# Chapter 6: *The Modeling Activity*

- *"All models are wrong, but some models are useful"*, **George Box**
- *A* good model is *"wrong"* only in ways that are irrelevant to current project.
- We model our *perception of reality* so that we can understand and change it, but *our models of reality are not perfect*.
- Analysis modeling helps you to understand the nature of the problem being addressed and the "shape" of the WebApp that will allow you to address that problem
- Design modeling is about understanding the internal structure of the WebApp being developed and how this creates the shape of the WebApp that was identified by the analysis model.

# Modeling Languages

- A *modeling language* (ML) incorporates a set of notations, terms, and/or symbols, as well as the rules for establishing associations between them
- A *modeling language* often has a formally structured representation as well as a set of graphical elements
- Some MLs are general purpose (e.g., UML) and others are more specific (e.g., WebML)

# *Analysis Modeling*

- Analysis modeling helps you to understand the detailed requirements that will allow you to satisfy user needs
- Analysis models look at *content, interaction, function and behavior*, and the WebApp configuration
- To determine how much analysis modeling to do, examine the:
    - Size and complexity of the WebApp increment
    - Number of stakeholders (analysis can help to *identify conflicting requirements* coming from different sources)
    - Size of the WebE team
    - Degree to which members of the WebE team have worked together before (analysis can help develop a *common understanding* of the project)
    - Degree to which the organization's success is directly dependent on the success of the WebApp

# Initial questions

- Is the *functionality or content* of the WebApp increment *complex* enough to require a solution of more than a few pages, or some simple scripts?
- Do *complex relationships* exist among different context classes? If so, do these have an impact on how the WebApp is implemented?
- Does the WebApp increment produce *information* that will be *critical to the user?*
- Does the WebApp increment deliver *functions that are not well understood currently*? Are said functions *algorithmically complex*?

# Initial questions, ctd.

- Does the WebApp increment have a *significant impact on already deployed increments, or on increments that will be developed later?*
- Is the WebApp *business-critical*?
- Will the WebApp increment have a *possible impact on the other operations of the business* or on the relationship(s) that the business organisation has with its stakeholders?
- Is there any possibility that *stakeholders are unsure of the requirements* for the WebApp increment?
- Are there *more than one or two user classes*? If so will they use the WebApp in a substantially different way?

# Initial questions, ctd.

**Note:**

If answer to any of preceding questions is ***"yes"***, we should consider building an *analysis model.*

*Focus modeling on those aspects where there is a lack of clarity or where there is potential for confusion*.

# Typical Analysis Tasks

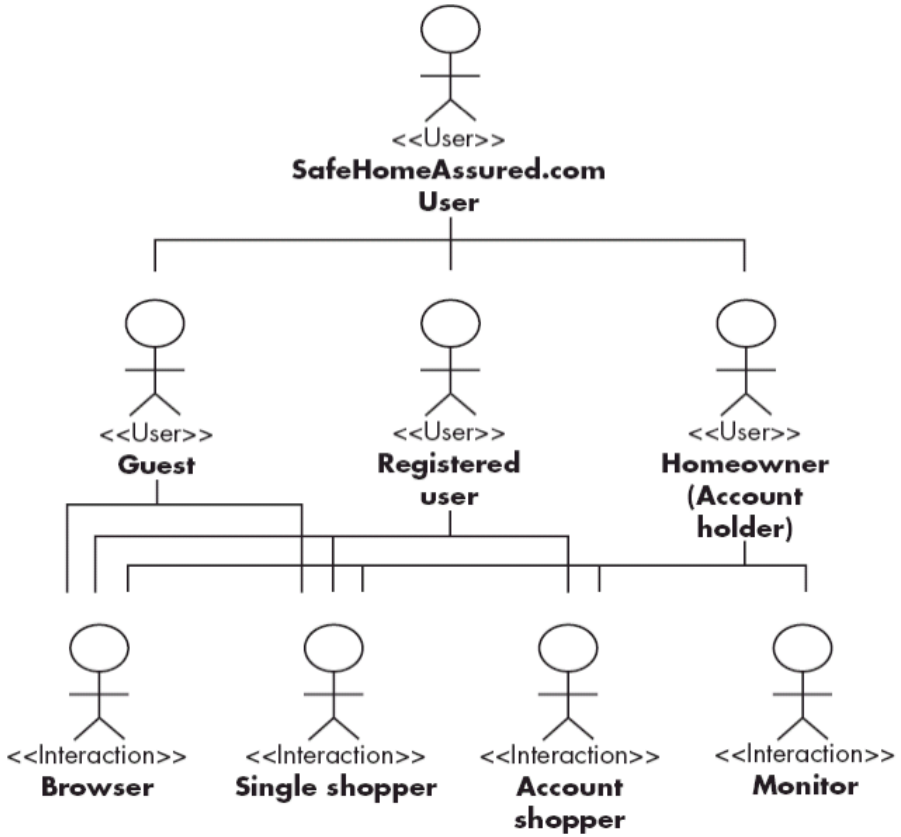
- Determine if requirements model is required.
- Represent WebApp content.
- Identify content relationships.
- Refine and extend usage scenarios.
- Review usage scenarios.
- Create interaction model for complex scenarios.
- Refine interface requirements.
- Define constraints and performance requirements.
- Represent functional requirements.
- Represent navigational requirements.

# Analysis Outputs

- **Interaction model.** Describes the manner in which users interact with the WebApp.
- **Information/Content model.** Identifies the full spectrum of content to be provided by the WebApp. Content includes text, graphics and images, and video and audio data.
- **Functional model.** Defines the operations that will be applied to WebApp content and describes other processing functions that are independent of content but necessary to the end user.
- **Configuration model.** Describes the environment and infrastructure in which the WebApp resides.

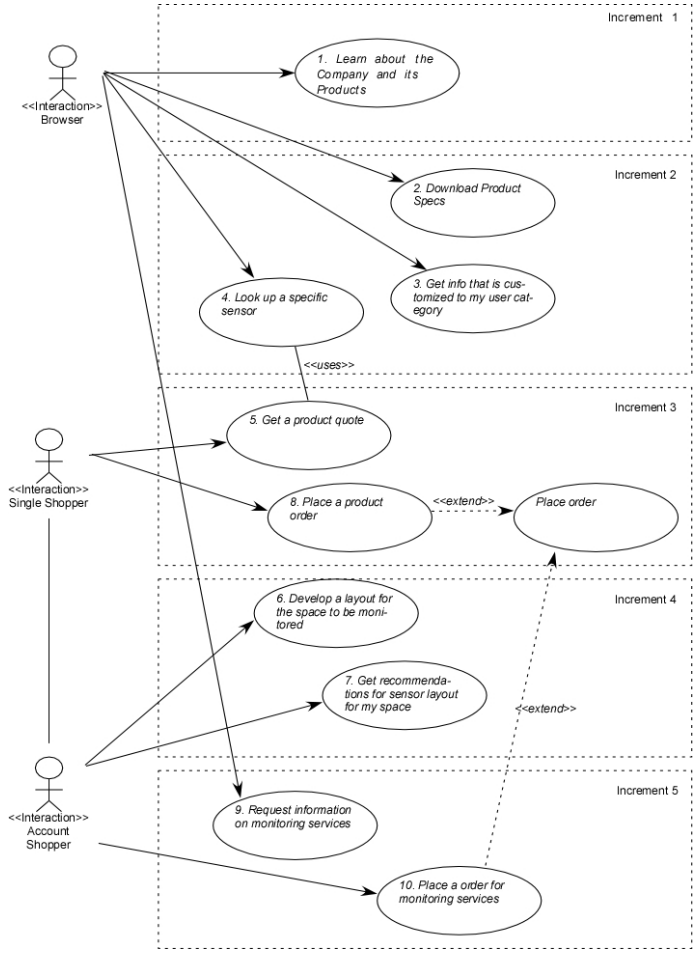
# Understanding Users

- Crucial to understand your users!
- For each user class:
  - What is the user's overall objective?
  - What is the user's background?
  - How will the user arrive at the WebApp?
  - What characteristics does the user like and dislike?



# Revisiting Use Cases

- Analyse and elaborate where necessary
  - Find gaps, missing details
- Identify overlaps and possible optimizations
  - Allows design simplification
  - E.g. often “view” task can be seen as a specialization of an “edit” task.



# The Content Model

- Contains *structural elements* that provide an important view of content requirements for a WebApp.
- Structural elements encompass both *content objects,* (e.g. text, photographs, video, audio), that are presented as part of WebApp, and *analysis classes* – user visible entities that are created or manipulated as a user interacts with the WebApp.

# Manifestation

- Analysis classes manifest themselves in one of the following ways:
  - *External entities* (e.g., other systems, databases, people) that produce or consume information to be used by the WebApp
  - *Things* (e.g., reports, displays, video images) that are part of the information domain for the problem
  - *Occurrences or events* (e.g., a quote or an order) that occur within the context of a user's interaction with a WebApp
  - *Roles* (e.g., retail purchasers, customer support, salesperson) played by people who interact with the WebApp
  - *Organizational units* (e.g., division, group, team) that are relevant to an application
  - *Places* (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the WebApp
  - *Structures* (e.g., sensors, monitoring devices) that define a class of objects or related classes of objects

- An *analysis class* encompasses:
  1. *Attributes* that define it,
  2. *Operations* that define required behaviour,
  3. *Collaborations* that allow class to communicate with others.

# Attributes

- Clarify *what* a class means in the context of the problem-space.
- Consider the following helpful attribute-identifying question that should be answered for each class:

  *What data items, (elementary and/or composite), fully define this class in the context of the problem at hand?*

# Operations

- Consider the following operation-categories:
  1. Operations that manipulate content, e.g. add/ delete/reformat/select.
  2. Operations that perform a computation.
  3. Operations that inquire about a the state of an object.
  4. Operations that monitor an object for the occurrence of a controlling event.

# Collaborations

- Collaborations identify relationships between classes.
- When classes collaborate, to achieve some requirement, they can be organised into an architectural element, e.g. a subsystem of the WebApp during design.

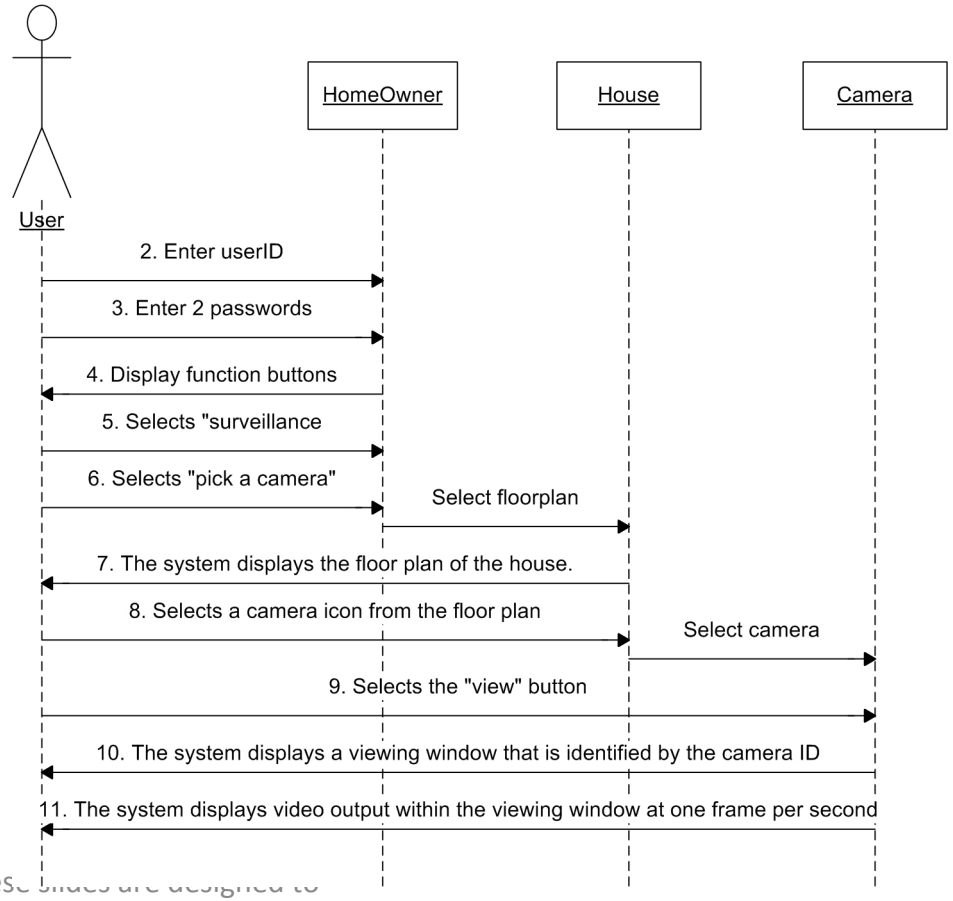
- **Rem:**

  A class may use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility. Alternatively, class may fulfill a responsibility by collaborating with other classes.

# The Interaction Model

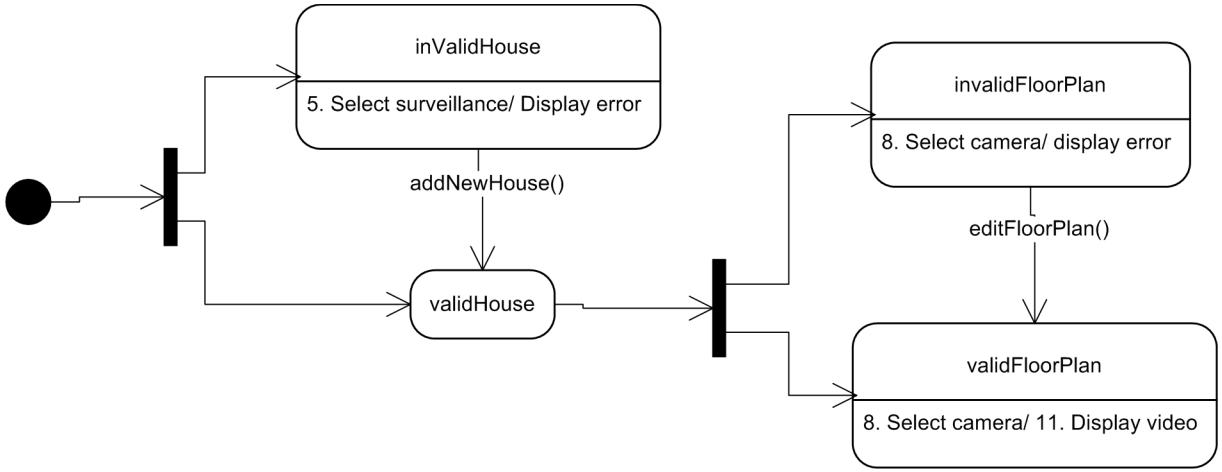
- Can be composed of one or more of:
  - Use cases
  - Sequence diagrams
  - State diagrams
  - User interface prototypes
- In many instances, *a set of use cases* is sufficient to describe the interaction at an analysis level (further refinement and detail will be introduced during design)
- However, when the sequence of interaction is complex and involves multiple analysis classes or many tasks, it is sometimes worthwhile to depict it using a more rigorous diagrammatic form.

# Sequence Diagram




UML *sequence diagrams* describe how user actions (dynamic elements) collaborate with analysis classes (the structural elements of a system).

# State Diagram




- UML *state diagrams* describe dynamic behavior of the WebApp as an interaction occurs.
- State diagrams are most useful when a user interaction triggers a change in the state of the WebApp—and hence changes the way in which it might react to a user.

- *Use cases, Sequence diagrams, and State diagrams* all present related information.

  **Q.** Are all three necessary?

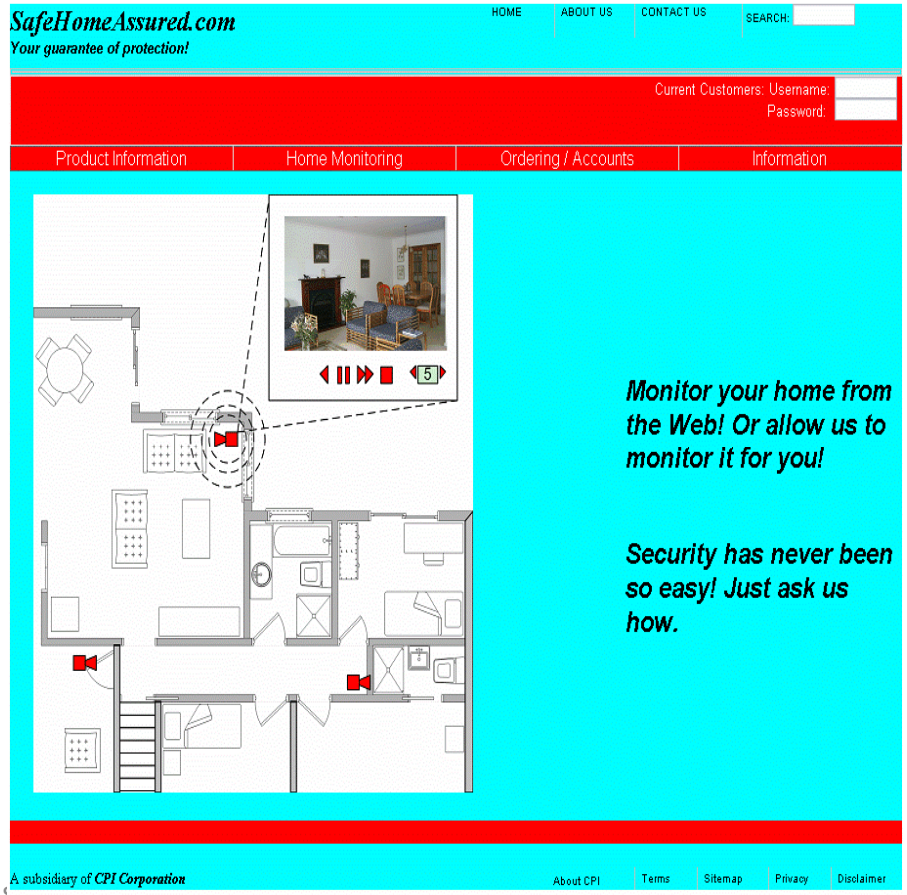
  **A.** It depends!

  1. Use cases may be sufficient, especially if *exceptions & alternatives* have been thoroughly documented.
  2. Sequence diagrams provide a more procedural / dynamic view / dimension.
  3. State diagram view is more *behavioural* & contains information about potential navigation pathways that is not provided by use cases or sequence diagrams.

- Large, complex WebApps can benefit from an interaction model that encompasses all three representations as *omissions* or *inconsistencies* that might escape discovery in one dimension become obvious when a second / third dimension is examined.

- Consider the following key factors when deciding on the interaction model elements:
  1. Complexity of overall increment.
  2. Consequence of an error for a user.
  3. Likely difficulty associated with rectifying an error later in the WebE process.

# Active Interface Prototype



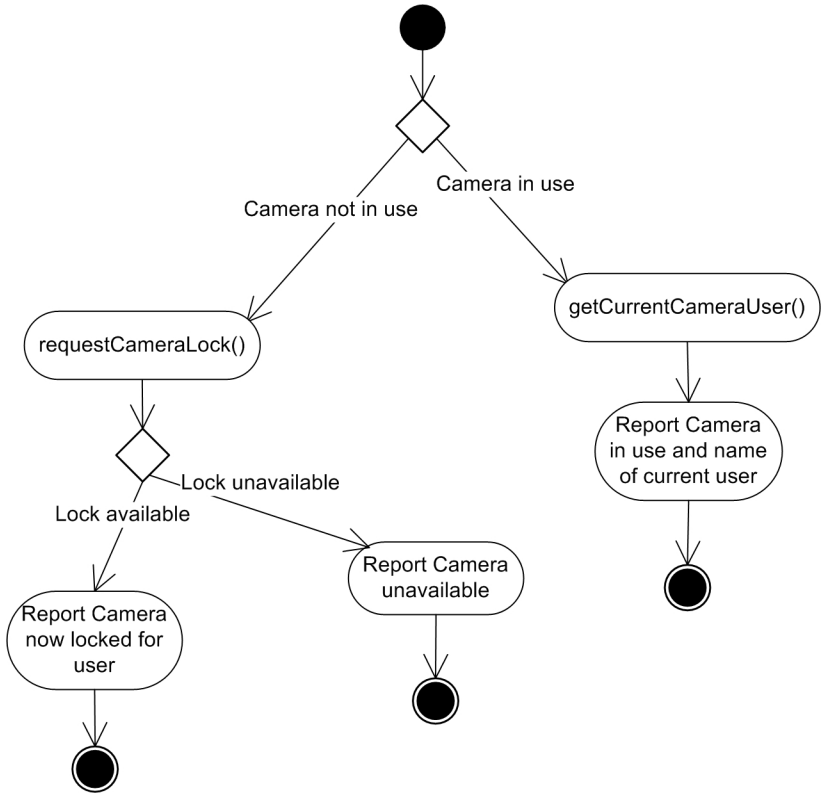

- A prototype shows the layout of the user interface, the content, interaction mechanisms and overall aesthetic
- Supports validation with the client of the requirements and analysis

# The Functional Model

- Addresses two processing elements of the WebApp, each representing a different level of procedural abstraction:
  - *user-observable functionality* that is delivered by the WebApp to end users, and
  - the *operations contained within analysis classes* that implement behaviors associated with the class.
- The UML *activity diagram* can be used to represent processing details

# Activity Diagram

- Illustrates the processing flow and logical decisions within the flow.
  - The construction details indicate how these operations are invoked, and the interface details for each operation are not considered until WebApp design commences.



# The Configuration Model

- Among the many configuration issues that should be addressed are:
  - Server hardware and operating system environments
  - Interoperability considerations on the server side (e.g., large database access, other IT applications, specialized communication protocols)
  - On the client side:
    - Local OS
    - Browser software
    - Client hardware variations

# Relationship-Navigation Analysis

- *Relationship-navigation analysis* (RNA) provides a series of analysis steps that strive to identify relationships among the elements uncovered as part of the creation of the analysis model
- The RNA approach is organized into five 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

# Establishing Relationships between Content Objects & Functionality

- Consider the following list of questions that can help assess the relationships among *elements* that have been identified within the analysis model:
  1. Is the element a member of a broader category of elements?
  2. What attributes or parameters have been identified for the element?
  3. Does descriptive information about the element already exist? If so, where is it?
  4. Does the element appear in different locations within the WebApp? If so, where?
  5. Is the element composed of other smaller elements? If so, what are they?

# Establishing Relationships between Content Objects & Functionality

6. Is the element a member of a larger collection of elements? If so, what is it, and what is its structure?
7. Is the element described by an analysis class?
8. Are other elements similar to the element being considered? If so, is it possible that they could be combined into one element?
9. Is the element used in a specific ordering of other elements? Does its appearance depend on other elements?
10. Does another element always follow the appearance of the element being considered?
11. What pre- and post-conditions must be met for the element to be used?

# Establishing Relationships between Content Objects & Functionality

12. Do specific user categories use the element? Do different user categories use the element differently? If so, how?
13. Can the element be associated with a specific goal or objective? - with a specific WebApp requirement?
14. Does this element always appear at the same time as other element appear? If so, what are they?
15. Does this element always appear in the same place, e.g. same location of screen / page, as other elements? If so, what are they?

# Analysing Navigational Requirements

- Once relationships have been established between analysis model *elements*, we need to consider the *requirements* that dictate how each user-category will *navigate* from one element to another.
- During analysis modelling, we need to consider the *overall* navigation requirements.
- The *mechanics* of navigation are defined as part of design.

# Analysing Navigational Requirements

- Consider the following questions that should be addressed:

1. Should certain elements be easier to reach , i.e. fewer navigation steps, than others? What is the priority for presentation?
2. Should certain elements be emphasised to force users to navigate in their direction?
3. How should navigation errors be handled?
4. Should navigation to related groups of elements be given priority over navigation to a specific element?
5. Should navigation be accomplished via links, search-based access, or some other means?

# Analysing Navigational Requirements

6. Should certain elements be presented to users based on the context of previous navigation actions?
7. Should a navigation log be maintained for users?
8. Should a full navigation map or menu, as opposed to a single *"back"* link or *directed pointer*, be available at every point in a user's interaction?
9. Should navigation design be driven by the most commonly expected user behaviours, or by the perceived importance of the defined WebApp element?

# Analysing Navigational Requirements

10. Can users “store” their previous navigation through the WebApp to expedite future usage?
11. For which user-category should optimal navigation be designed?
12. How should links, external to the Webapp, be handled – overlaying the existing browser window / as a new browser window / as a separate frame?