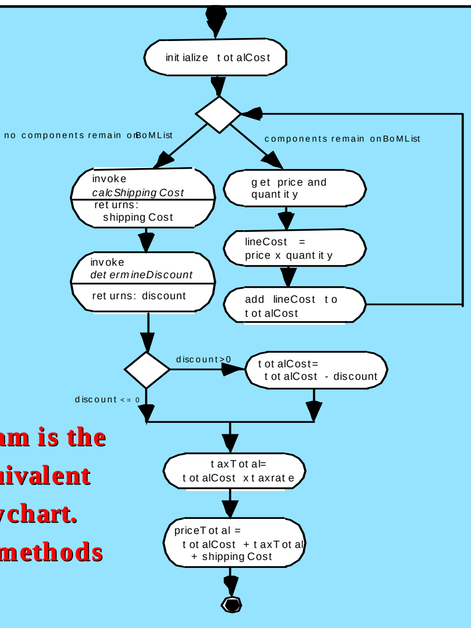

init ialize t ot alCost
no components remain onBoMList
components remain onBoMList
invoke *calcShipping Cost*
ret urns: shipping Cost
invoke *det ermineDiscount*
ret urns: discount
g et price and quant it y
lineCost = price x quant it y
add lineCost t o t ot alCost
discount>0
discount <= 0
t ot alCost= t ot alCost - discount
t axT ot al= t ot alCost x t axrat e
priceT ot al = t ot alCost + t axT ot al + shipping Cost

Figure 1 8 .7 Activity diagram for ***computePrice()* operatio**

The Activity diagram is the UML extended equivalent of the classic Flowchart. (Shows details of methods in a class.)

The Configuration Model

- Server-side
  - Server hardware and operating system environment must be specified
  - Interoperability considerations on the server-side must be considered
  - Appropriate interfaces, communication protocols and related collaborative information must be specified
- Client-side
  - Browser configuration issues must be identified
  - Testing requirements should be defined

Relationship-Navigation Analysis (RNA)

- Relationship-navigation analysis (RNA) identifies relationships among the elements uncovered as part of the creation of the analysis model
- Steps:
  - *Stakeholder analysis*—identifies the various user categories and establishes an appropriate stakeholder hierarchy
  - *Element analysis*—identifies the content objects and functional elements that are of interest to end users
  - *Relationship analysis*—describes the relationships that exist among the WebApp elements
  - *Navigation analysis*—examines how users might access individual elements or groups of elements
  - *Evaluation analysis*—considers pragmatic issues (e.g., cost/benefit) associated with implementing the relationships defined earlier

Navigation Analysis-I

- Should certain elements be easier to reach (require fewer navigation steps) than others? What is the priority for presentation?
- Should certain elements be emphasized to force users to navigate in their direction?
- How should navigation errors be handled?
- Should navigation to related groups of elements be given priority over navigation to a specific element.
- Should navigation be accomplished via links (portal), via search-based access, or by some other means?
- Should certain elements be presented to users based on the context of previous navigation actions?
- Should a navigation log be maintained for users?
  - (visitor counters, statistics, etc.)

Navigation Analysis-II

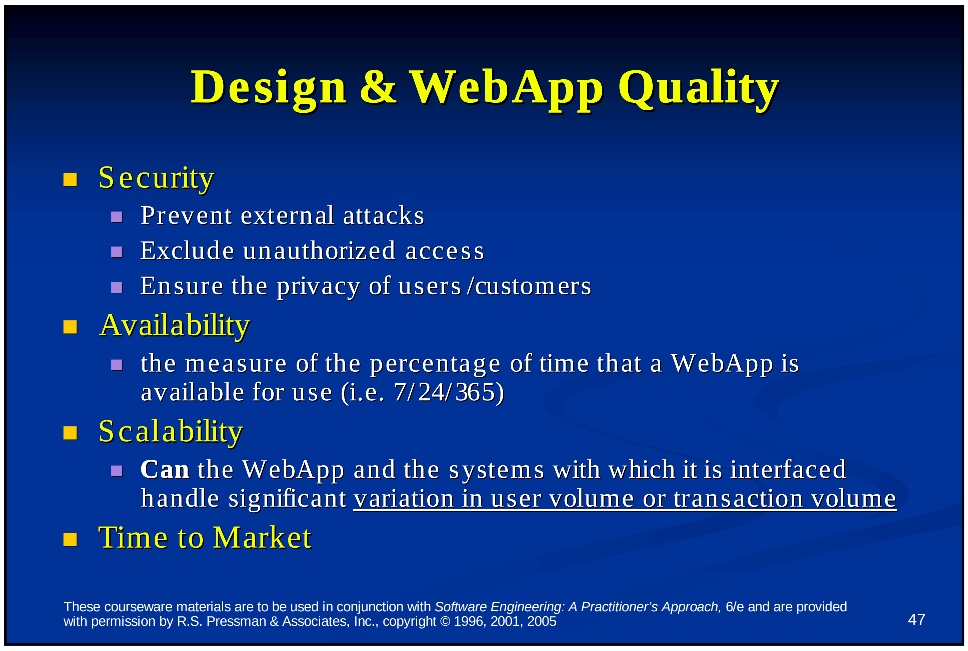
- Should a full navigation map or menu be available at every point in a user's interaction?
- Should navigation design be driven by the most commonly expected user behaviors or by the perceived importance of the defined WebApp elements?
- Can a user "store" his previous navigation through the WebApp to expedite future usage?
- For which user category should optimal navigation be designed?
- How should links external to the WebApp be handled?
  - overlaying the existing browser window?
  - as a new browser window?
  - as a separate frame?

Software Engineering: A Practitioner's Approach, 6/e

Chapter 19
Design Modeling for WebApps

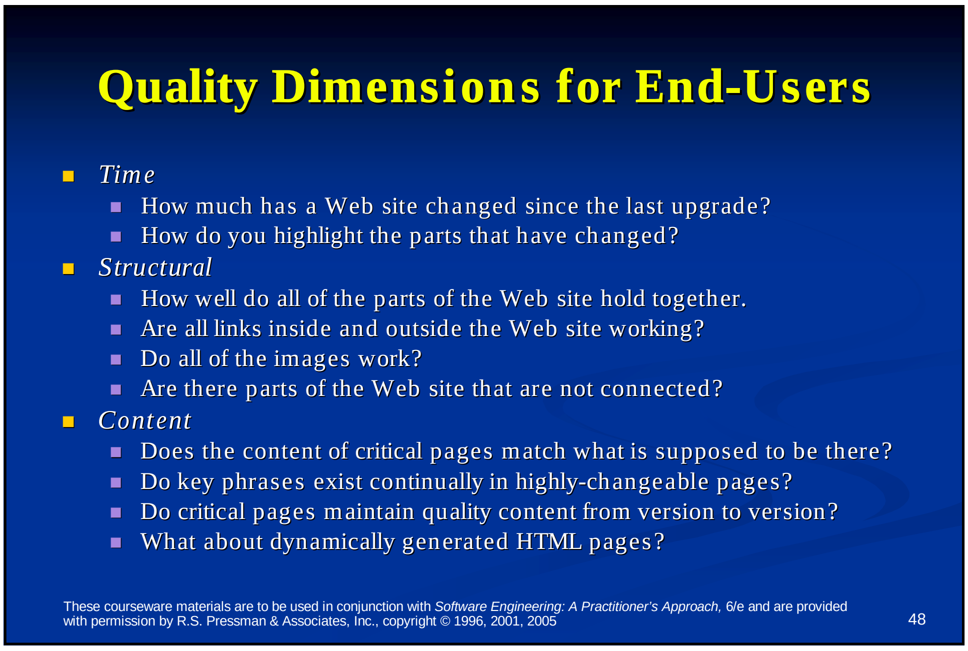
Design & WebApps

- When should we emphasize WebApp design?
  - when content and function are complex
  - when the size of the WebApp encompasses hundreds of content objects, functions, and analysis classes
  - when the success of the WebApp will have a direct impact on the success of the business



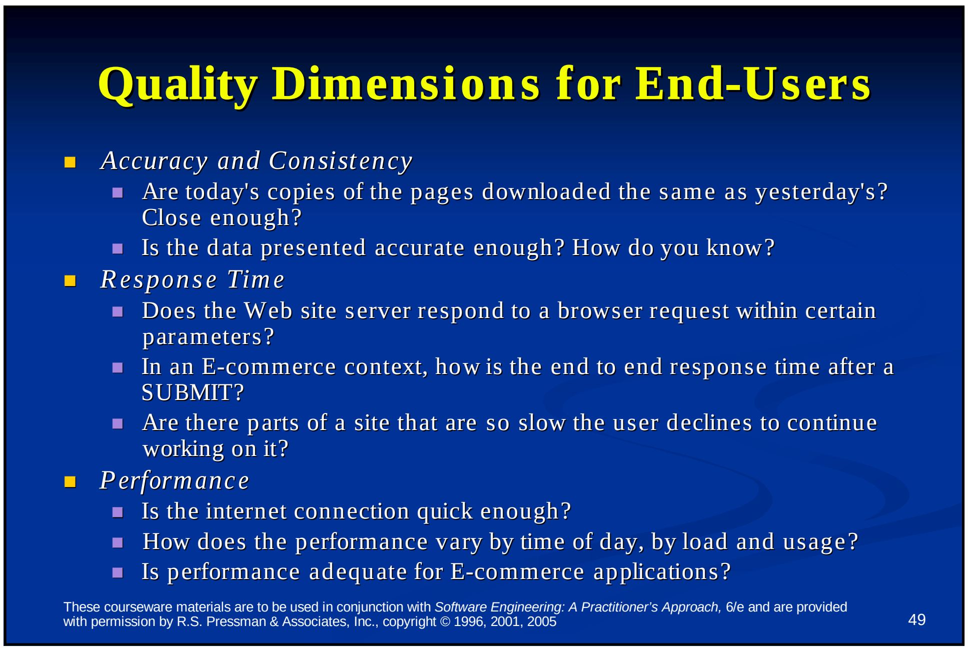
Design & WebApp Quality

- Security
  - Prevent external attacks
  - Exclude unauthorized access
  - Ensure the privacy of users /customers
- Availability
  - the measure of the percentage of time that a WebApp is available for use (i.e. 7/24/365)
- Scalability
  - **Can** the WebApp and the systems with which it is interfaced handle significant variation in user volume or transaction volume
- Time to Market



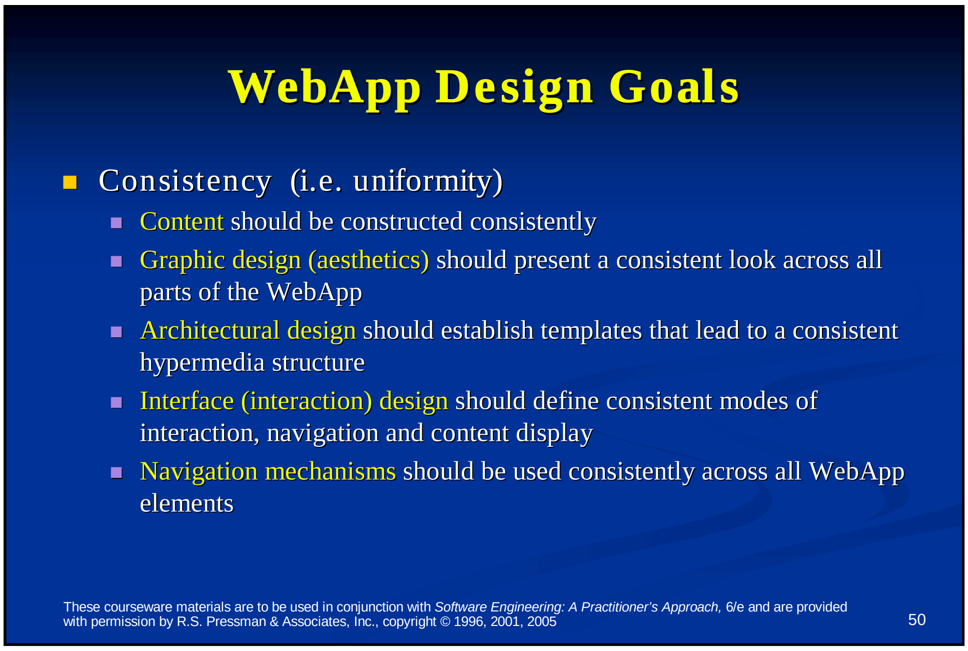
Quality Dimensions for End-Users

- *Time*
  - How much has a Web site changed since the last upgrade?
  - How do you highlight the parts that have changed?
- *Structural*
  - How well do all of the parts of the Web site hold together.
  - Are all links inside and outside the Web site working?
  - Do all of the images work?
  - Are there parts of the Web site that are not connected?
- *Content*
  - Does the content of critical pages match what is supposed to be there?
  - Do key phrases exist continually in highly-changeable pages?
  - Do critical pages maintain quality content from version to version?
  - What about dynamically generated HTML pages?



Quality Dimensions for End-Users

- *Accuracy and Consistency*
  - Are today's copies of the pages downloaded the same as yesterday's? Close enough?
  - Is the data presented accurate enough? How do you know?
- *Response Time*
  - Does the Web site server respond to a browser request within certain parameters?
  - In an E-commerce context, how is the end to end response time after a SUBMIT?
  - Are there parts of a site that are so slow the user declines to continue working on it?
- *Performance*
  - Is the internet connection quick enough?
  - How does the performance vary by time of day, by load and usage?
  - Is performance adequate for E-commerce applications?



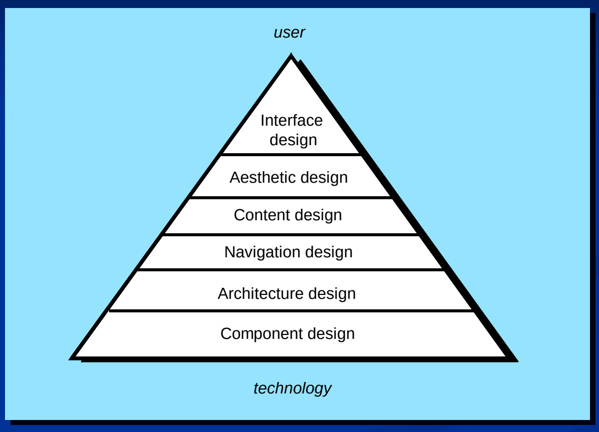
WebApp Design Goals

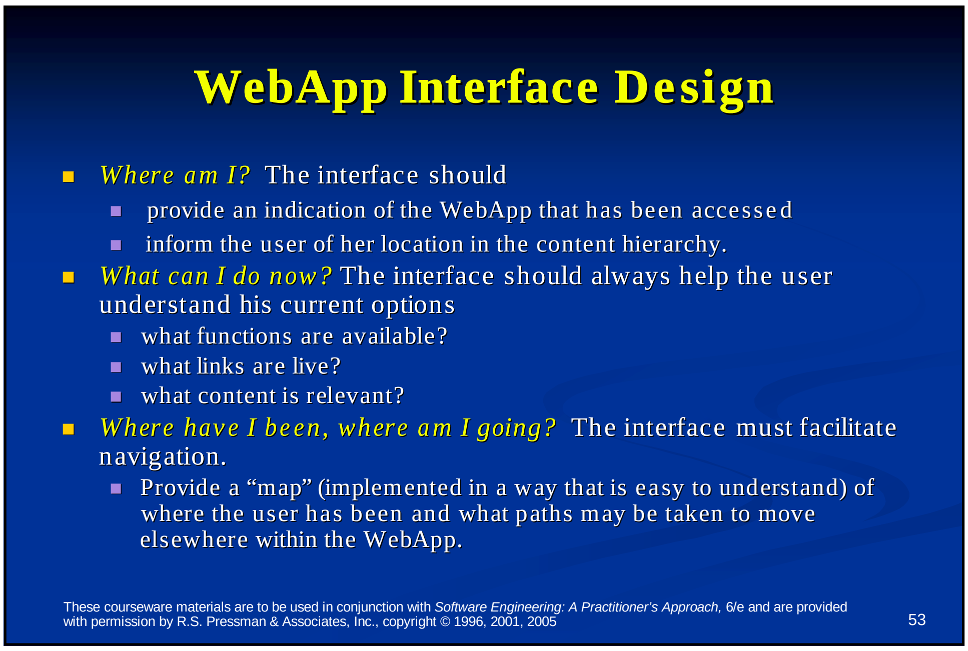
- Consistency (i.e. uniformity)
  - Content should be constructed consistently
  - Graphic design (aesthetics) should present a consistent look across all parts of the WebApp
  - Architectural design should establish templates that lead to a consistent hypermedia structure
  - Interface (interaction) design should define consistent modes of interaction, navigation and content display
  - Navigation mechanisms should be used consistently across all WebApp elements

WebApp Design Goals

- Robustness
  - The user expects robust content and functions that are relevant to the user’s needs
- Navigability
  - designed in a manner that is intuitive and predictable
- Visual appeal
  - the look and feel of content, interface layout, color coordination, the balance of text, graphics and other media, navigation mechanisms must appeal to end-users
- Compatibility
  - With all appropriate environments and configurations

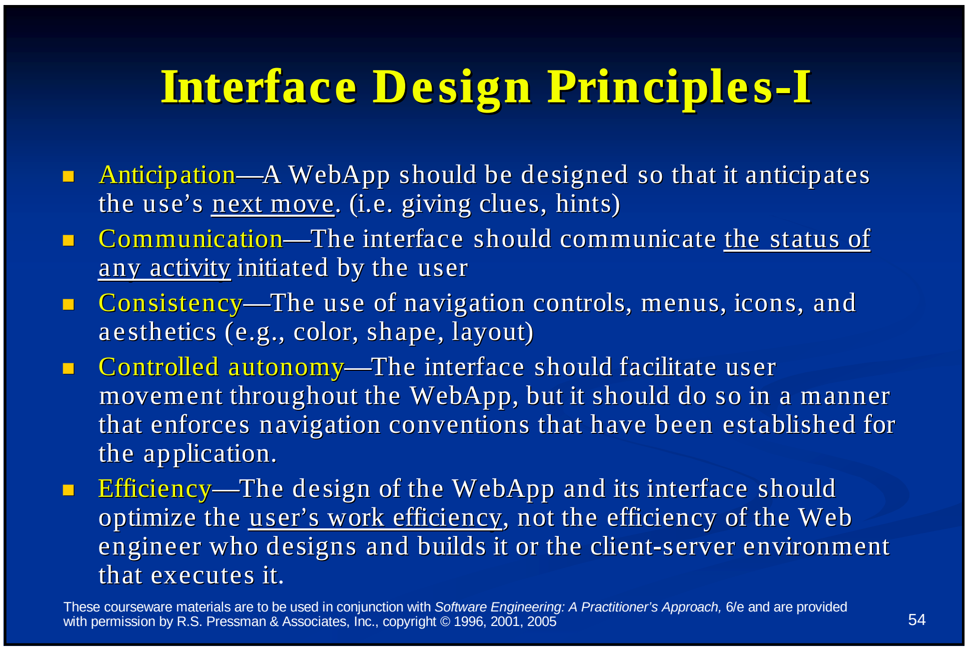
WebE Design Pyramid





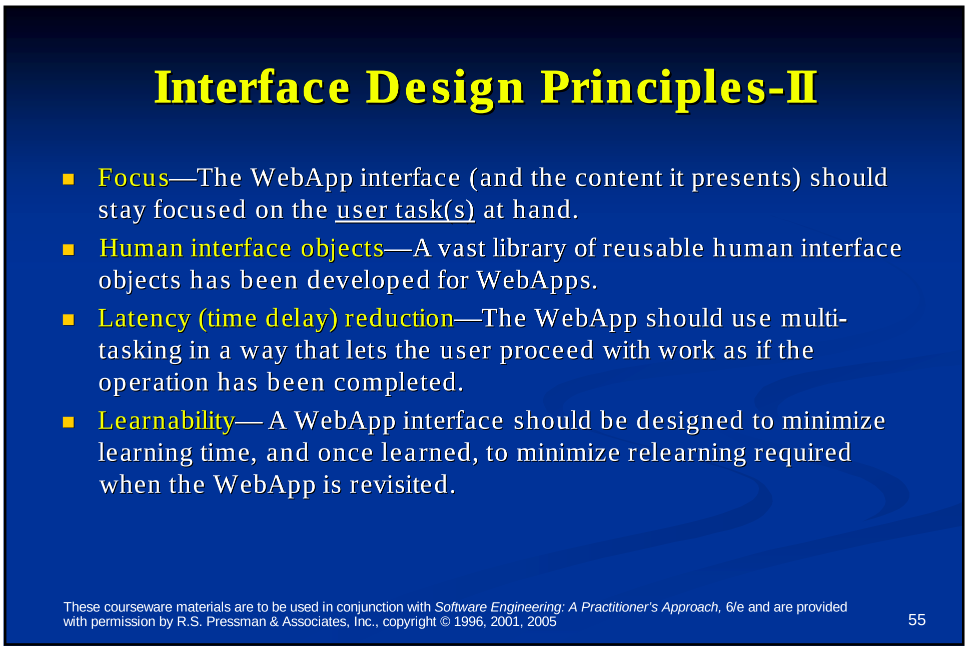
WebApp Interface Design

- *Where am I?* The interface should
  - provide an indication of the WebApp that has been accessed
  - inform the user of her location in the content hierarchy.
- *What can I do now?* The interface should always help the user understand his current options
  - what functions are available?
  - what links are live?
  - what content is relevant?
- *Where have I been, where am I going?* The interface must facilitate navigation.
  - Provide a "map" (implemented in a way that is easy to understand) of where the user has been and what paths may be taken to move elsewhere within the WebApp.



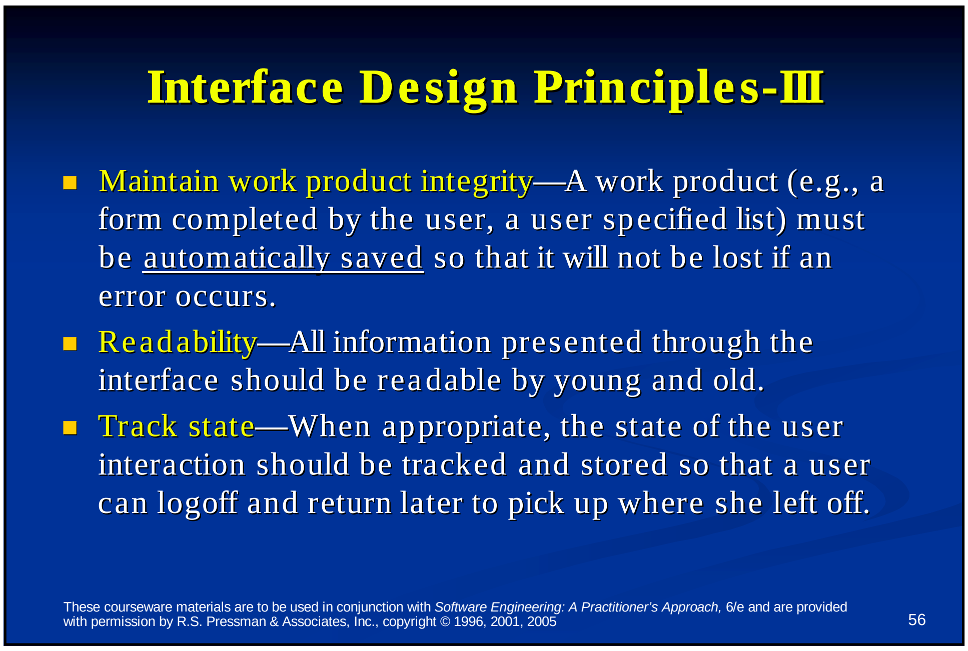
Interface Design Principles-I

- Anticipation—A WebApp should be designed so that it anticipates the use's <u>next move</u>. (i.e. giving clues, hints)
- Communication—The interface should communicate <u>the status of any activity</u> initiated by the user
- Consistency—The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)
- Controlled autonomy—The interface should facilitate user movement throughout the WebApp, but it should do so in a manner that enforces navigation conventions that have been established for the application.
- Efficiency—The design of the WebApp and its interface should optimize the <u>user's work efficiency</u>, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.



Interface Design Principles-II

- Focus—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- Human interface objects—A vast library of reusable human interface objects has been developed for WebApps.
- Latency (time delay) reduction—The WebApp should use multi-tasking in a way that lets the user proceed with work as if the operation has been completed.
- Learnability— A WebApp interface should be designed to minimize learning time, and once learned, to minimize relearning required when the WebApp is revisited.



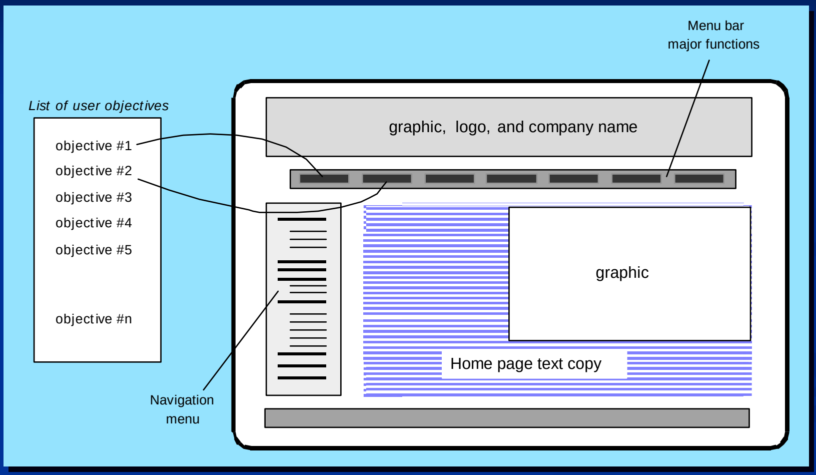
Interface Design Principles-III

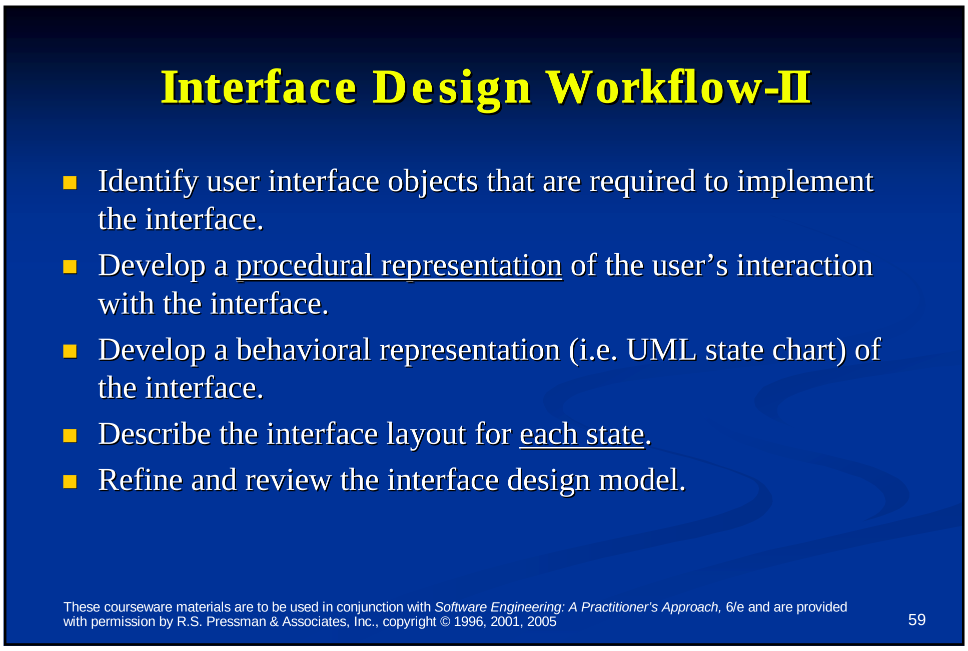
- Maintain work product integrity—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- Readability—All information presented through the interface should be readable by young and old.
- Track state—When 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.

Interface Design Workflow-I

- Review information contained in the analysis model and refine as required.
- Develop a rough sketch of the WebApp interface layout.
- Map user objectives into specific interface actions.
- Define a set of user tasks that are associated with each action.
- Storyboard (i.e. display in sequence) the screen images for each interface action.

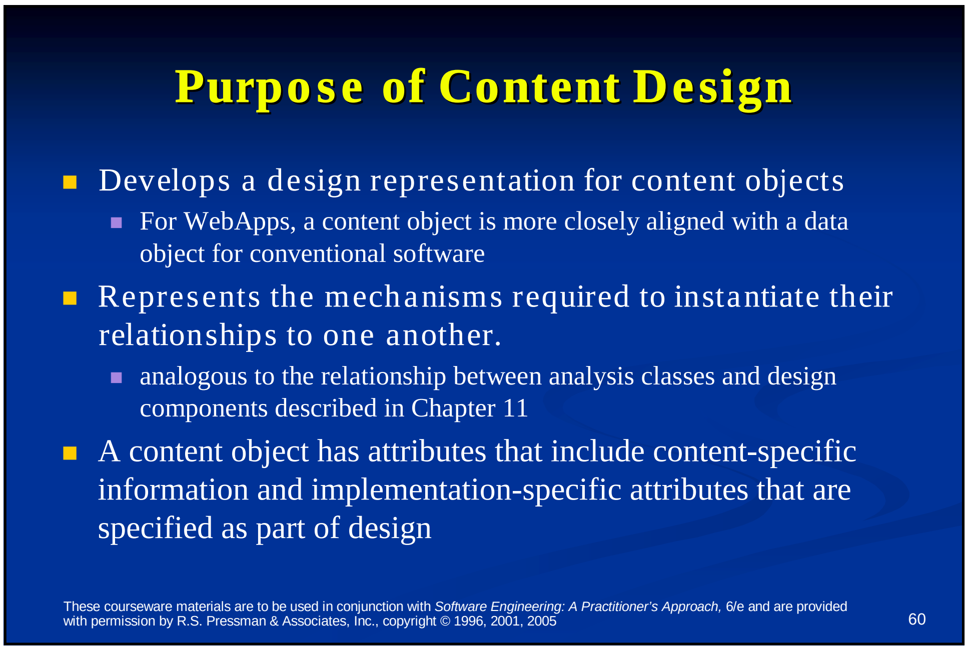
Mapping User Objectives





Interface Design Workflow-II

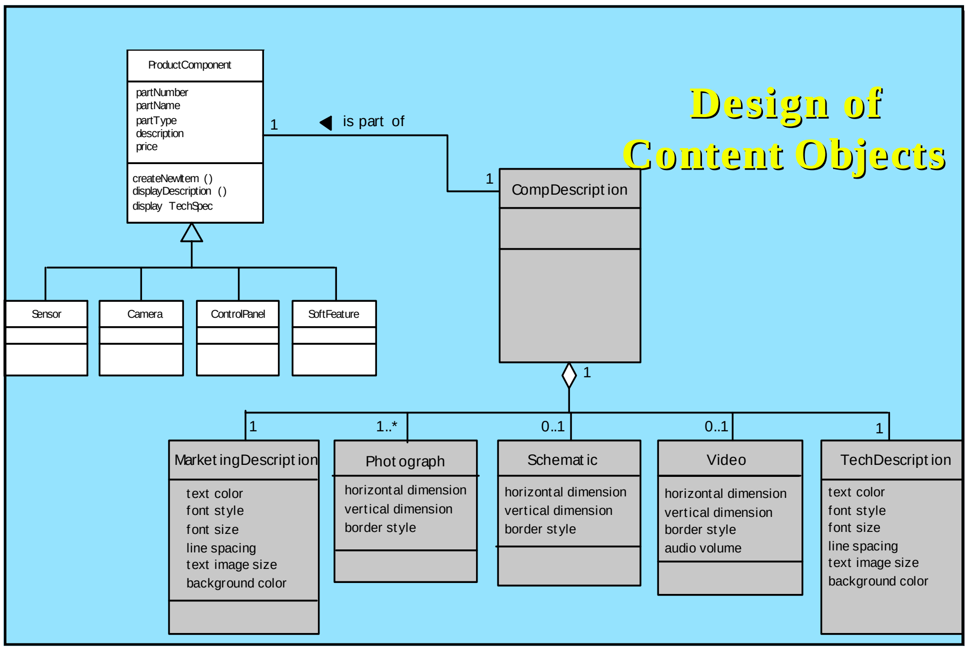
- Identify user interface objects that are required to implement the interface.
- Develop a procedural representation of the user’s interaction with the interface.
- Develop a behavioral representation (i.e. UML state chart) of the interface.
- Describe the interface layout for each state.
- Refine and review the interface design model.



Purpose of Content Design

- Develops a design representation for content objects
  - For WebApps, a content object is more closely aligned with a data object for conventional software
- Represents the mechanisms required to instantiate their relationships to one another.
  - analogous to the relationship between analysis classes and design components described in Chapter 11
- A content object has attributes that include content-specific information and implementation-specific attributes that are specified as part of design

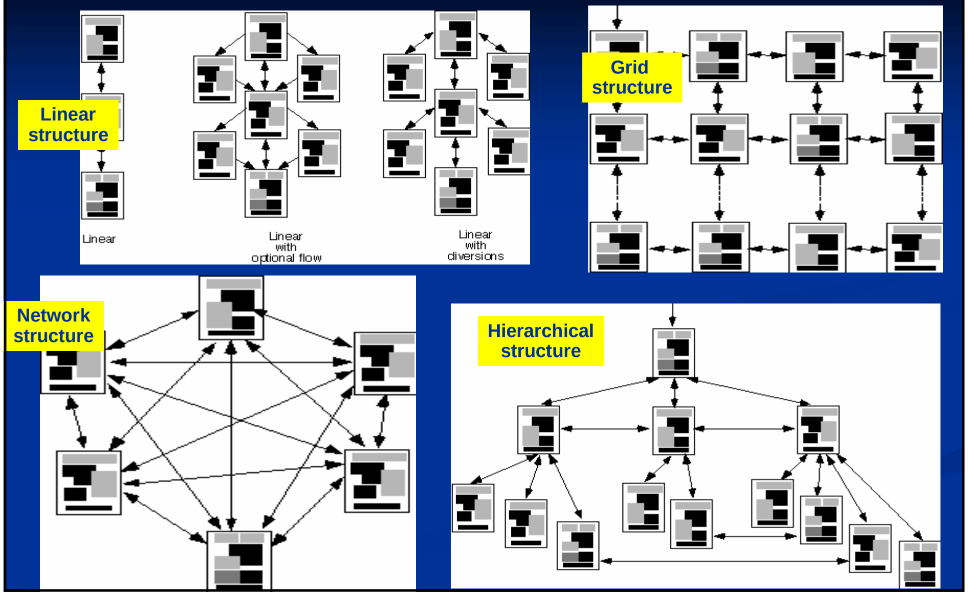
Design of Content Objects



Architecture Design

- *Content architecture* focuses on the manner in which content objects (or composite objects such as Web pages) are structured for presentation and navigation.
  - The term information architecture is also used to mean structures that lead to better organization, labeling, navigation, and searching of content objects.
- *WebApp architecture* addresses the manner in which the application is structured to manage user interaction, handle internal processing tasks, effect navigation, and present content.
  - Webapp architecture design is conducted in parallel with interface design and content design.

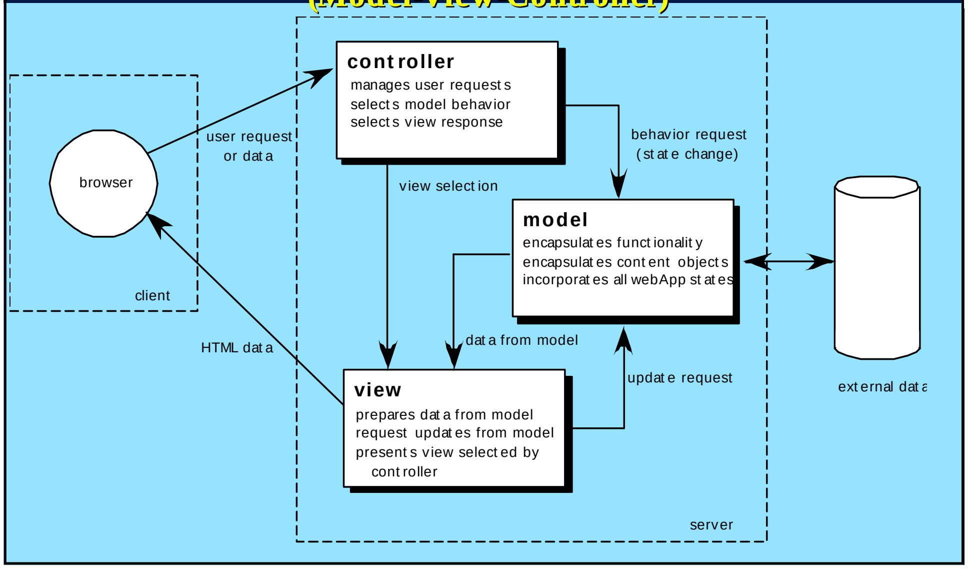
Content Architecture



MVC Architecture
(Model-View-Controller)

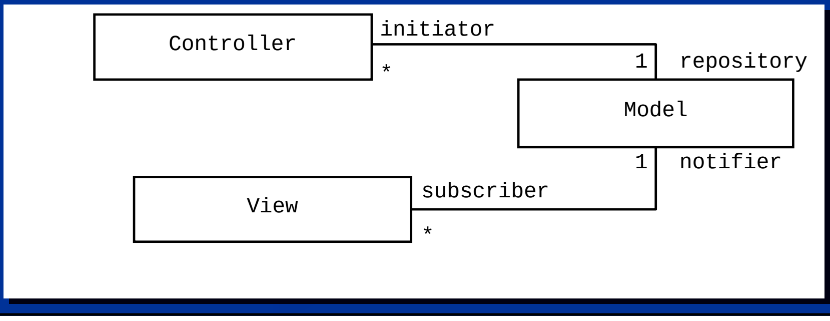
- The *model* contains all application specific content and processing logic, including
  - all content objects
  - access to external data/information sources,
  - all processing functionality that are application specific
- The *view* contains all interface specific functions and enables
  - the presentation of content and processing logic
  - access to external data/information sources,
  - all processing functionality required by the end-user.
- The *controller* manages access to the model and the view and coordinates the flow of data between them.

MVC Architecture
(Model-View-Controller)

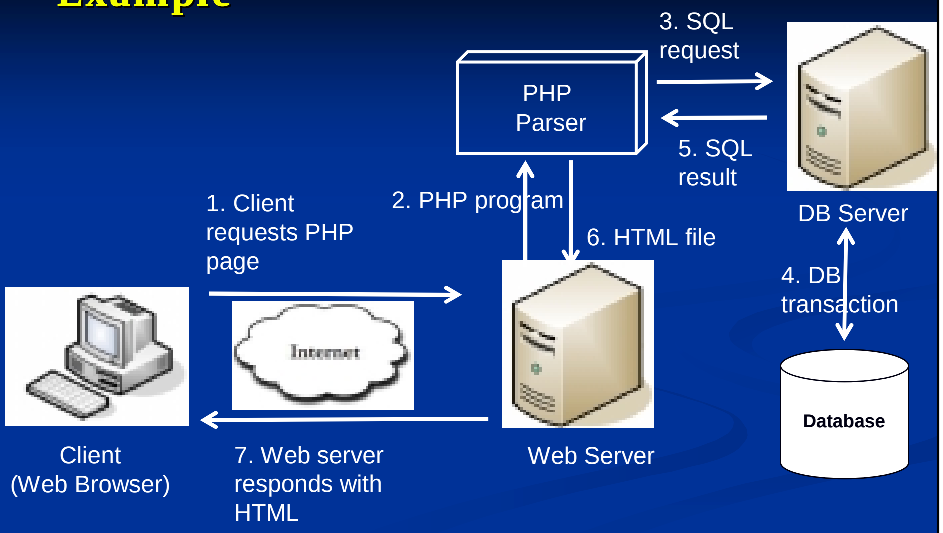


MVC Architecture

- Subsystems are classified into 3 different types
  - Model subsystem: Responsible for application domain knowledge
  - View subsystem: Responsible for displaying application domain objects to the user
  - Controller subsystem: Responsible for sequence of interactions with the user and notifying views of changes in the model.
- Well suited for interactive applications.



Example

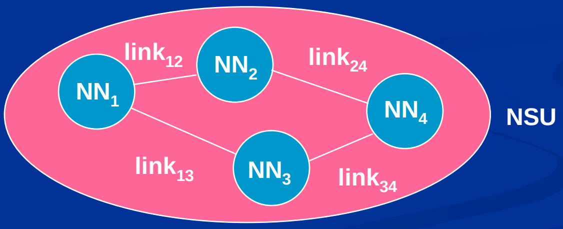


Navigation Design

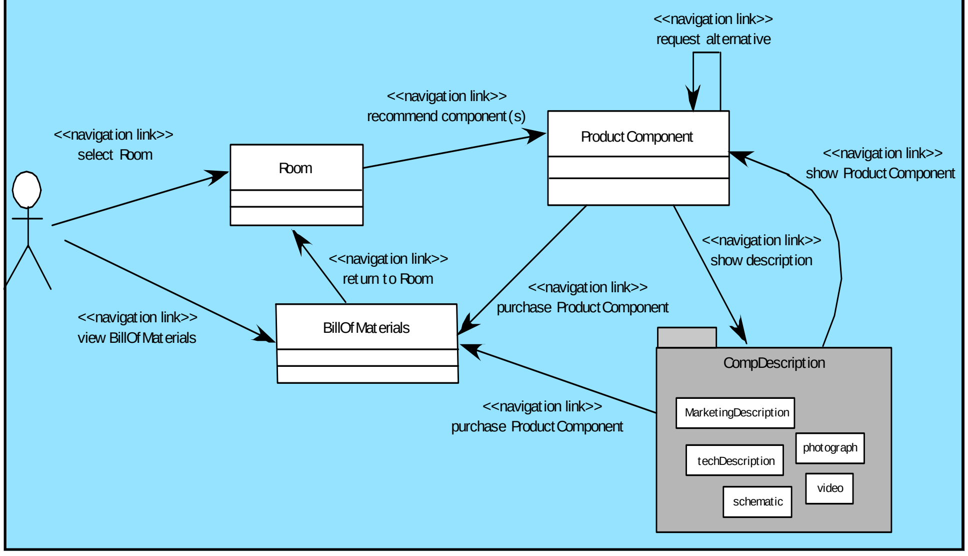
- Begins with a consideration of the user hierarchy and related use-cases
  - Each actor may use the WebApp somewhat differently and therefore have different navigation requirements
- As each user interacts with the WebApp, she encounters a series of *navigation semantic units* (NSUs)
  - NSU—"a set of information and related navigation structures that collaborate in the fulfillment of a subset of related user requirements"

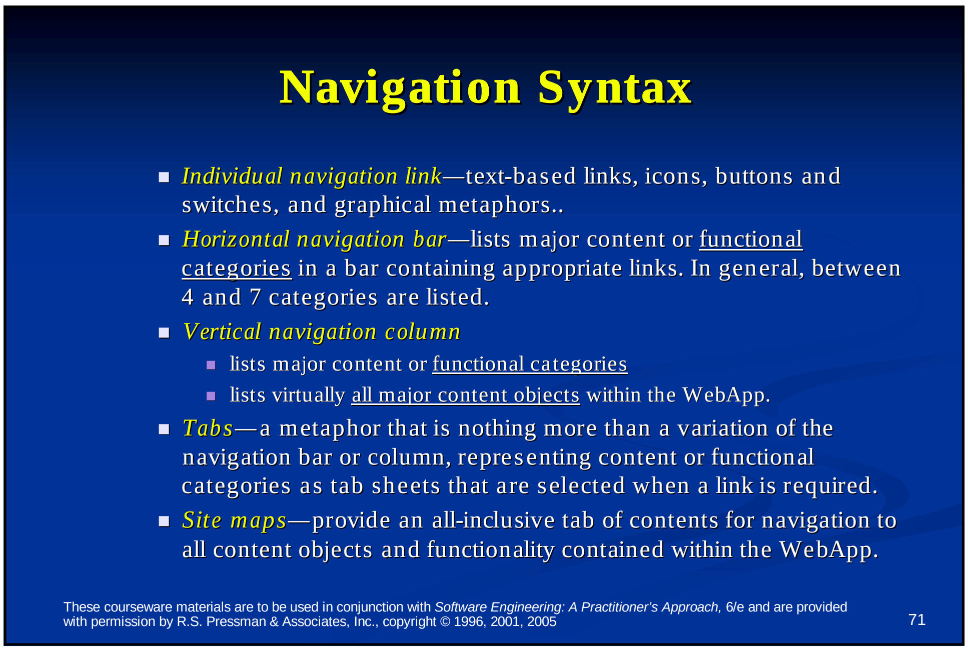
Navigation Semantic Units

- Navigation semantic unit
  - Ways of navigation—represents the best navigation way or path for users with certain profiles to achieve their desired goal or sub-goal. Composed of …
    - Navigation nodes (NN) connected by Navigation links



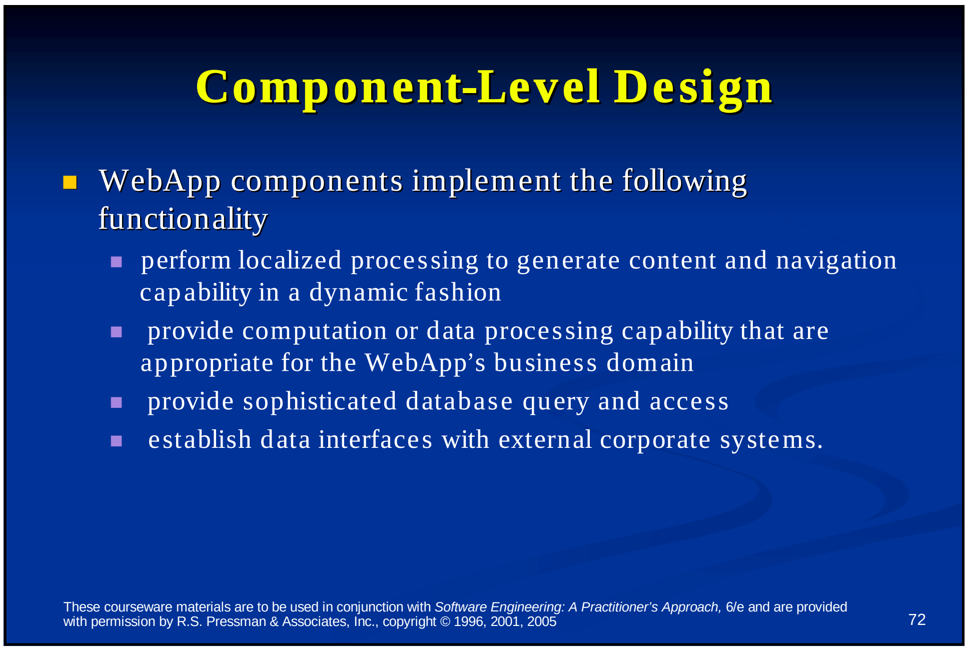
Creating a Navigation Semantic Unit


<<navigation link>> request alternative
<<navigation link>> recommend component(s)
<<navigation link>> select Room
Product Component
Room
<<navigation link>> show Product Component
<<navigation link>> show description
<<navigation link>> return to Room
<<navigation link>> purchase Product Component
<<navigation link>> view BillOfMaterials
BillOfMaterials
CompDescription
<<navigation link>> purchase Product Component
MarketingDescription
techDescription
photograph
video
schematic



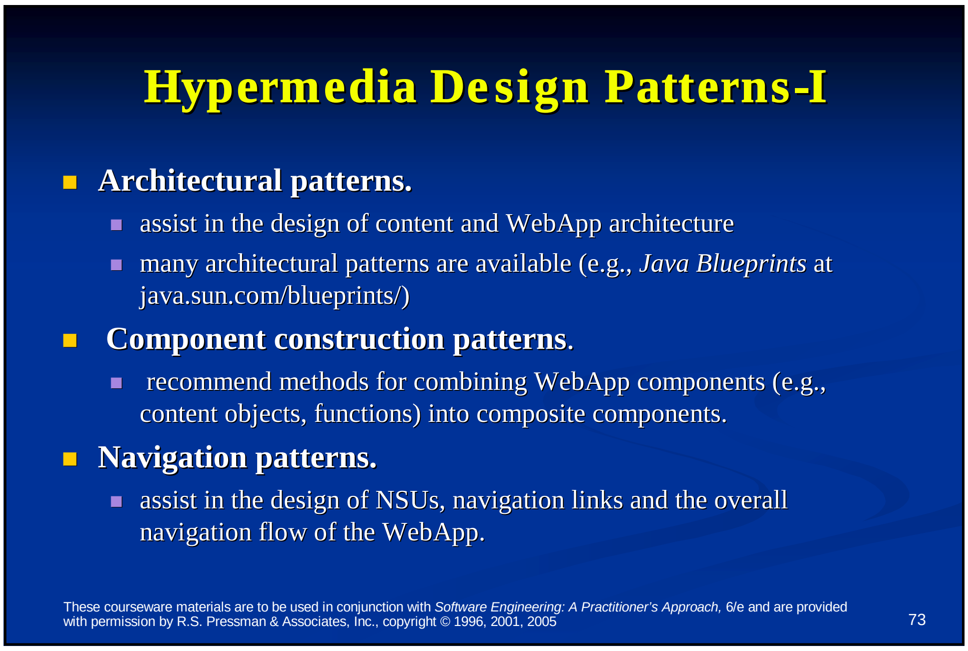
Navigation Syntax

- *Individual navigation link*—text-based links, icons, buttons and switches, and graphical metaphors..
- *Horizontal navigation bar*—lists major content or functional categories in a bar containing appropriate links. In general, between 4 and 7 categories are listed.
- *Vertical navigation column*
  - lists major content or functional categories
  - lists virtually all major content objects within the WebApp.
- *Tabs*—a metaphor that is nothing more than a variation of the navigation bar or column, representing content or functional categories as tab sheets that are selected when a link is required.
- *Site maps*—provide an all-inclusive tab of contents for navigation to all content objects and functionality contained within the WebApp.

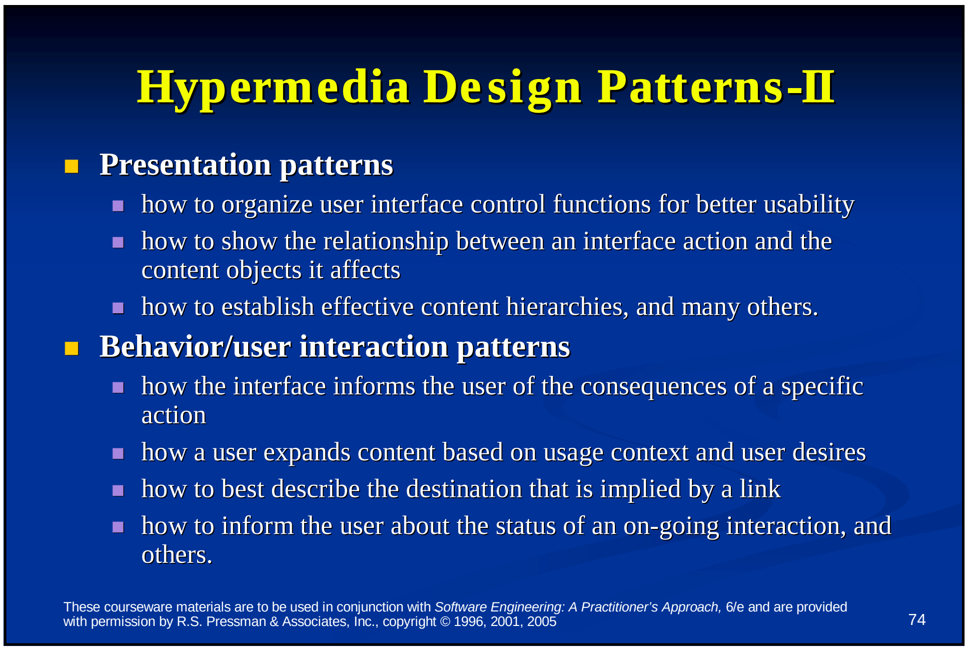


Component-Level Design

- WebApp components implement the following functionality
  - perform localized processing to generate content and navigation capability in a dynamic fashion
  - provide computation or data processing capability that are appropriate for the WebApp's business domain
  - provide sophisticated database query and access
  - establish data interfaces with external corporate systems.


Hypermedia Design Patterns-I

- **Architectural patterns.**
  - assist in the design of content and WebApp architecture
  - many architectural patterns are available (e.g., *Java Blueprints* at java.sun.com/blueprints/)
- **Component construction patterns.**
  - recommend methods for combining WebApp components (e.g., content objects, functions) into composite components.
- **Navigation patterns.**
  - assist in the design of NSUs, navigation links and the overall navigation flow of the WebApp.


Hypermedia Design Patterns-II

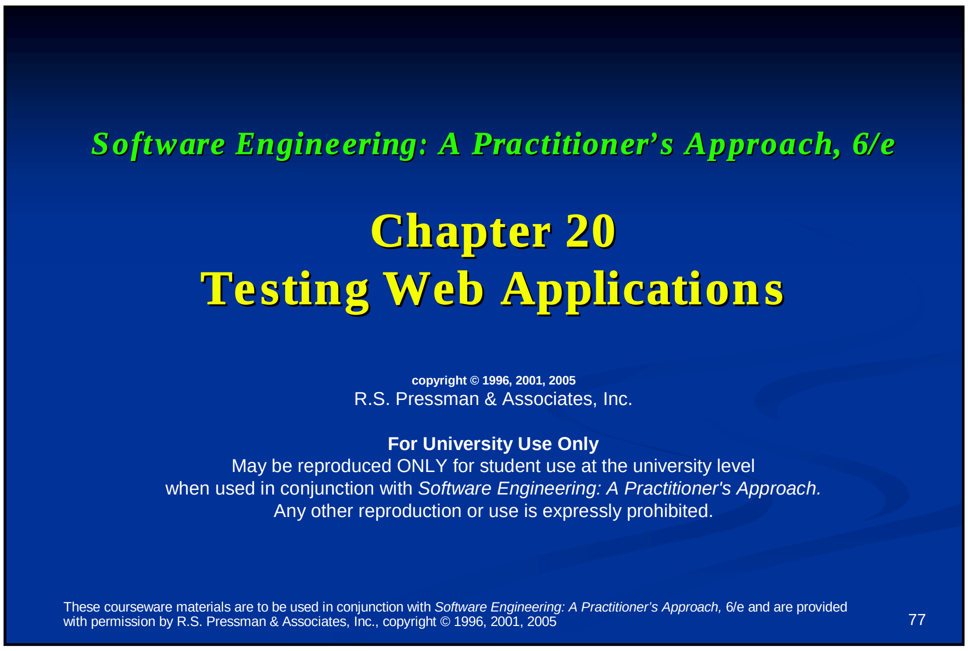
- **Presentation patterns**
  - how to organize user interface control functions for better usability
  - how to show the relationship between an interface action and the content objects it affects
  - how to establish effective content hierarchies, and many others.
- **Behavior/user interaction patterns**
  - how the interface informs the user of the consequences of a specific action
  - how a user expands content based on usage context and user desires
  - how to best describe the destination that is implied by a link
  - how to inform the user about the status of an on-going interaction, and others.

Patterns Repositories

- **Hypermedia Design Patterns Repository**
  - http://www.designpattern.lu.unisi.ch/
- **InteractionPatterns by TomErickson**
  - http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html
- **Web Design Patterns by Martijn vanWelie**
  - http://www.welie.com/patterns/
- **Improving Web Information Systems with Navigational Patterns**
  - http://www8.org/w8-papers/5b-hypertext-media/improving/improving.html
- **An HTML 2.0 Pattern Language**
  - http://www.anamorph.com/docs/patterns/default.html
- **Common Ground - A Pattern Language for HCI Design**
  - http://www.mit.edu/~jtidwell/interaction_patterns.html
- **Patterns for Personal Web Sites**
  - http://www.rdrop.com/~half/Creations/Writings/Web.patterns/index.html
- **Indexing Pattern Language**
  - http://www.cs.brown.edu/~rms/InformationStructures/Indexing/Overview.html

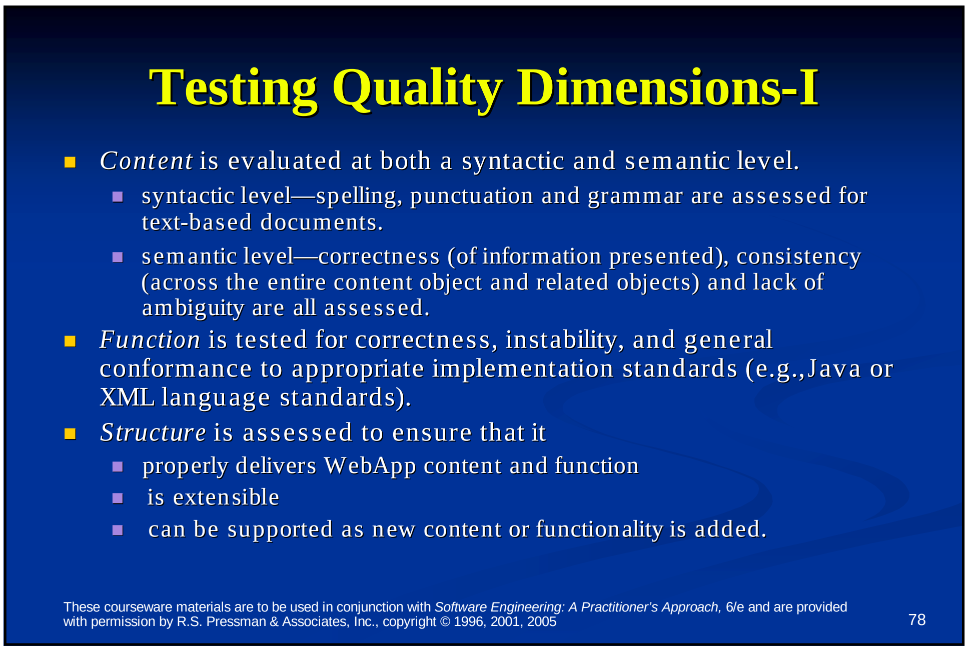
Design Metrics

- Does the user interface promote usability?
- Are the aesthetics of the WebApp appropriate for the application domain and pleasing to the user?
- Is the content designed in a manner that gives the most information with the least effort?
- Is navigation efficient and straightforward?
- Has the WebApp architecture been designed to accommodate the special goals and objectives of WebApp users, the structure of content and functionality, and the flow of navigation required to use the system effectively?
- Are components designed in a manner that reduces procedural complexity and enhances the correctness, reliability and performance?



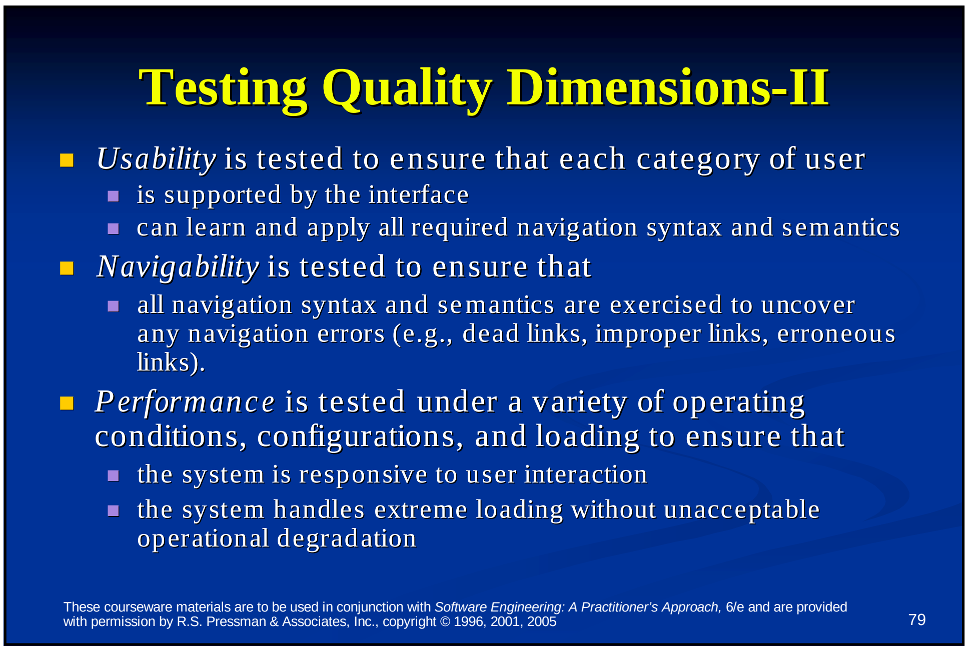
Software Engineering: A Practitioner's Approach, 6/e

Chapter 20
Testing Web Applications



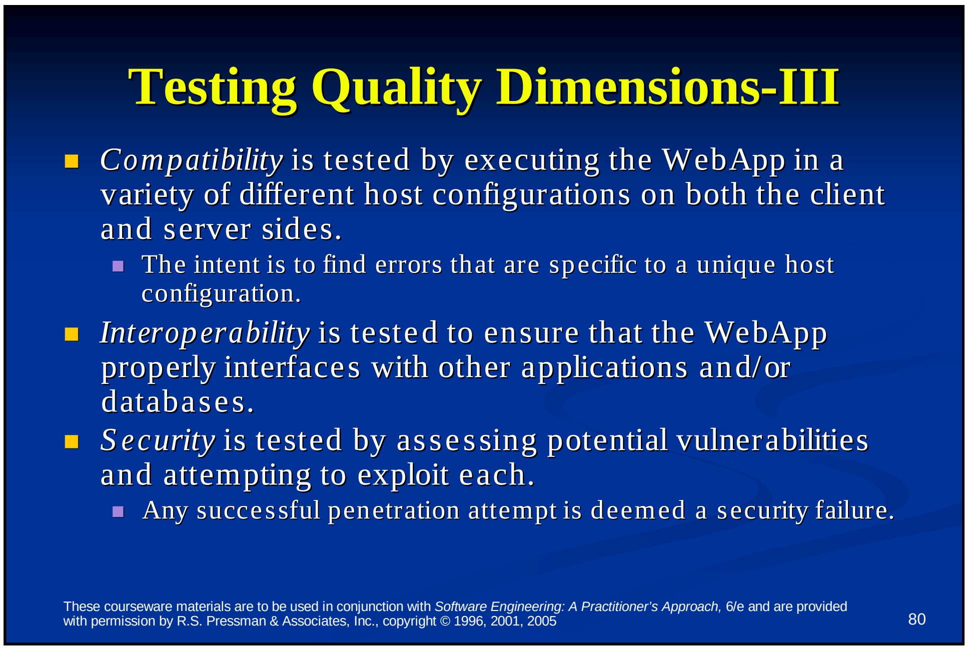
Testing Quality Dimensions-I

- *Content* is evaluated at both a syntactic and semantic level.
  - syntactic level—spelling, punctuation and grammar are assessed for text-based documents.
  - semantic level—correctness (of information presented), consistency (across the entire content object and related objects) and lack of ambiguity are all assessed.
- *Function* is tested for correctness, instability, and general conformance to appropriate implementation standards (e.g.,Java or XML language standards).
- *Structure* is assessed to ensure that it
  - properly delivers WebApp content and function
  - is extensible
  - can be supported as new content or functionality is added.



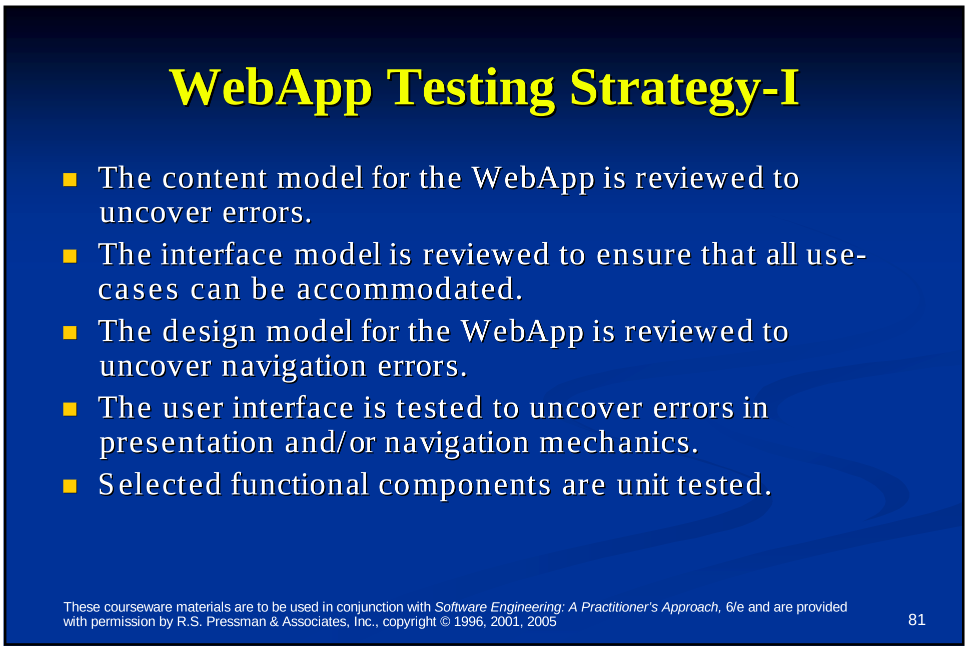
Testing Quality Dimensions-II

- *Usability* is tested to ensure that each category of user
  - is supported by the interface
  - can learn and apply all required navigation syntax and semantics
- *Navigability* is tested to ensure that
  - all navigation syntax and semantics are exercised to uncover any navigation errors (e.g., dead links, improper links, erroneous links).
- *Performance* is tested under a variety of operating conditions, configurations, and loading to ensure that
  - the system is responsive to user interaction
  - the system handles extreme loading without unacceptable operational degradation



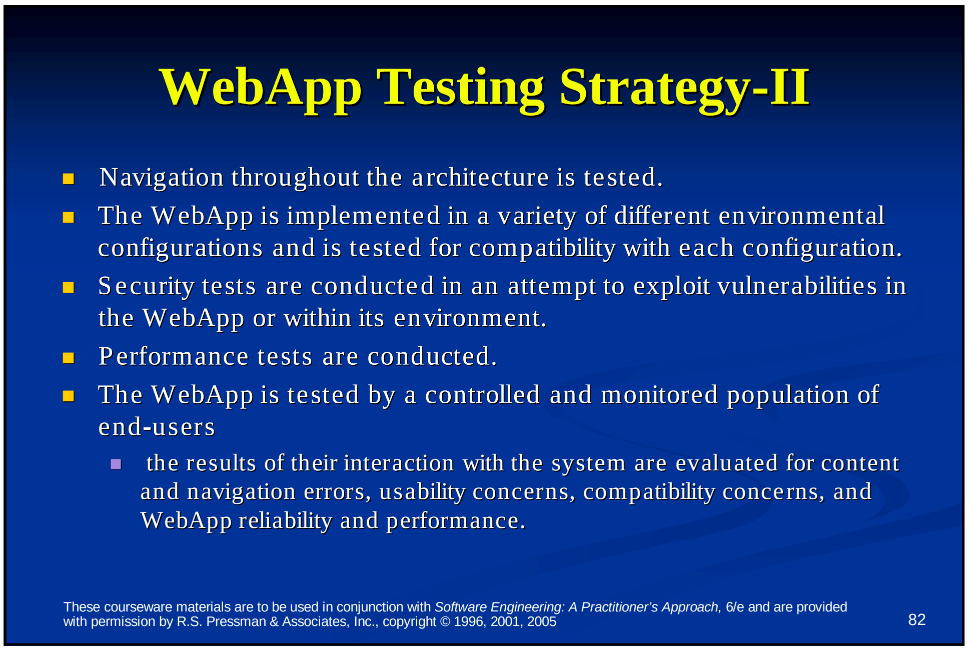
Testing Quality Dimensions-III

- *Compatibility* is tested by executing the WebApp in a variety of different host configurations on both the client and server sides.
  - The intent is to find errors that are specific to a unique host configuration.
- *Interoperability* is tested to ensure that the WebApp properly interfaces with other applications and/or databases.
- *Security* is tested by assessing potential vulnerabilities and attempting to exploit each.
  - Any successful penetration attempt is deemed a security failure.



WebApp Testing Strategy-I

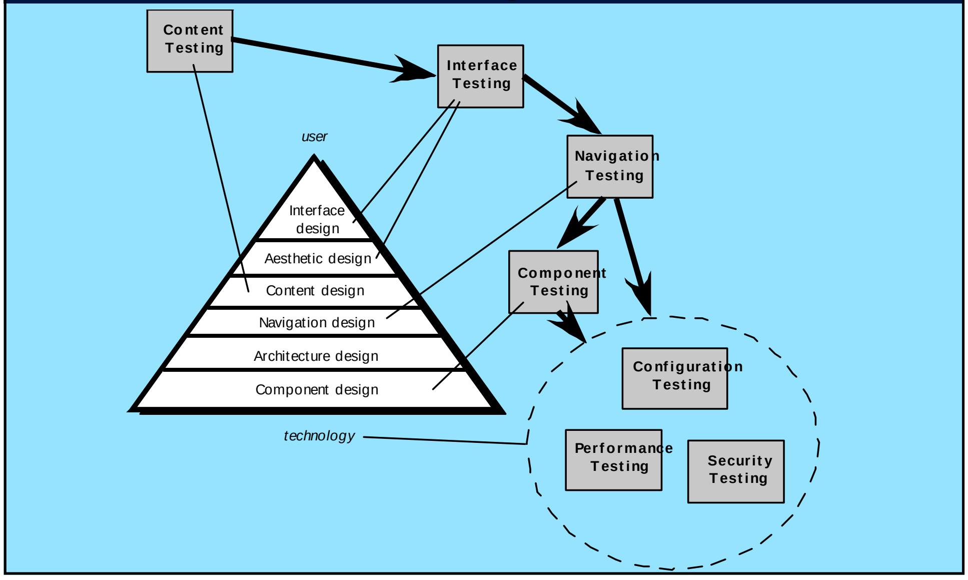
- The content model for the WebApp is reviewed to uncover errors.
- The interface model is reviewed to ensure that all use-cases can be accommodated.
- The design model for the WebApp is reviewed to uncover navigation errors.
- The user interface is tested to uncover errors in presentation and/or navigation mechanics.
- Selected functional components are unit tested.



WebApp Testing Strategy-II

- Navigation throughout the architecture is tested.
- The WebApp is implemented in a variety of different environmental configurations and is tested for compatibility with each configuration.
- Security tests are conducted in an attempt to exploit vulnerabilities in the WebApp or within its environment.
- Performance tests are conducted.
- The WebApp is tested by a controlled and monitored population of end-users
  - the results of their interaction with the system are evaluated for content and navigation errors, usability concerns, compatibility concerns, and WebApp reliability and performance.

The Testing Process



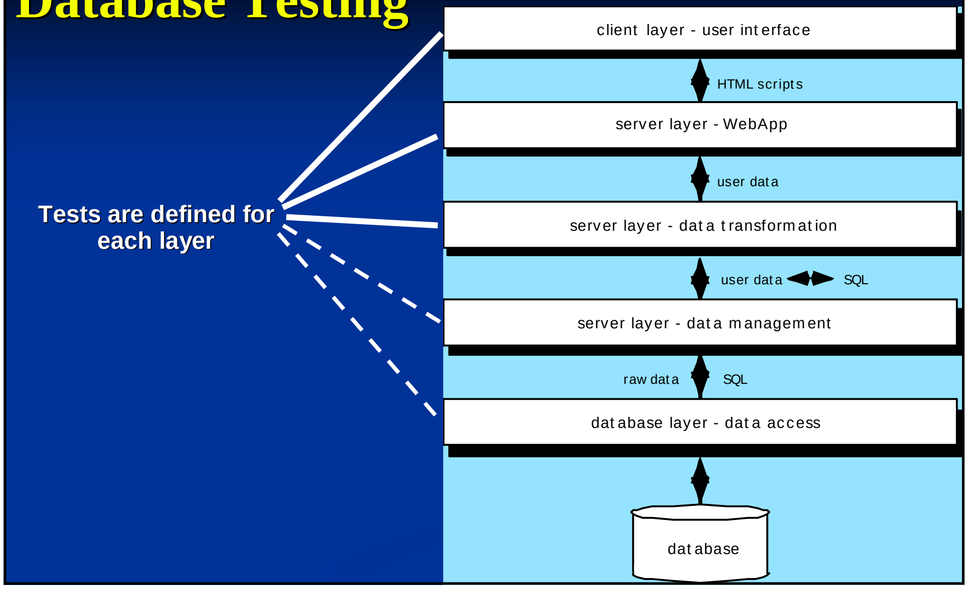
Content Testing

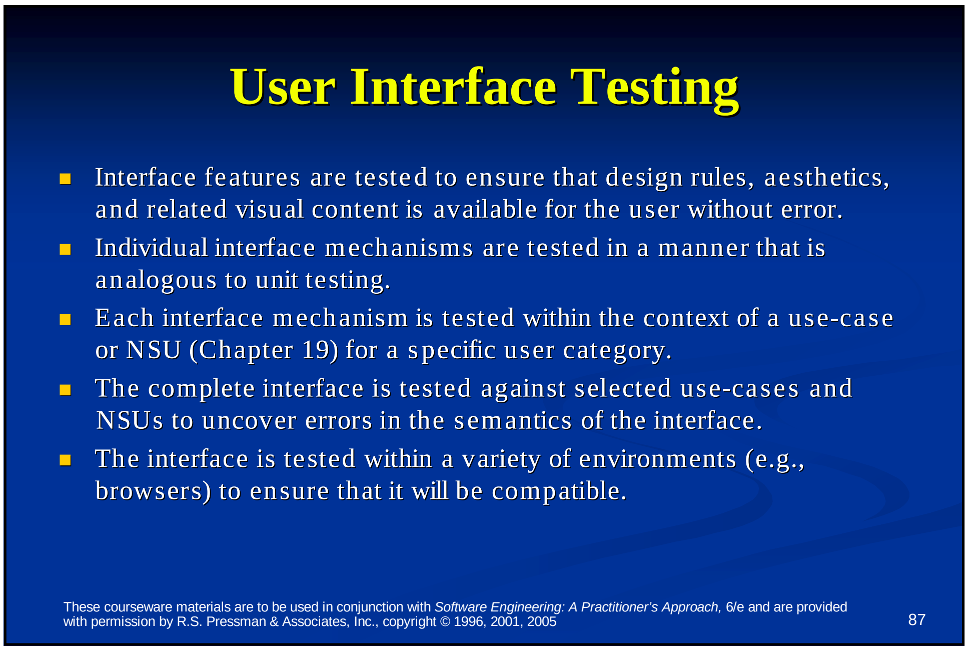
- Content testing has three important objectives:
  - to uncover syntactic errors (e.g., typos, grammar mistakes) in text-based documents, graphical representations, and other media
  - to uncover semantic errors (i.e., errors in the accuracy or completeness of information) in any content object presented as navigation occurs, and
  - to find errors in the organization or structure of content that is presented to the end-user.

Assessing Content Semantics

- Is the information factually accurate?
- Is the information concise and to the point?
- Is the layout of the content object easy for the user to understand?
- Can information embedded within a content object be found easily?
- Have proper references been provided for all information derived from other sources?
- Is the information presented consistent internally and consistent with information presented in other content objects?
- Is the content offensive, misleading, or does it open the door to litigation?
- Does the content infringe on existing copyrights or trademarks?
- Does the content contain internal links that supplement existing content? Are the links correct?
- Does the aesthetic style of the content conflict with the aesthetic style of the interface?

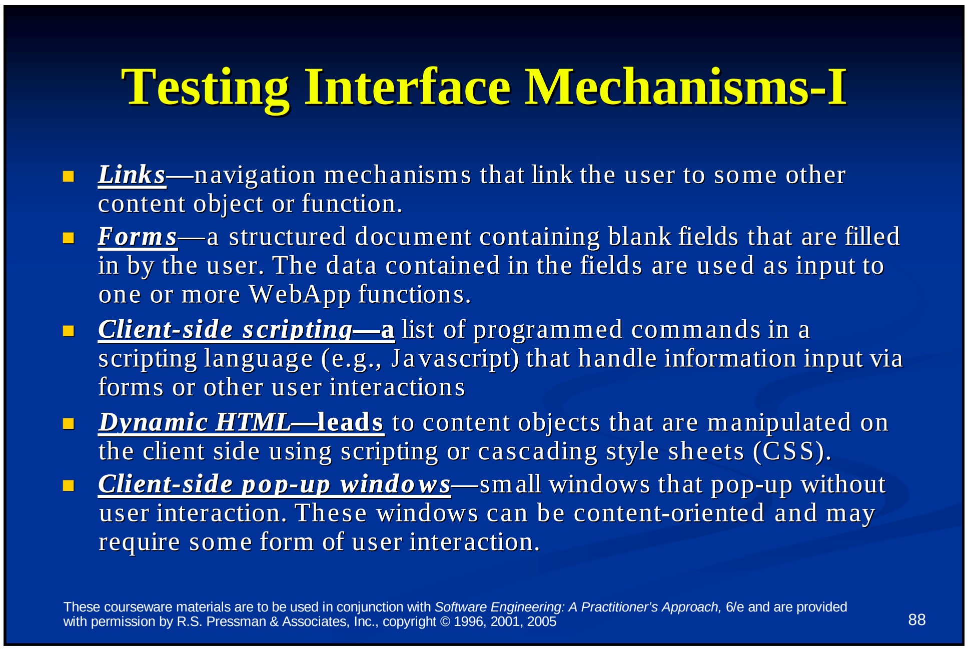
Database Testing


Tests are defined for each layer
client layer - user interface
HTML scripts
server layer - WebApp
user data
server layer - data transformation
user data
SQL
server layer - data management
raw data
SQL
database layer - data access
database



User Interface Testing

- Interface features are tested to ensure that design rules, aesthetics, and related visual content is available for the user without error.
- Individual interface mechanisms are tested in a manner that is analogous to unit testing.
- Each interface mechanism is tested within the context of a use-case or NSU (Chapter 19) for a specific user category.
- The complete interface is tested against selected use-cases and NSUs to uncover errors in the semantics of the interface.
- The interface is tested within a variety of environments (e.g., browsers) to ensure that it will be compatible.



Testing Interface Mechanisms-I

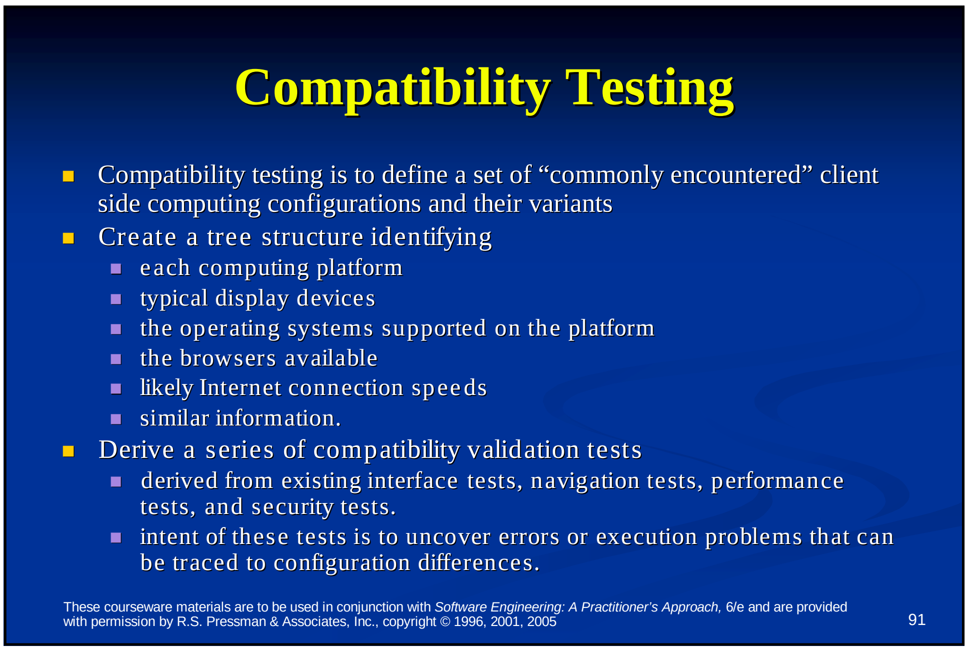
- ***Links***—navigation mechanisms that link the user to some other content object or function.
- ***Forms***—a structured document containing blank fields that are filled in by the user. The data contained in the fields are used as input to one or more WebApp functions.
- ***Client-side scripting*—a** list of programmed commands in a scripting language (e.g., Javascript) that handle information input via forms or other user interactions
- ***Dynamic HTML*—leads** to content objects that are manipulated on the client side using scripting or cascading style sheets (CSS).
- ***Client-side pop-up windows***—small windows that pop-up without user interaction. These windows can be content-oriented and may require some form of user interaction.

Testing Interface Mechanisms-II

- ***CGI scripts***—a common gateway interface (CGI) script like PHP or ASP implements a standard method that allows a Web server to interact dynamically with users (e.g., a WebApp that contains forms may use a CGI script to process the data contained in the form once it is submitted by the user).
- ***Streaming content*** —rather than waiting for a request from the client-side, content objects are downloaded automatically from the server side. This approach is sometimes called “push” technology because the server pushes data to the client.
- ***Cookies*** —a block of data sent by the server and stored by a browser as a consequence of a specific user interaction. The content of the data is WebApp-specific (e.g., user identification data or a list of items that have been selected for purchase by the user).
- ***Application specific interface mechanisms*** —include one or more “macro” interface mechanisms such as a shopping cart, credit card processing, or a shipping cost calculator.

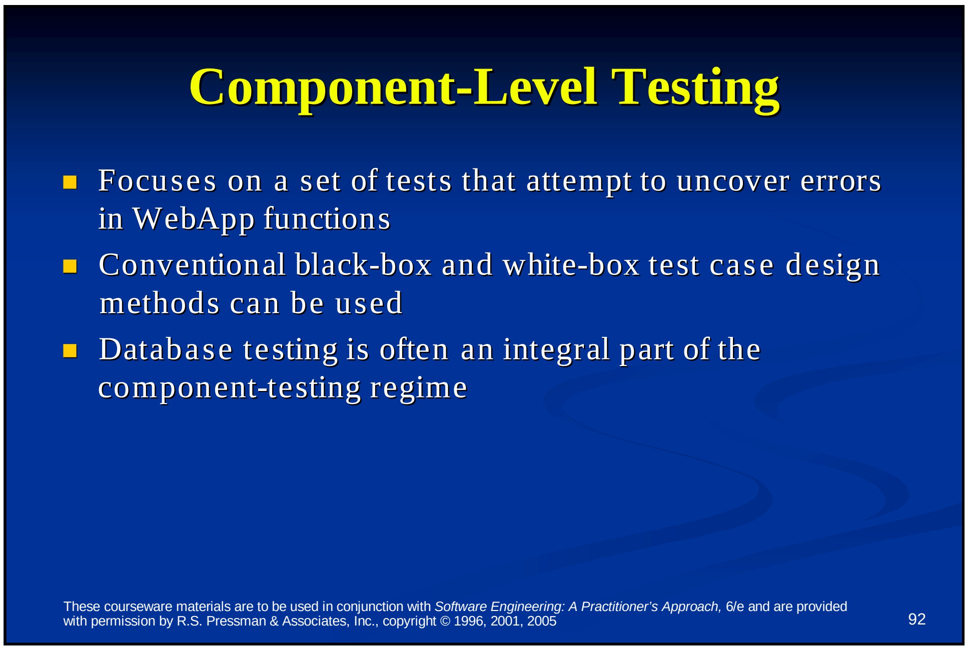
Usability Tests

- Design by WebE team … executed by end-users
- Testing sequence …
  - Define a set of usability testing categories and identify goals for each.
  - Design tests that will enable each goal to be evaluated.
  - Select participants who will conduct the tests.
  - Instrument participants’ interaction with the WebApp while testing is conducted.
  - Develop a mechanism for assessing the usability of the WebApp
- different levels of abstraction:
  - the usability of a specific interface mechanism (e.g., a form) can be assessed
  - the usability of a complete Web page (encompassing interface mechanisms, data objects and related functions) can be evaluated
  - the usability of the complete WebApp can be considered.



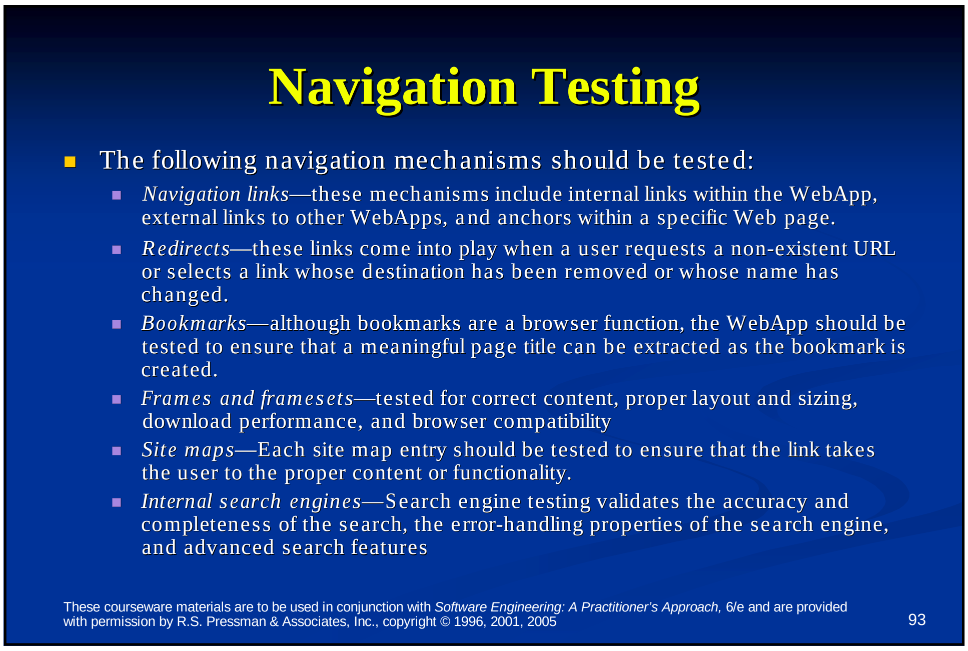
Compatibility Testing

- Compatibility testing is to define a set of “commonly encountered” client side computing configurations and their variants
- Create a tree structure identifying
  - each computing platform
  - typical display devices
  - the operating systems supported on the platform
  - the browsers available
  - likely Internet connection speeds
  - similar information.
- Derive a series of compatibility validation tests
  - derived from existing interface tests, navigation tests, performance tests, and security tests.
  - intent of these tests is to uncover errors or execution problems that can be traced to configuration differences.



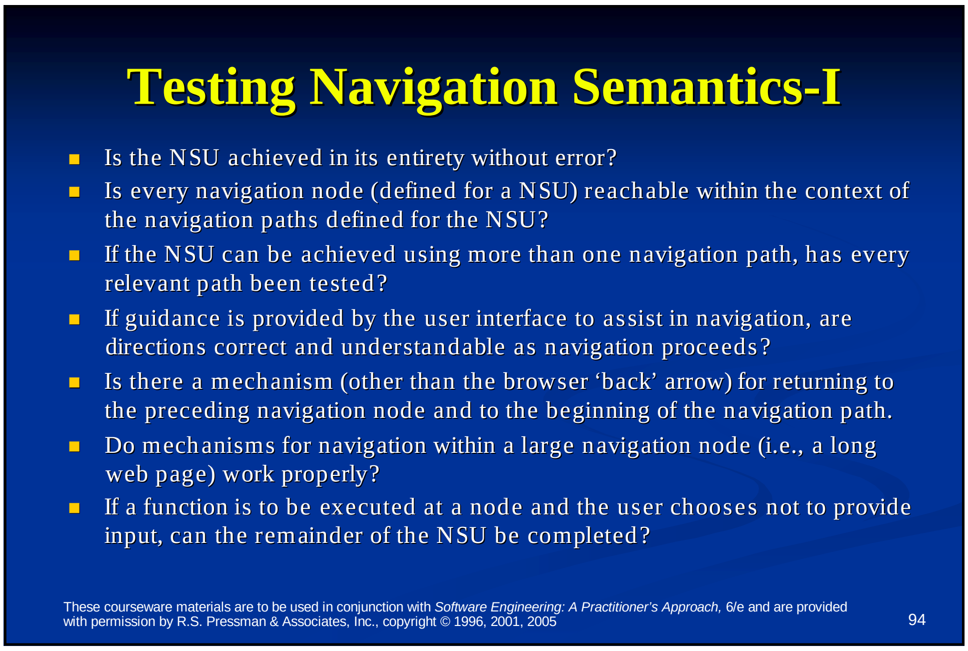
Component-Level Testing

- Focuses on a set of tests that attempt to uncover errors in WebApp functions
- Conventional black-box and white-box test case design methods can be used
- Database testing is often an integral part of the component-testing regime



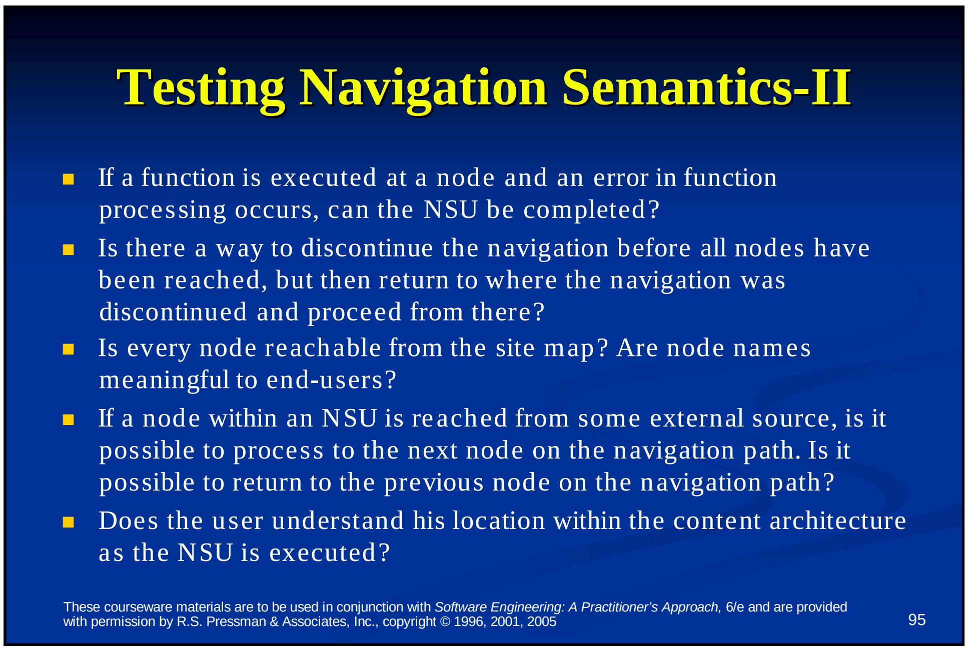
Navigation Testing

- The following navigation mechanisms should be tested:
  - *Navigation links*—these mechanisms include internal links within the WebApp, external links to other WebApps, and anchors within a specific Web page.
  - *Redirects*—these links come into play when a user requests a non-existent URL or selects a link whose destination has been removed or whose name has changed.
  - *Bookmarks*—although bookmarks are a browser function, the WebApp should be tested to ensure that a meaningful page title can be extracted as the bookmark is created.
  - *Frames and framesets*—tested for correct content, proper layout and sizing, download performance, and browser compatibility
  - *Site maps*—Each site map entry should be tested to ensure that the link takes the user to the proper content or functionality.
  - *Internal search engines*—Search engine testing validates the accuracy and completeness of the search, the error-handling properties of the search engine, and advanced search features



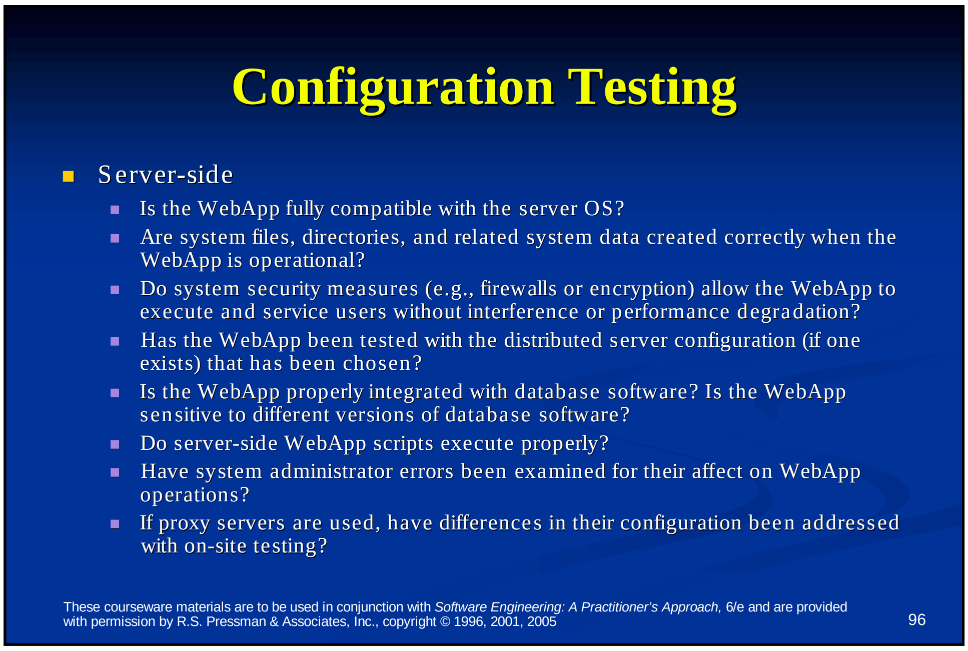
Testing Navigation Semantics-I

- Is the NSU achieved in its entirety without error?
- Is every navigation node (defined for a NSU) reachable within the context of the navigation paths defined for the NSU?
- If the NSU can be achieved using more than one navigation path, has every relevant path been tested?
- If guidance is provided by the user interface to assist in navigation, are directions correct and understandable as navigation proceeds?
- Is there a mechanism (other than the browser 'back' arrow) for returning to the preceding navigation node and to the beginning of the navigation path.
- Do mechanisms for navigation within a large navigation node (i.e., a long web page) work properly?
- If a function is to be executed at a node and the user chooses not to provide input, can the remainder of the NSU be completed?



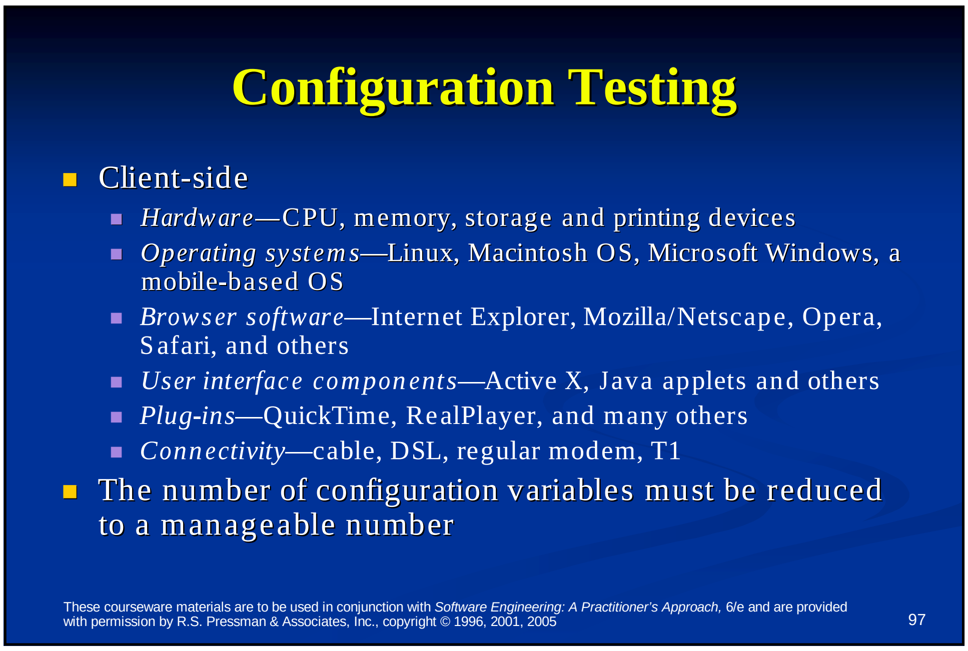
Testing Navigation Semantics-II

- If a function is executed at a node and an error in function processing occurs, can the NSU be completed?
- Is there a way to discontinue the navigation before all nodes have been reached, but then return to where the navigation was discontinued and proceed from there?
- Is every node reachable from the site map? Are node names meaningful to end-users?
- If a node within an NSU is reached from some external source, is it possible to process to the next node on the navigation path. Is it possible to return to the previous node on the navigation path?
- Does the user understand his location within the content architecture as the NSU is executed?



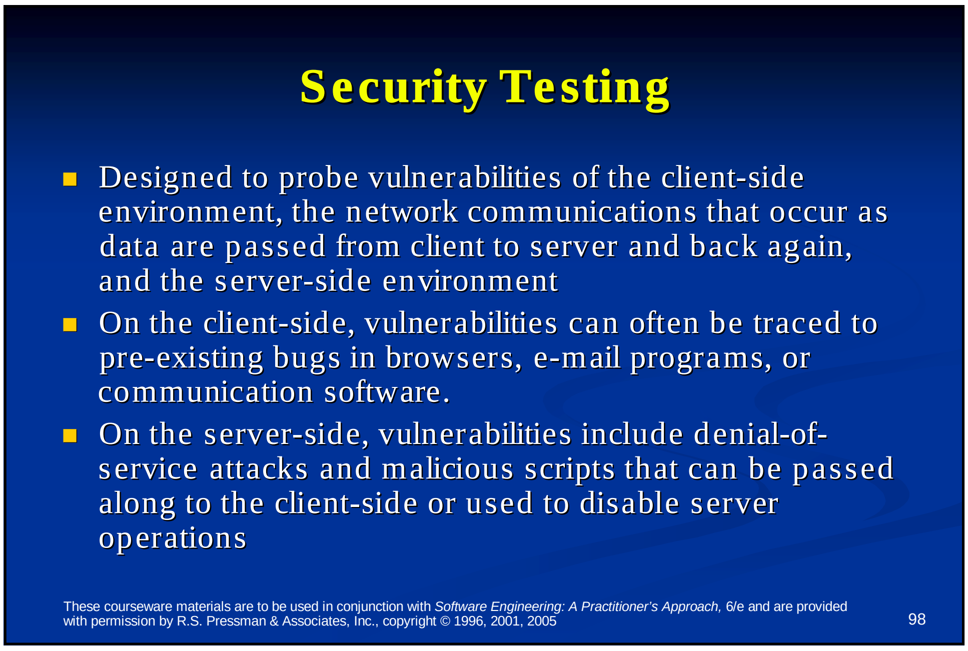
Configuration Testing

- Server-side
  - Is the WebApp fully compatible with the server OS?
  - Are system files, directories, and related system data created correctly when the WebApp is operational?
  - Do system security measures (e.g., firewalls or encryption) allow the WebApp to execute and service users without interference or performance degradation?
  - Has the WebApp been tested with the distributed server configuration (if one exists) that has been chosen?
  - Is the WebApp properly integrated with database software? Is the WebApp sensitive to different versions of database software?
  - Do server-side WebApp scripts execute properly?
  - Have system administrator errors been examined for their affect on WebApp operations?
  - If proxy servers are used, have differences in their configuration been addressed with on-site testing?



Configuration Testing

- Client-side
  - *Hardware*—CPU, memory, storage and printing devices
  - *Operating systems*—Linux, Macintosh OS, Microsoft Windows, a mobile-based OS
  - *Browser software*—Internet Explorer, Mozilla/Netscape, Opera, Safari, and others
  - *User interface components*—Active X, Java applets and others
  - *Plug-ins*—QuickTime, RealPlayer, and many others
  - *Connectivity*—cable, DSL, regular modem, T1
- The number of configuration variables must be reduced to a manageable number



Security Testing

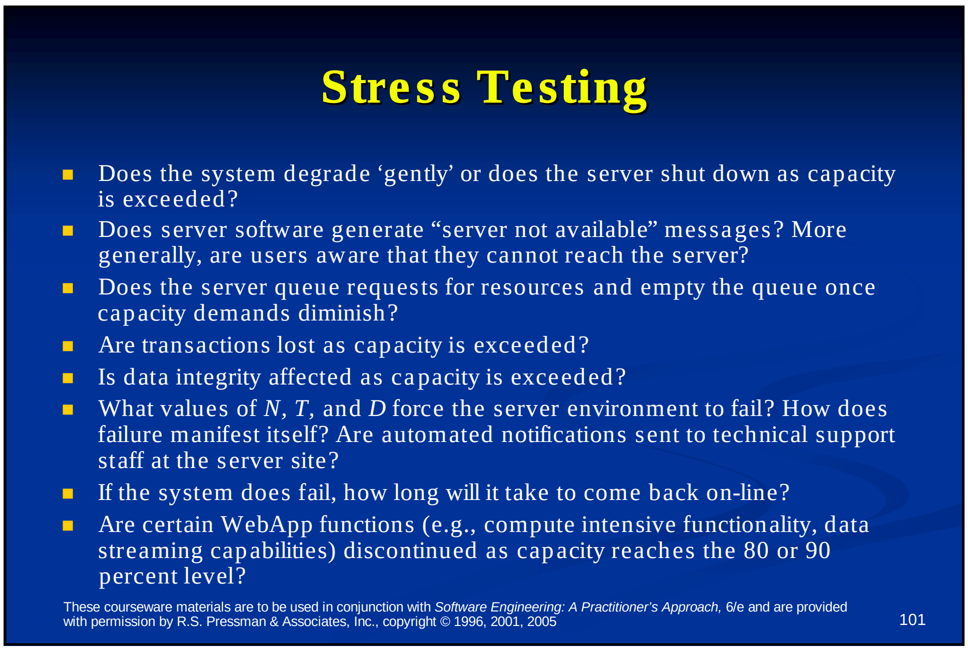
- Designed to probe vulnerabilities of the client-side environment, the network communications that occur as data are passed from client to server and back again, and the server-side environment
- On the client-side, vulnerabilities can often be traced to pre-existing bugs in browsers, e-mail programs, or communication software.
- On the server-side, vulnerabilities include denial-of-service attacks and malicious scripts that can be passed along to the client-side or used to disable server operations

Performance Testing

- Does the server response time degrade to a point where it is noticeable and unacceptable?
- At what point (in terms of users, transactions or data loading) does performance become unacceptable?
- What system components are responsible for performance degradation?
- What is the average response time for users under a variety of loading conditions?
- Does performance degradation have an impact on system security?
- Is WebApp reliability or accuracy affected as the load on the system grows?
- What happens when loads that are greater than maximum server capacity are applied?

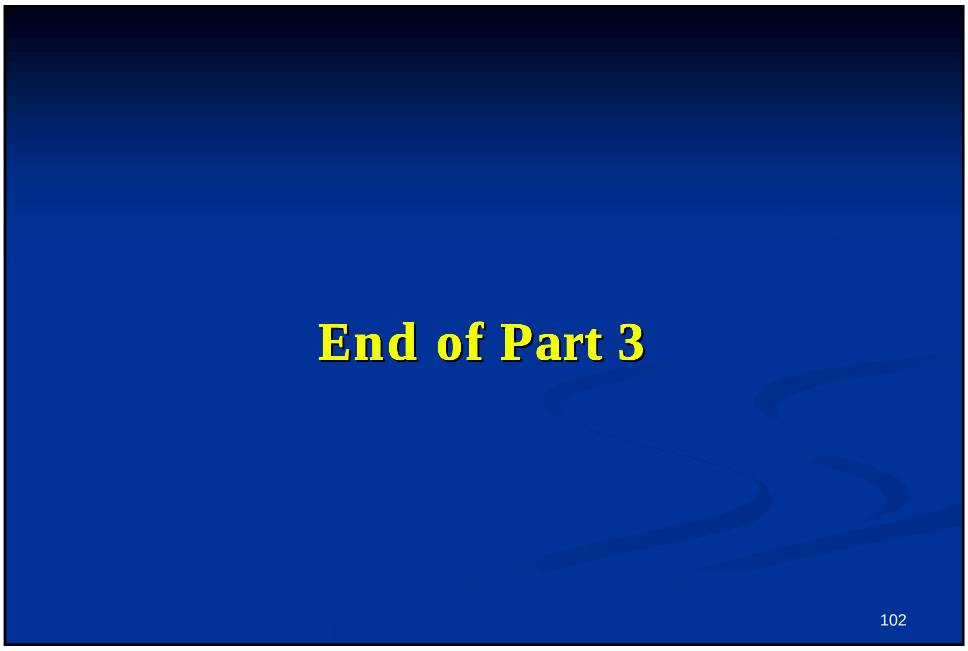
Load Testing

- The intent is to determine how the WebApp and its server-side environment will respond to various loading conditions
  - *N*, the number of concurrent users
  - *T*, the number of on-line transactions per unit of time
  - *D*, the data load processed by the server per transaction
- Overall throughput, *P*, is computed in the following manner:
  - $P = N \times T \times D$



Stress Testing

- Does the system degrade 'gently' or does the server shut down as capacity is exceeded?
- Does server software generate "server not available" messages? More generally, are users aware that they cannot reach the server?
- Does the server queue requests for resources and empty the queue once capacity demands diminish?
- Are transactions lost as capacity is exceeded?
- Is data integrity affected as capacity is exceeded?
- What values of *N*, *T*, and *D* force the server environment to fail? How does failure manifest itself? Are automated notifications sent to technical support staff at the server site?
- If the system does fail, how long will it take to come back on-line?
- Are certain WebApp functions (e.g., compute intensive functionality, data streaming capabilities) discontinued as capacity reaches the 80 or 90 percent level?



End of Part 3