



Ch.17 WebApp Design





- **Design & WebApps**

- “There are essentially two basic approaches to design: the artistic ideal of expressing yourself and the engineering ideal of solving a problem for a customer.”

Jakob Nielsen

- *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*

- Rebuff external attacks
- Exclude unauthorized access
- Ensure the privacy of users/customers

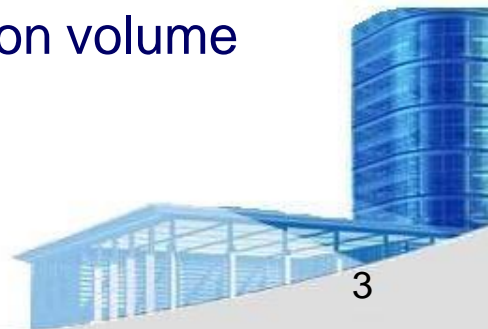
- *Availability*

- the measure of the percentage of time that a WebApp is available for use

- *Scalability*

- **Can** the WebApp and the systems with which it is interfaced handle significant variation in user 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 and Latency*

- 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 Browser-Web-Web site-Web-Browser connection quick enough?
- How does the performance vary by time of day, by load and usage?
- Is performance adequate for E-commerce applications?

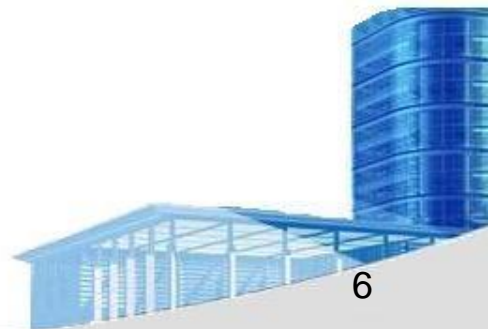




- **WebApp Design Goals**

- *Consistency*

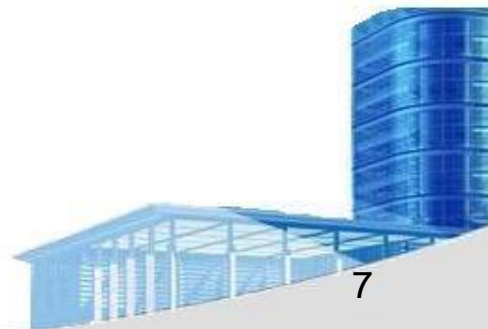
- **Content** should be constructed consistently
- **Graphic design (aesthetics)** should present a consistent look across all parts of the WebApp
- **Architectural design** should establish templates that lead to a consistent hypermedia structure
- **Interface design** should define consistent modes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





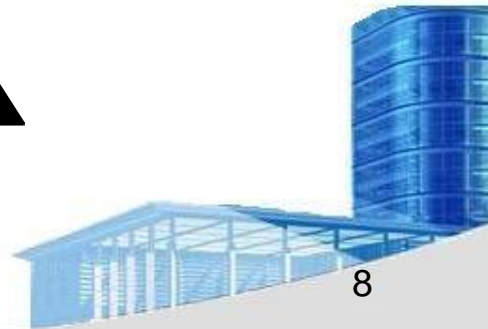
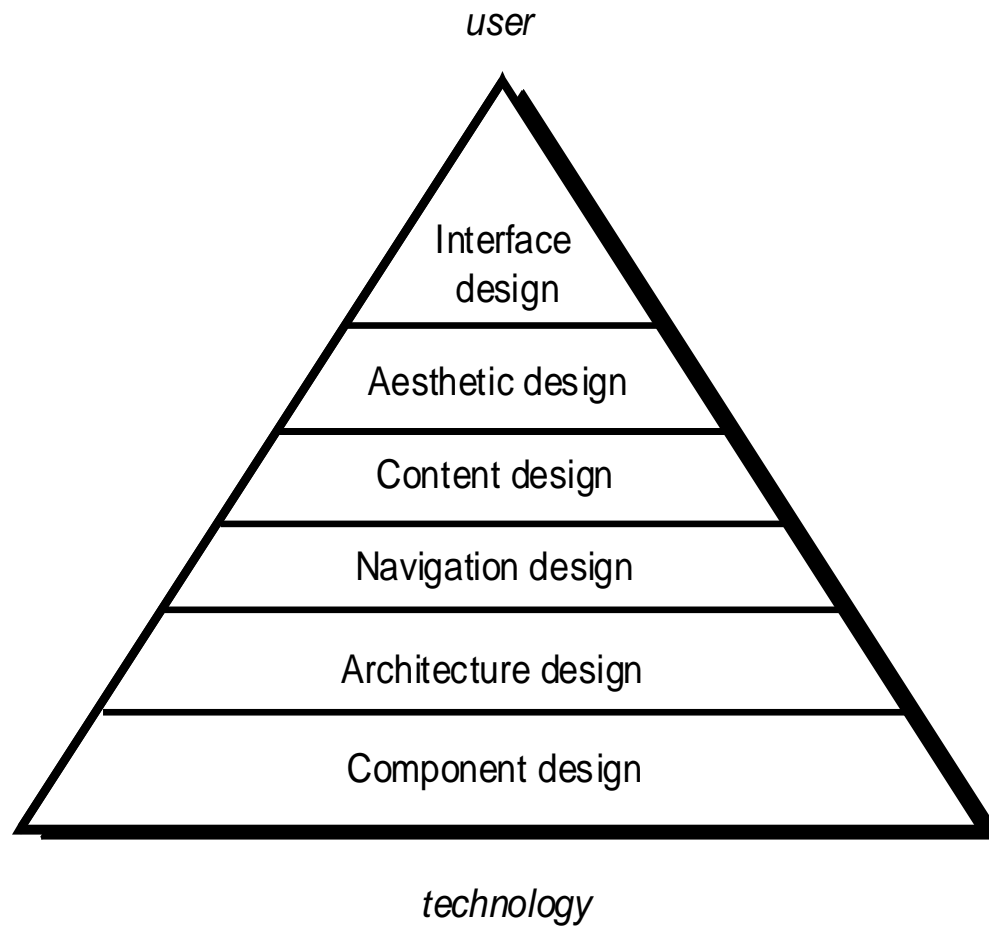
- **WebApp Design Goals**

- *Identity*
 - Establish an “identity” that is appropriate for the business purpose
- *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





- **WebApp Design Pyramid**





• WebApp Interface Design

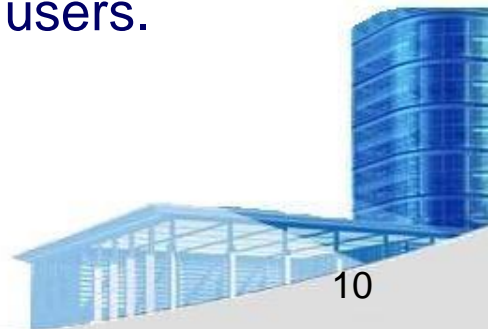
- *Where am I?* The interface should
 - provide an indication of the WebApp that has been accessed
 - inform the user of her location in the content hierarchy.
- *What can I do now?* The interface should always help the user understand his current options
 - what functions are available?
 - what links 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.





- **Effective WebApp Interfaces**

- Bruce Tognozzi [TOG01] suggests...
 - *Effective interfaces are visually apparent and forgiving*, instilling in their users a sense of control. Users quickly see the breadth of their options, grasp how to achieve their goals, and do their work.
 - *Effective interfaces do not concern the user with the inner workings of the system.* Work is carefully and continuously saved, with full option for the user to undo any activity at any time.
 - *Effective applications and services perform a maximum of work*, while requiring a minimum of information from users.





• Interface Design Principles - I

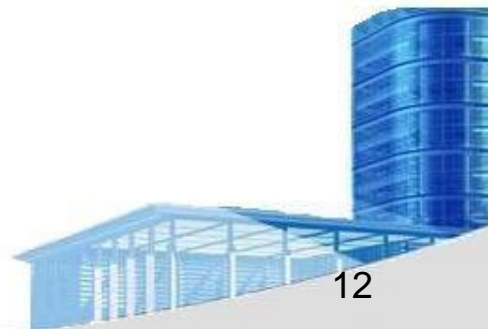
- *Anticipation*—A WebApp should be designed so that it anticipates the user's next move.
- *Communication*—The interface should communicate the status of any activity initiated by the user
- *Consistency*—The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)
- *Controlled 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 user's work efficiency, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.





• Interface Design Principles - II

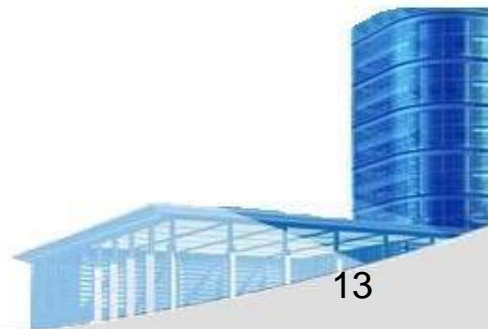
- *Focus*—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- *Fitt's Law*—“The time to acquire a target is a function of the distance to and size of the target.”
- *Human interface objects*—A vast library of reusable human interface objects has been developed for WebApps.
- *Latency 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

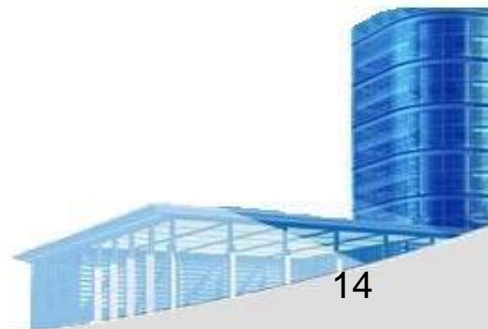
- *Maintain work product integrity*—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- *Readability*—All information presented through the interface should be readable by young and old.
- *Track state*—When 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.
- *Visible navigation*—A well-designed WebApp interface provides “the illusion that users are in the same place, with the work brought to them.”





- **Aesthetic Design**

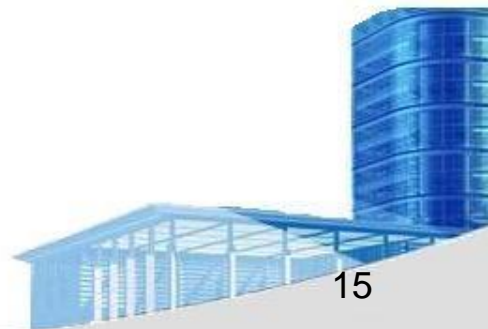
- Don't be afraid of white space.
- Emphasize content.
- Organize layout elements from top-left to bottom right.
- Group navigation, content, and function geographically within the page.
- Don't extend your real estate with the scrolling bar.
- Consider resolution and browser window size when designing layout.





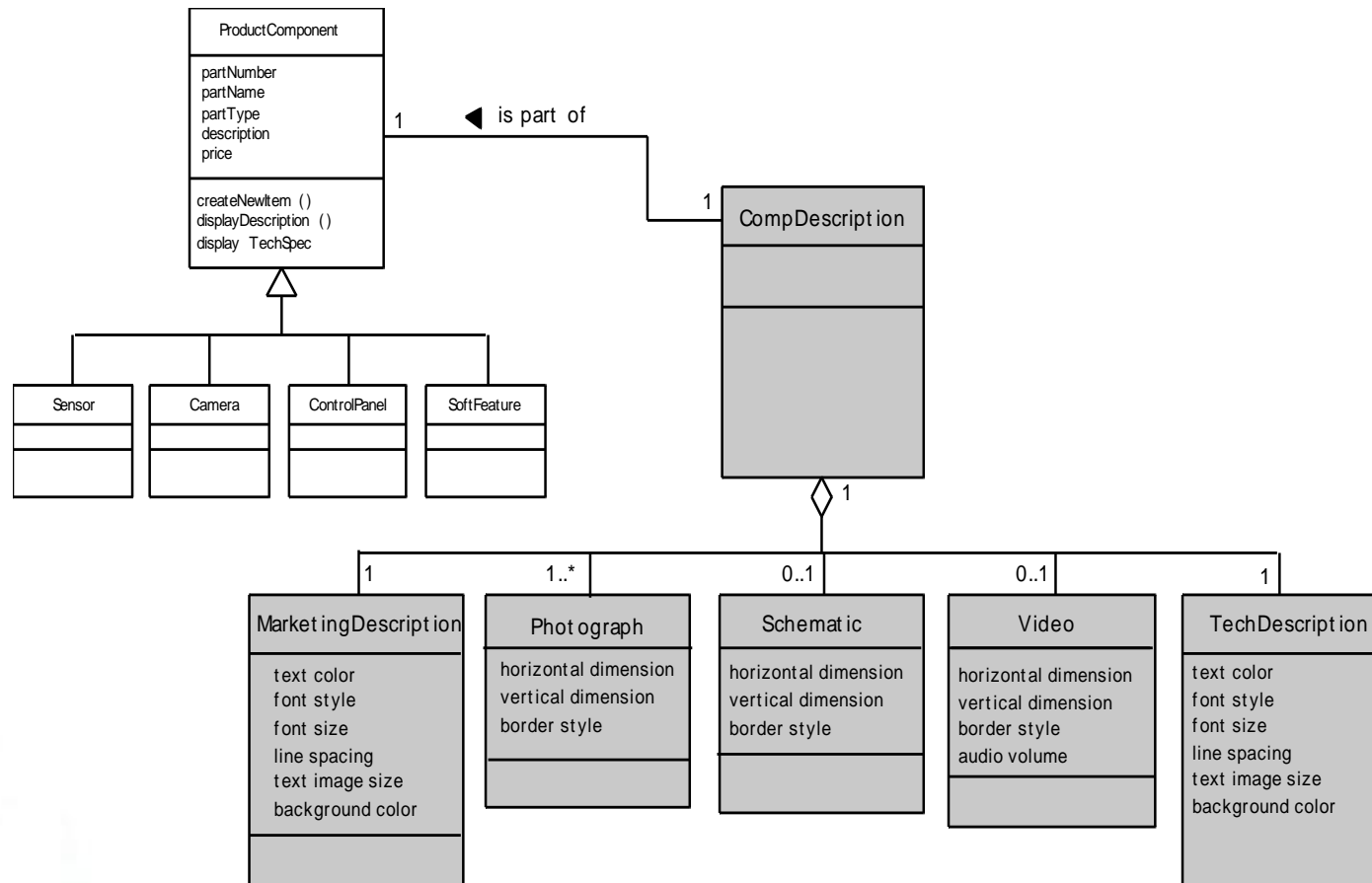
- **Content Design**

- 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





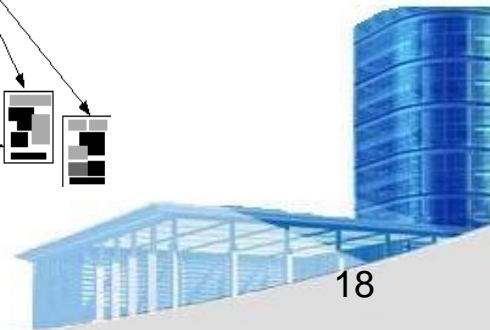
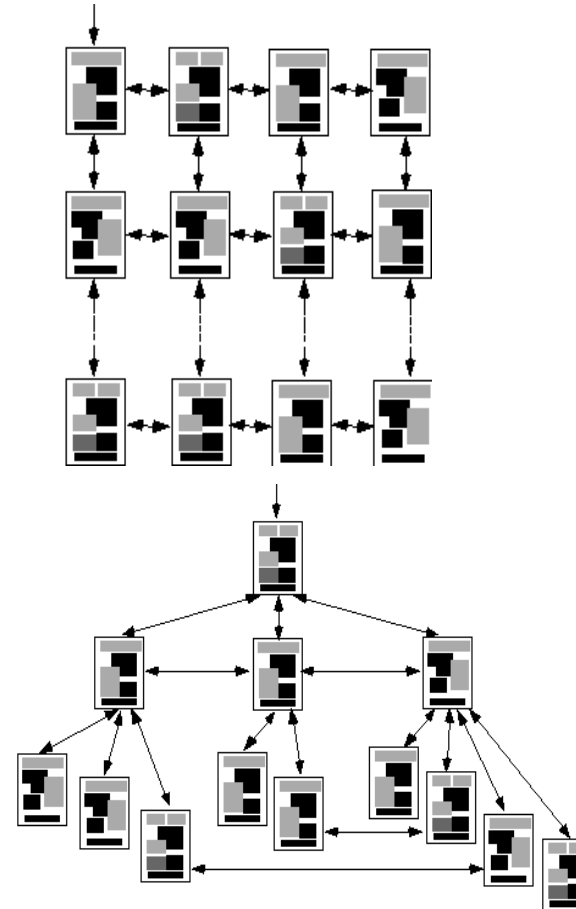
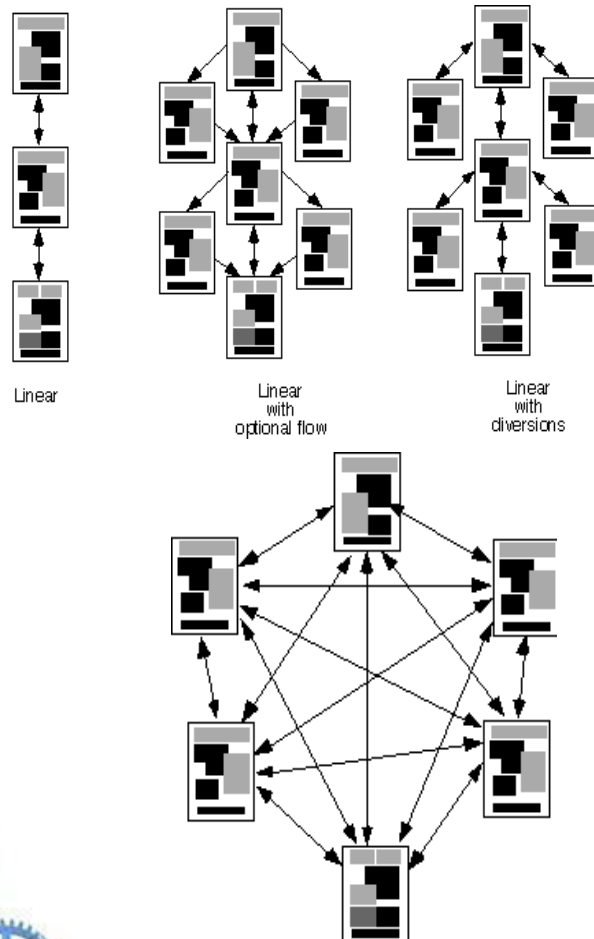
• Architecture Design

- *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 connote 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.
- Architecture design is conducted in parallel with interface design, aesthetic design and content design.





• Content Architecture





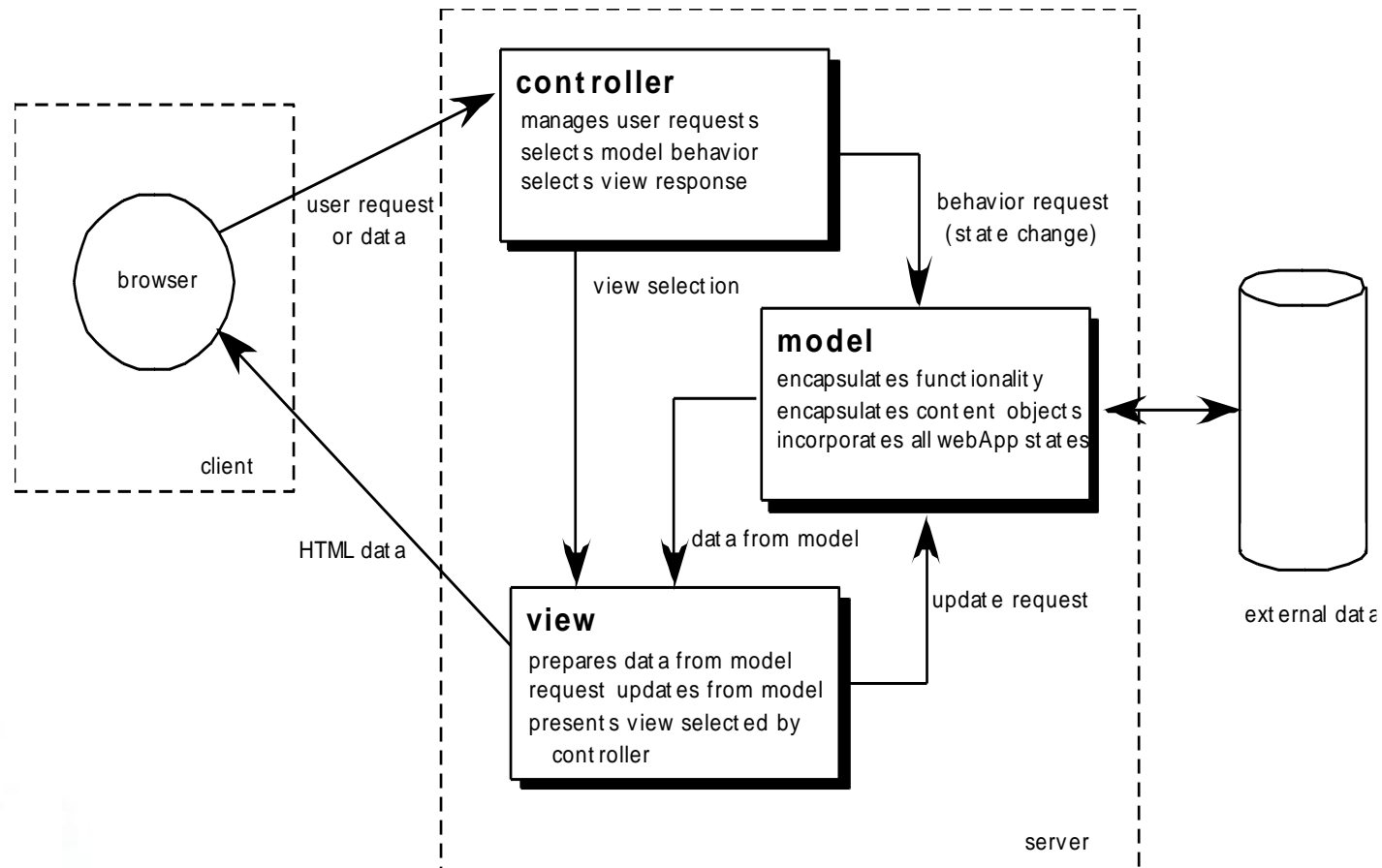
• MVC Architecture

- 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.





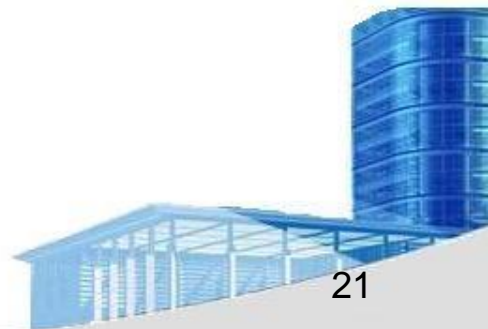
- MVC Architecture





- **Navigation Design**

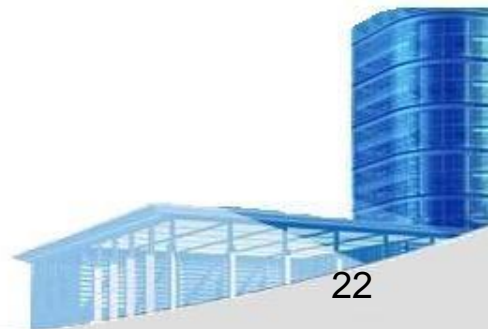
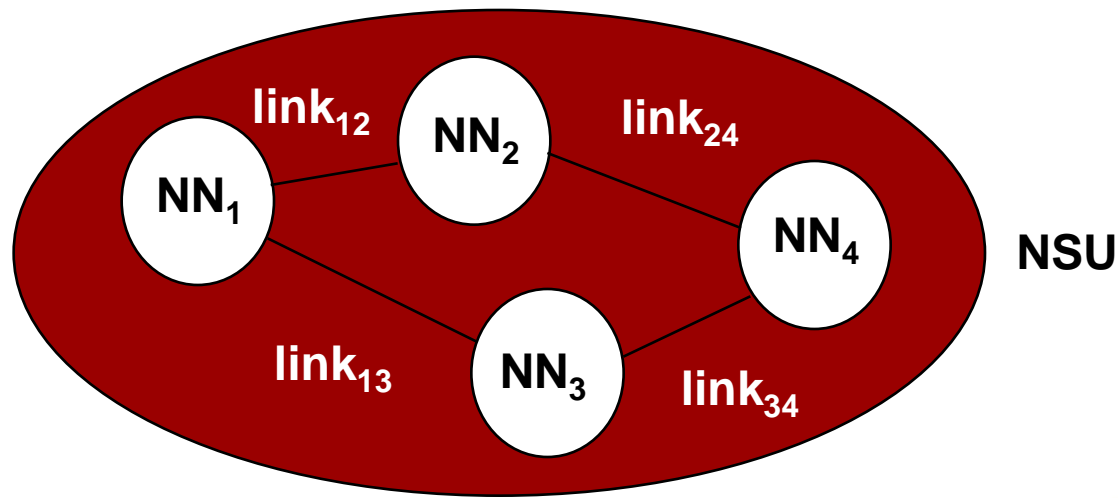
- Begins with a consideration of the user hierarchy and related use-cases
 - Each actor may use the WebApp somewhat differently and therefore have different navigation requirements
- As each user interacts with the WebApp, she encounters a series of *navigation semantic units* (NSUs)
 - NSU—“a set of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





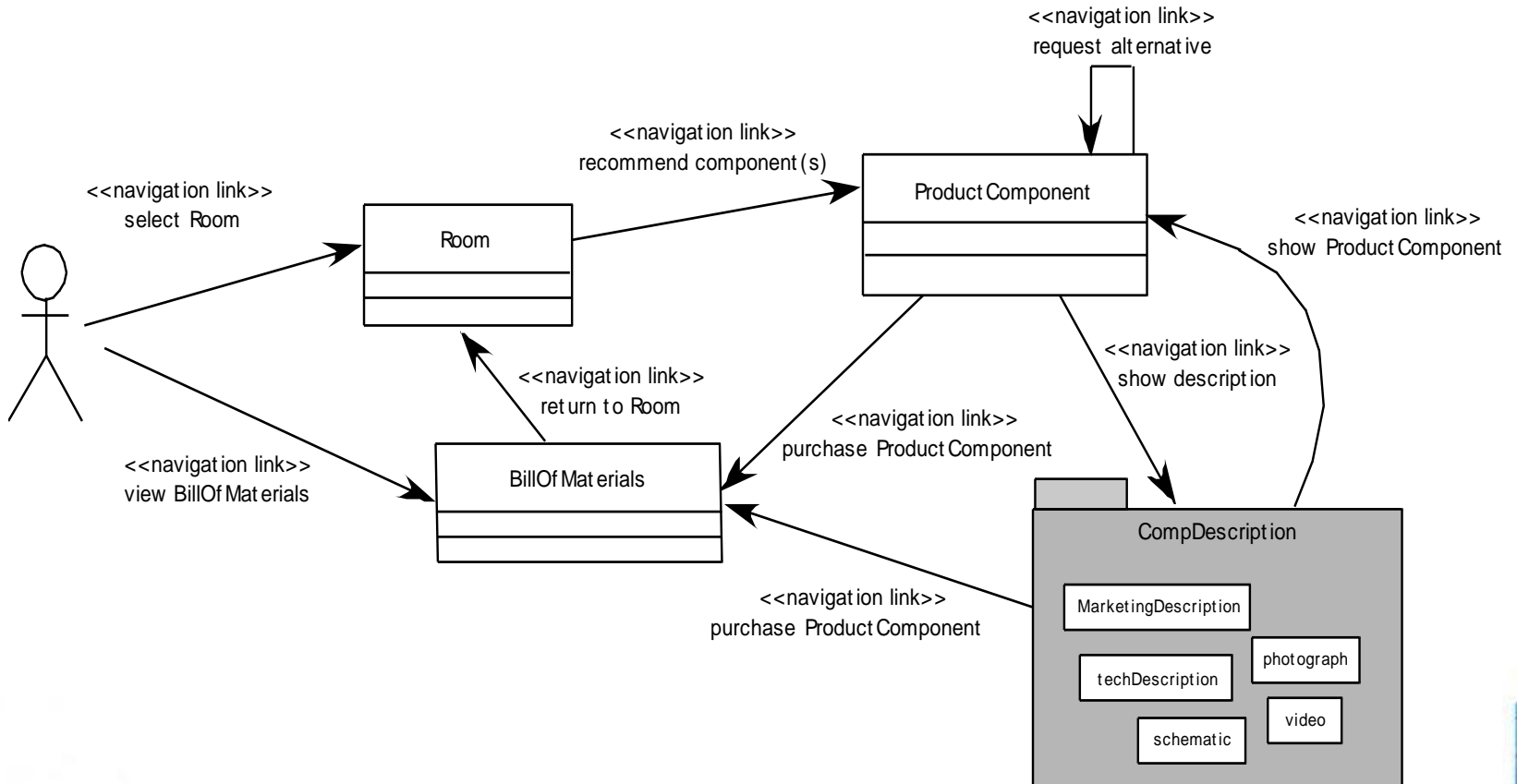
- **Navigation Semantic Units**

- *Navigation semantic unit*
 - **Ways of navigation (WoN)**—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 an NSU





• Navigation Syntax

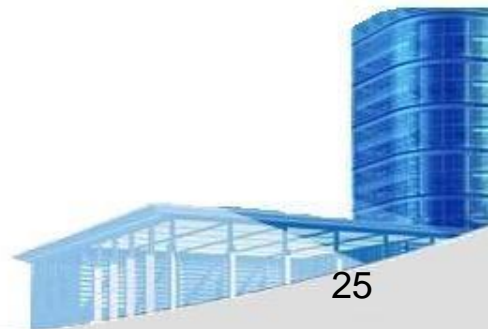
- *Individual navigation link*—text-based links, icons, buttons and switches, and graphical metaphors..
- *Horizontal navigation bar*—lists major content or functional categories in a bar containing appropriate links. In general, between 4 and 7 categories are listed.
- *Vertical navigation column*
 - 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**

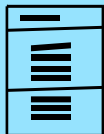
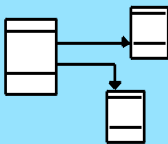
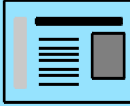

- WebApp components implement the following functionality
 - perform localized processing to generate content and navigation capability in a dynamic fashion
 - provide computation or data processing capability that are appropriate for the WebApp's business domain
 - provide sophisticated database query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

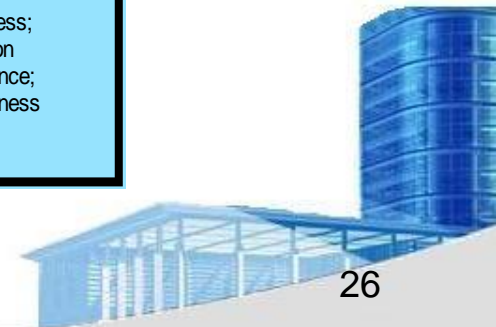




• OOHDM

• Object-Oriented Hypermedia Design Method (OOHDM)

	 conceptual design	 navigational design	 abstract interface design	 implementation design
work products	Classes, sub-systems, relationships, attributes	Nodes, links, access structures, navigational contexts, navigational transformations	Abstract interface objects, responses to external events, transformations	executable WebApp
design mechanisms	Classification, composition, aggregation, generalization specialization	Mapping between conceptual and navigation objects	Mapping between navigation and perceptible objects	Resource provided by target environment
design concerns	Modeling semantics of the application domain	Takes into account user profile and task. Emphasis on cognitive aspects.	Modeling perceptible objects, implementing chosen metaphors. Describe interface for navigational objects	Correctness; Application performance; completeness





• Conceptual Schema

