Functional Design

Week 9

Agenda (Lecture)

- Functional design

Agenda (Lab)

- Weekly progress report
- Homework/Lab assignments

Announcement

- Walk-through all lab assignment documents (1 – 9) for your group project
  - 3/28
  - Prepare the latest documents

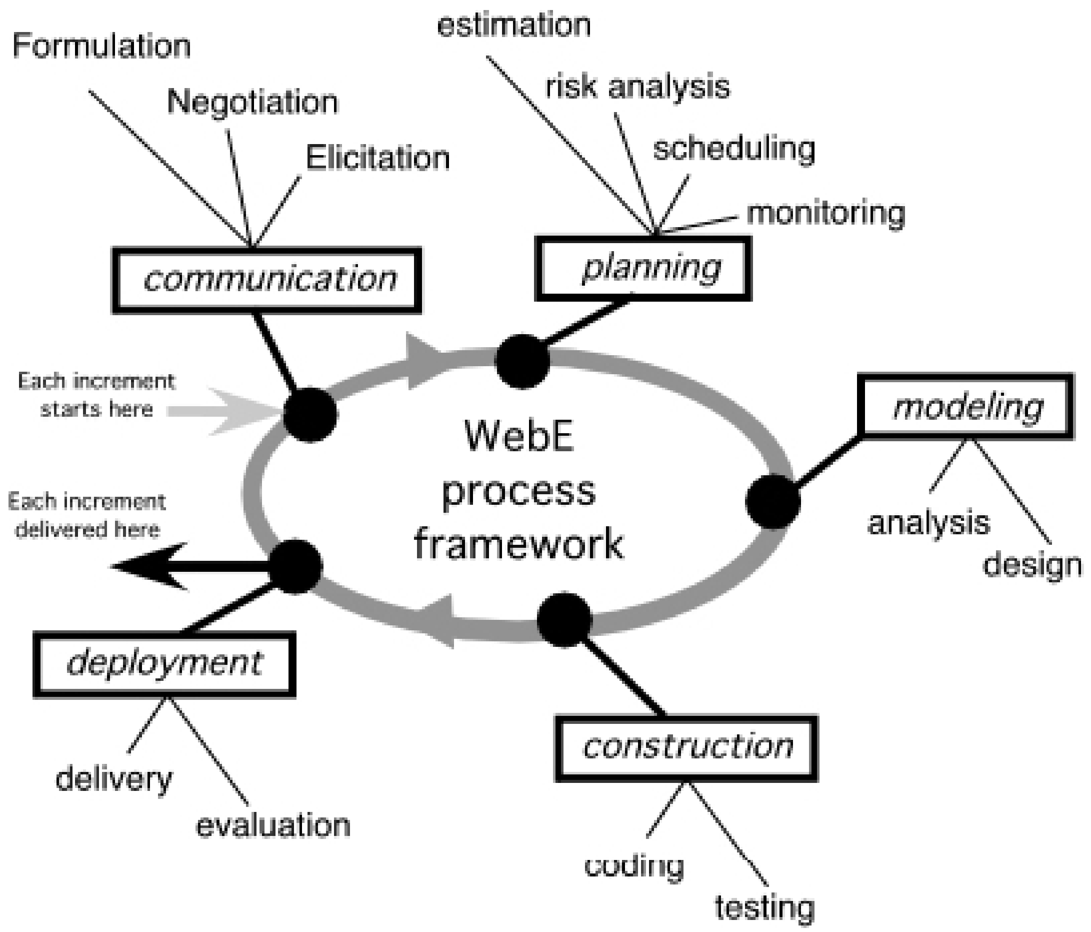
Team Lab Assignment #8

- Submit the first version of the functional design document for your group project
  - Make slides for presentation
- Due date
  - The beginning of the 3/28 lab session

Team Lab Assignment #9

- Submit an implementation plan document for your group project
  - Make slides for presentation
- Due date
  - The beginning of the 3/28 lab session

WebE Process Activities & Actions



Chapter 11 *Functional Design*

- Users of modern WebApps expect that robust **content** will be coupled with sophisticated **functionality**
- This functionality will allow them to magnify their understanding of content, characterize content in different ways, personalize their interaction, and provide added value to their website visit
- Functional design of WebApps is almost always component based and compartmentalized
- The designer must consider the substantial constraints imposed by the Web infrastructure—such as a distributed model (which complicates aspects like information handling and user responsiveness), security issues, and the limited interface model inherent in Web browsers

Functionality Categories

- Group 1: User-Level (External) Functionality. These categories include functionality that directly affects users' experience of the WebApp
  - Category 1A: User Interaction Support (e.g. highlighting a link on mouse-over)
  - Category 1B: User Information Support (e.g. presentation of live sensor readings)
  - Category 1C: User Task Support (e.g. dynamic checking and feedback on user-provided information)
- Group 2: Application-Level (Internal) Functionality. These categories relate to functionality that is necessary to support the WebApp, but which will only be visible to users as a second-order effect.
  - Category 2A: Application Interaction Support (e.g. logging of user navigation behaviours)
  - Category 2B: Application Information Support (e.g. database content maintenance)
  - Category 2C: Application Task Support (e.g. payment system)

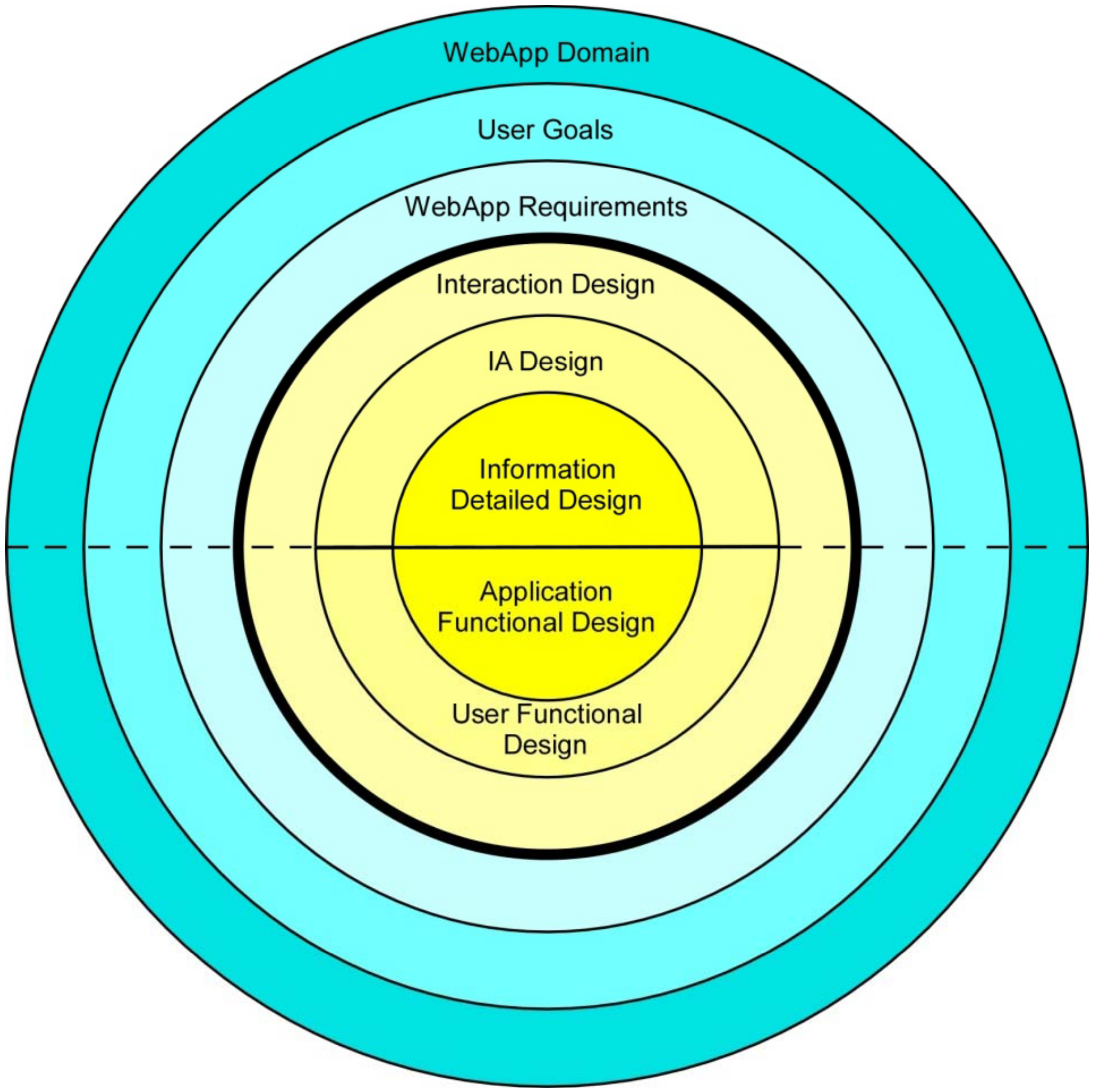
Functionality Examples

- Client-side interaction support
  - Drop-down menus
- Client-side information management
  - Image zooming and scrolling
- Server-side content handling
  - Live score updates
- Server-side management of large data sets
  - Searching a product
- Process and/or work flow support
  - A workflow

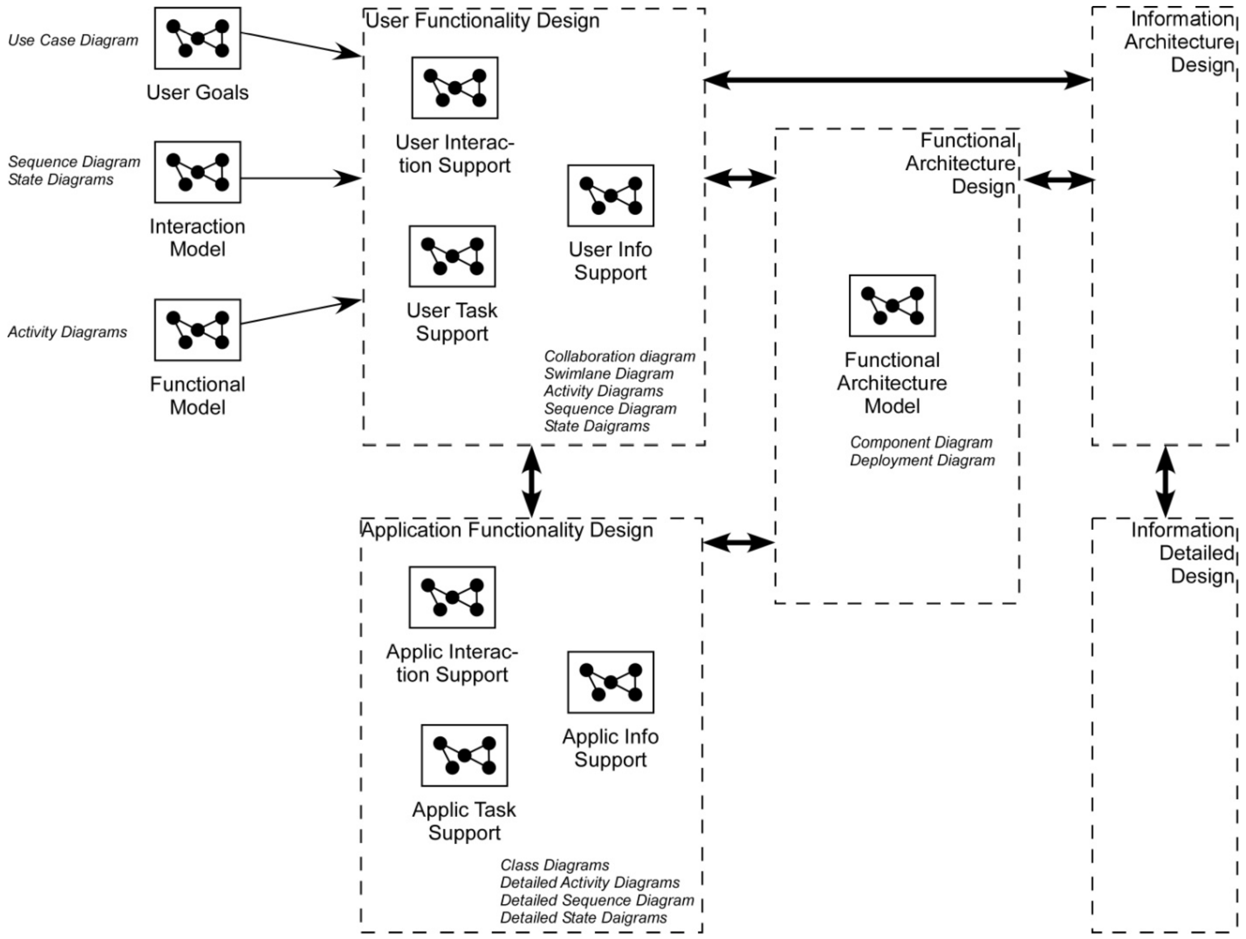
Functional Design

- Functional design is not a discrete task that is performed at just one point in the design process. Rather, it is interwoven with other design activities.
  - *User-level functionality* is the expression of the WebApp capabilities that support users in achieving their goals.
  - *Application-level functionality* represents a lower-level design of internal functionality that may not be directly visible to users
- Application-level functionality is more deeply embedded within the structure of the WebApp and will often emerge out of the progressive design of the user-level functionality

Functionality Levels and Design Tasks



Functional Design: Overview



Getting Started

- **SafeHomeAssured.com** has an interesting mix of information-focused and functionally focused components. In the initial communication activity (Chapter 4), we identified an initial set of informational and applicative goals for **SafeHomeAssured.com** reproduced in part here:
    - To provide users with requested product specs.
    - To provide tools that will enable users to represent the layout of a “space” (i.e., house, office/retail space) that is to be protected.
    - To make customized recommendations about security and monitoring products that can be used within the user space.
    - To enable users to obtain a quote for product cost.
    - To allow users to place an order for security hardware.
    - To allow users to control monitoring equipment (e.g., cameras, microphones) with their space.
    - To enable users to “sign up” for monitoring services.
    - To allow monitoring customers to query the monitoring database about their account activity.

Rough Functional Outline

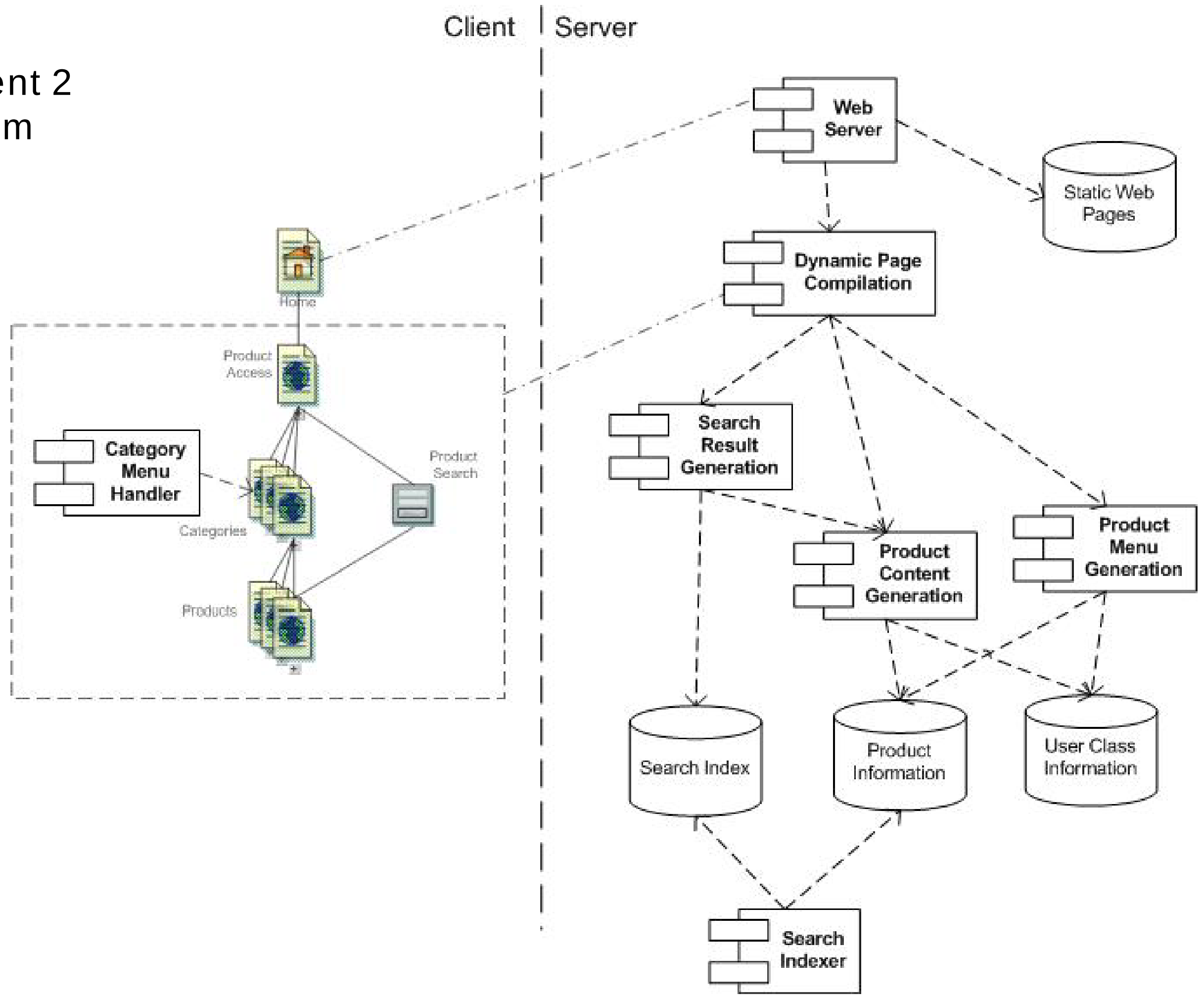
- These goals were then refined into the following list of functions to be performed:
  - Provide product quotation.
  - Process security system order.
  - Process user data.
  - Create user profile.
  - Draw user space layout.
  - Recommend security system for layout.
  - Process monitoring order.
  - Get and display account info.
  - Get and display monitoring info.
  - Customer service functions (to be defined later).
  - Tech support functions (to be defined later).
- Ultimately these functions are elaborated into a set of use cases that capture the key user information and functional interactions.

Functional Architecture

- A representation of the functional domain of the WebApp.
- Answers two key questions:
  - How do we partition the functionality into components that have clearly defined roles and interfaces?
  - Where does each functional component exist, and what does it interact with?
- Decomposes the WebApp into constituent functional components.

An Example

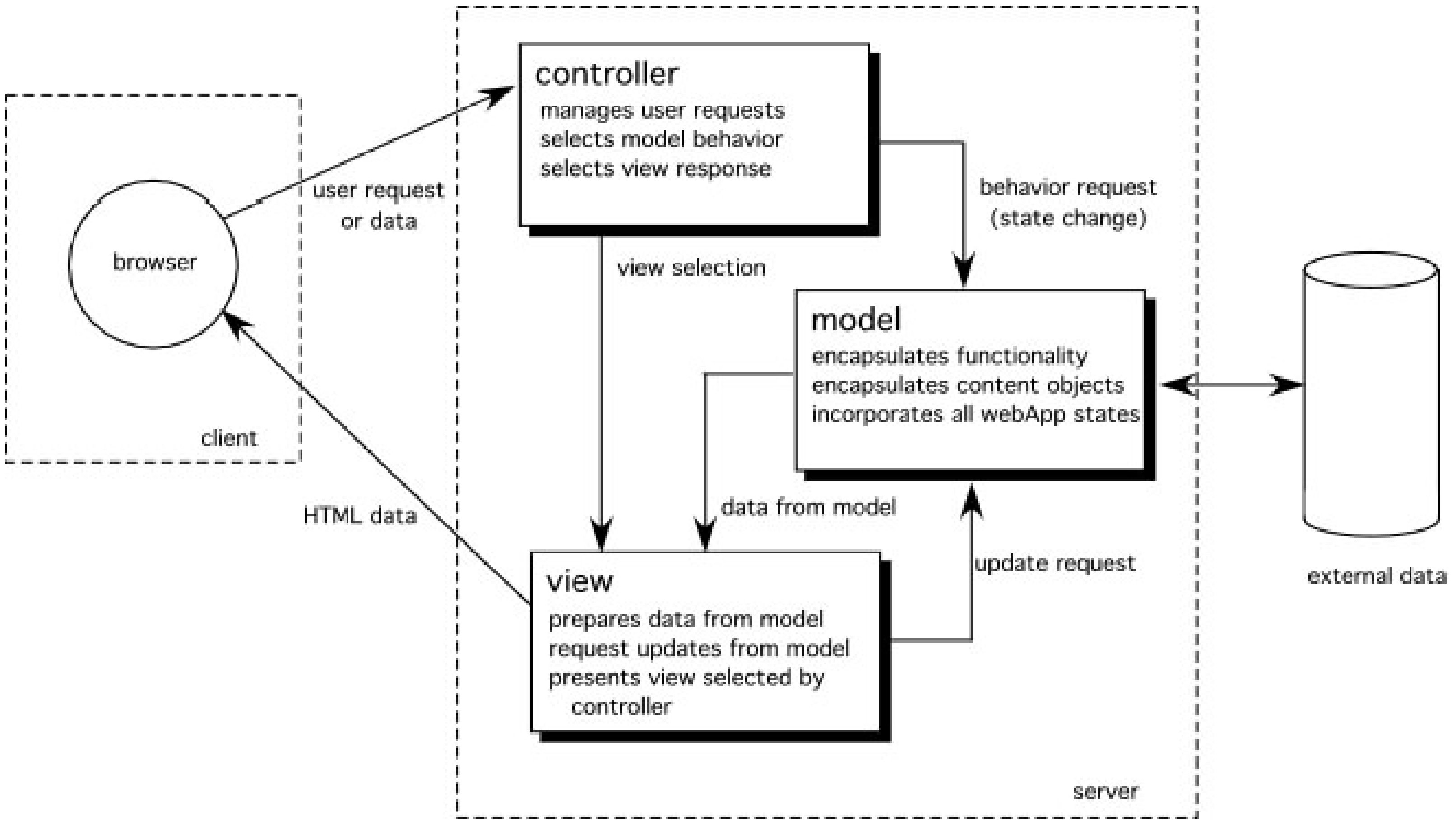
Preliminary functional architecture for Increment 2 of SafeHomeAssured.com



Developing the Architecture

- Consider both the WebApp analysis model (along with any specifications that accompany it) and the initial information architecture
- Decompose use cases into the following generic component categories:
    - *Information selection* (i.e., functionality associated with the identification and/or selection of information to be presented to the user).
    - *Information compilation* (i.e., functionality associated with merging information together into a composite to be presented to the user).
    - *Information processing* (i.e., the analysis or calculation of data).
    - *System interaction* (i.e., functionality associated with interactions with other systems external to the WebApp).
- Consider whether the specific scenario component should be invoked dynamically on user request, dynamically on event initiation, or manually.

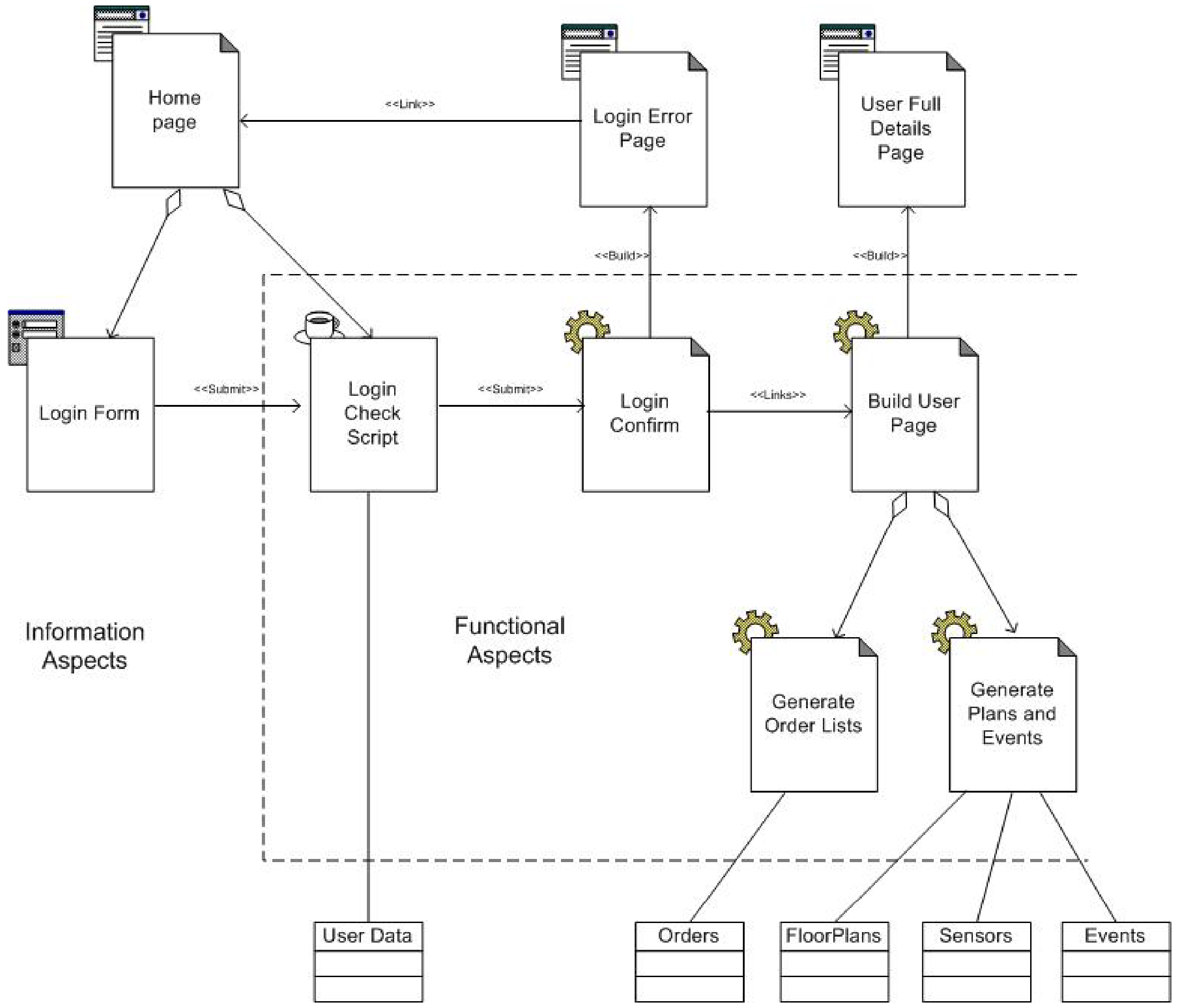
Architectural Patterns—MVC



Detailed Functional Design

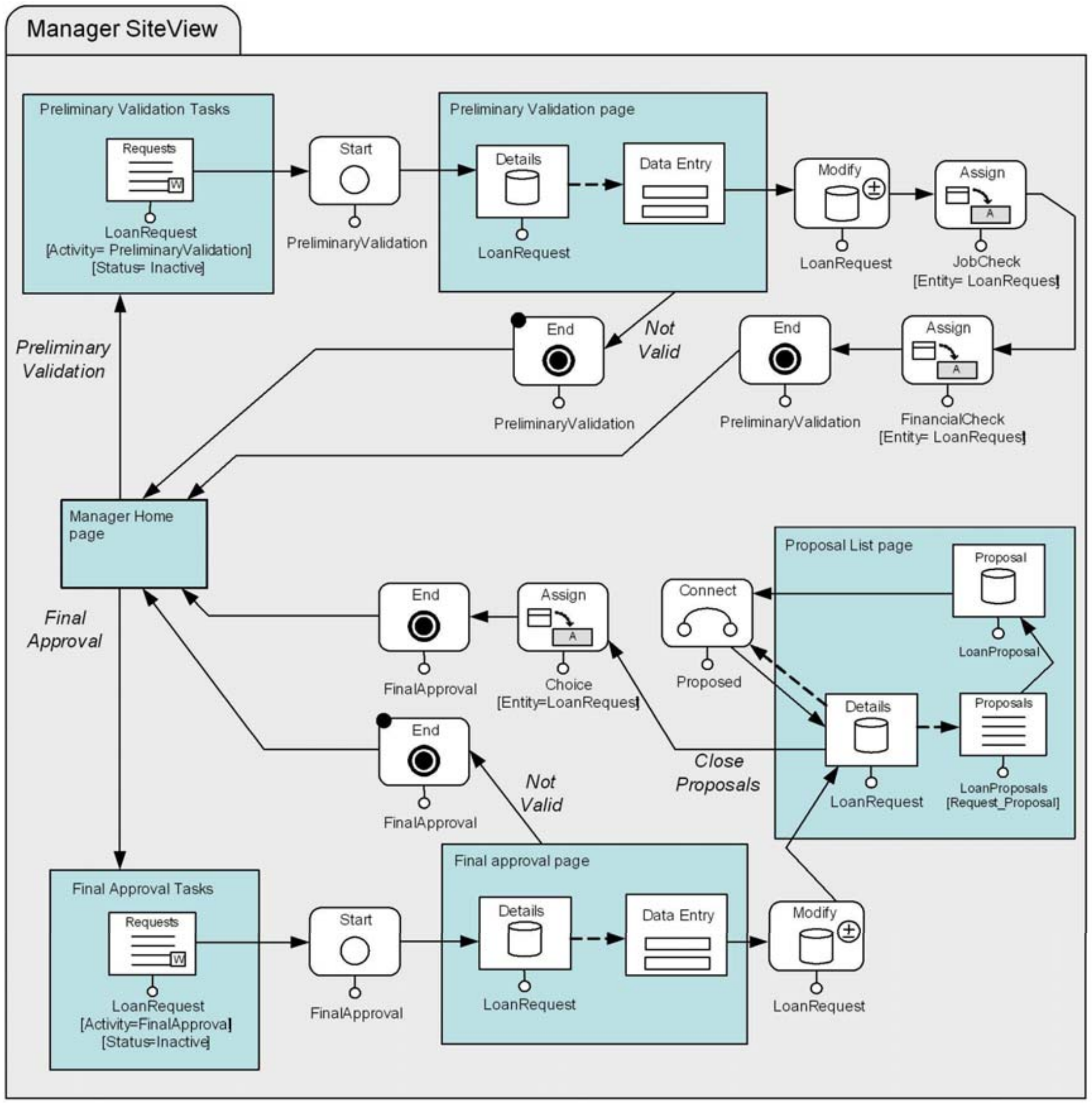
- Detailed functional modeling for WebApps is usually only carried out for those components that are extremely complex or extremely critical
- WAE establishes a set of extensions to UML that facilitate the modeling of WebApp low-level design
  - Particularly useful for connecting the information architecture to the functional components which generate the information views
- WebML has been adapted to model workflow-oriented applications.

WAE



Home page
<<Link>>
Login Error Page
User Full Details Page
<<Build>>
<<Build>>
Login Form
<<Submit>>
Login Check Script
<<Submit>>
Login Confirm
<<Links>>
Build User Page
Information Aspects
Functional Aspects
Generate Order Lists
Generate Plans and Events
User Data
Orders
FloorPlans
Sensors
Events

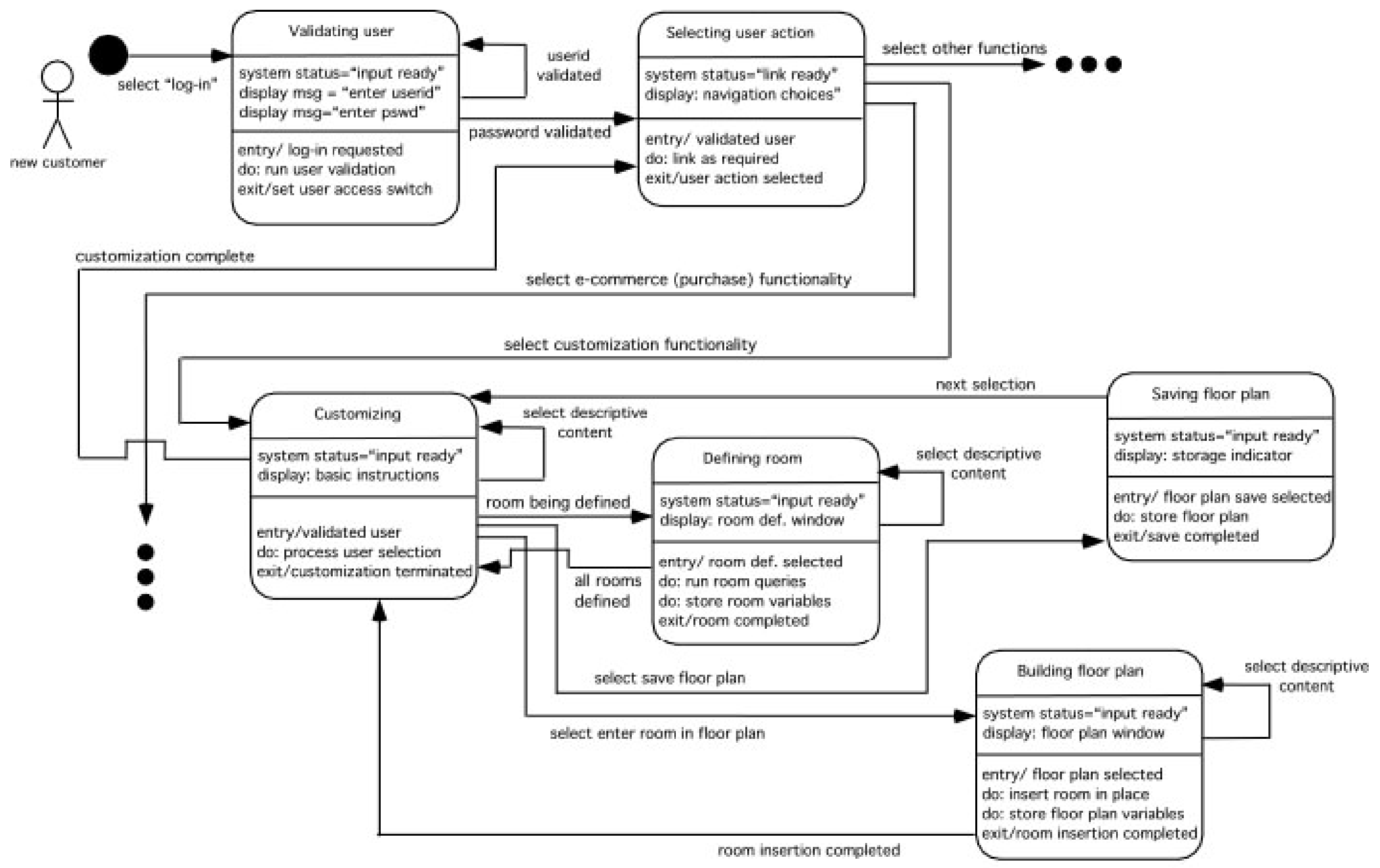
WebML


Manager SiteView
Preliminary Validation Tasks
Requests
LoanRequest [Activity= PreliminaryValidation] [Status= Inactive]
Start
PreliminaryValidation
Preliminary Validation page
Details
LoanRequest
Data Entry
Modify
LoanRequest
Assign
JobCheck [Entity= LoanRequest]
Not Valid
End
PreliminaryValidation
End
PreliminaryValidation
Assign
FinancialCheck [Entity= LoanRequest]
Preliminary Validation
Manager Home page
Final Approval
End
FinalApproval
Assign
Choice [Entity=LoanRequest]
Connect
Proposed
Proposal List page
Proposal
LoanProposal
Details
LoanRequest
Proposals
LoanProposals [Request_Proposal]
Close Proposals
End
FinalApproval
Not Valid
Final Approval Tasks
Requests
LoanRequest [Activity=FinalApproval] [Status=Inactive]
Start
FinalApproval
Final approval page
Details
LoanRequest
Data Entry
Modify
LoanRequest

State Modeling

- State modeling is necessary when:
  - You must accommodate interacting processes, particularly with multiple simultaneous users (or at least multiple users whose interactions with the Web servers are interleaved).
  - You must ensure that the state of the underlying information is correctly preserved when we have complex interacting processes.
- A *state* is an externally observable mode of behavior.
  - External *stimuli* cause *transitions* between states.
  - A *state model* represents the behavior of a WebApp by depicting its states and the events that cause the WebApp to change state.
  - A *state model* indicates what actions (e.g., process activation) are taken as a consequence of a particular event.
  - State models are created using *state diagrams*

State Model



new customer
select "log-in"
Validating user
system status="input ready"
display msg = "enter userid"
display msg="enter pswd"
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
select descriptive content
system status="input ready"
display: basic instructions
entry/validated user
do: process user selection
exit/customization terminated
room being defined
Defining room
select descriptive content
system status="input ready"
display: room def. window
entry/ room def. selected
do: run room queries
do: store room variables
exit/room completed
all rooms defined
select save floor plan
Building floor plan
select descriptive content
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 enter room in floor plan
room insertion completed