ITA

Web Engineering: A Practitioner's Approach

by Roger S. Pressman and David Lowe

Chapter 1

- *Web-Based Systems*

The Web

- An indispensable technology
  - In every aspect of modern living - buy products (e-commerce), meet people (online dating), understand the world (portals), acquire our news (online media), voice our opinions (blogs), entertain ourselves (everything from music downloads to online casinos), and go to school (online learning).
- A transformative technology
  - Changes the way we do things
  - Changes the way we acquire and disseminate information
- An evolving technology
- Bottom line—high impact on everyone in the modern world

WebApps

- The term *Web application* (WebApp) encompasses:
  - Everything from **a simple Web page** that might help a consumer to compute an automobile lease payment to **a comprehensive website** that provides complete travel services for business people and vacationers.
- Category:
  - Complete websites
  - Specialized functionality within websites
  - Information-processing applications that **reside on the Internet or on an Intranet or Extranet**

WebApps

- Means HTML, Java, XML, or any of the countless technologies that must be understood to build successful Web-based systems/applications (WebApps)
- WebApps can be pivotal to the success of all businesses and organizations

Web-Based Systems

- In the early days, the Web systems built using informality, urgency, intuition, and art
  - *Informality* leads to an easy work environment—one in which you can do your own thing.
  - *Urgency* leads to action and rapid decision making.
  - *Intuition* is an intangible quality that enables you to "feel" your way through complex situations.
  - *Art* leads to aesthetic form and function—to something that pleases those who encounter it.
- Problem is—this approach can and often does lead to problems

Web-Based Systems

- As WebApps become larger and more complex,
  - Informality remains, but some degree of requirements gathering and planning are necessary
  - Urgency remains, but it must be tempered by a recognition that decisions may have broad consequences
  - Intuition remains, but it must be augmented by proven management and technical patterns
  - Art remains, but it must be complemented with solid design
- Bottom line—we must adapt the old-school approach to the realities of a Web 2.0 world….and now Web 3.0 world

WebApp Attributes

- Data driven
- Performance – Not to wait too long for serverside processing, for client-side formatting and display
- Continuous evolution
- Immediacy - exhibit a time-to-market
- Network intensiveness - diverse community of clients on net
- Concurrency - Large number of users may access at one time
- Unpredictable load- No. of users of may vary from day to day.
- Availability
- Content sensitive- simple, yet meaningful for nontechnical user
- Security
- Aesthetics- appeal of a WebApp's look and feel

WebApp Types

- Informational- readonly content with simple navigation and links
- Download - informational and *download capability*
- Customizable – different for each different user
- Interaction – chat room
- User input – take input from user in form for automization
- Transaction-oriented – automated based on user request
- Service-oriented
- Portals - providing website links having answers for customer
- Database access
- Data warehousing

(see http://digitalenterprise.org/models/models.html for examples)

Web Apps

- Why Web Applications/Web based systems fail?
- Because many built in an ad hoc manner
  - With little regard to the
    - Fundamental principles of problem analysis
    - Effective design
    - Solid testing
    - Change management

And What's the Solution?

Web Engineering

Web Engineering

- Goal is to build WebApps or Web based system that satisfy users' needs and provide real benefit to their clients' businesses or organizations.

- *i.e. To build industry-quality WebApps*

Chapter 2: *Web Engineering*

- Definition
  - *An agile, yet disciplined framework for building industry-quality WebApps*

Agile Approach

- Business strategies and rules change rapidly
- Management demands near-instantaneous responsiveness (even when such **demands are completely unreasonable**)
- Stakeholders often don't understand the consequences of the Web and **keep changing their mind** even as they **demand rapid delivery**

An agile approach helps to manage with this fluidity and uncertainty

Agile Approach

- Able to appropriately respond to changes, Change is to
  - The software being built
  - The team members
  - New technology
  - Of all kinds that may have an impact on the product they build or the project that creates the product
- Support for changes should be built-in everything we do in software
- An agile team recognizes that software is developed by individuals working in teams and that the skills of these people, their ability to collaborate is at the core for the success of the project

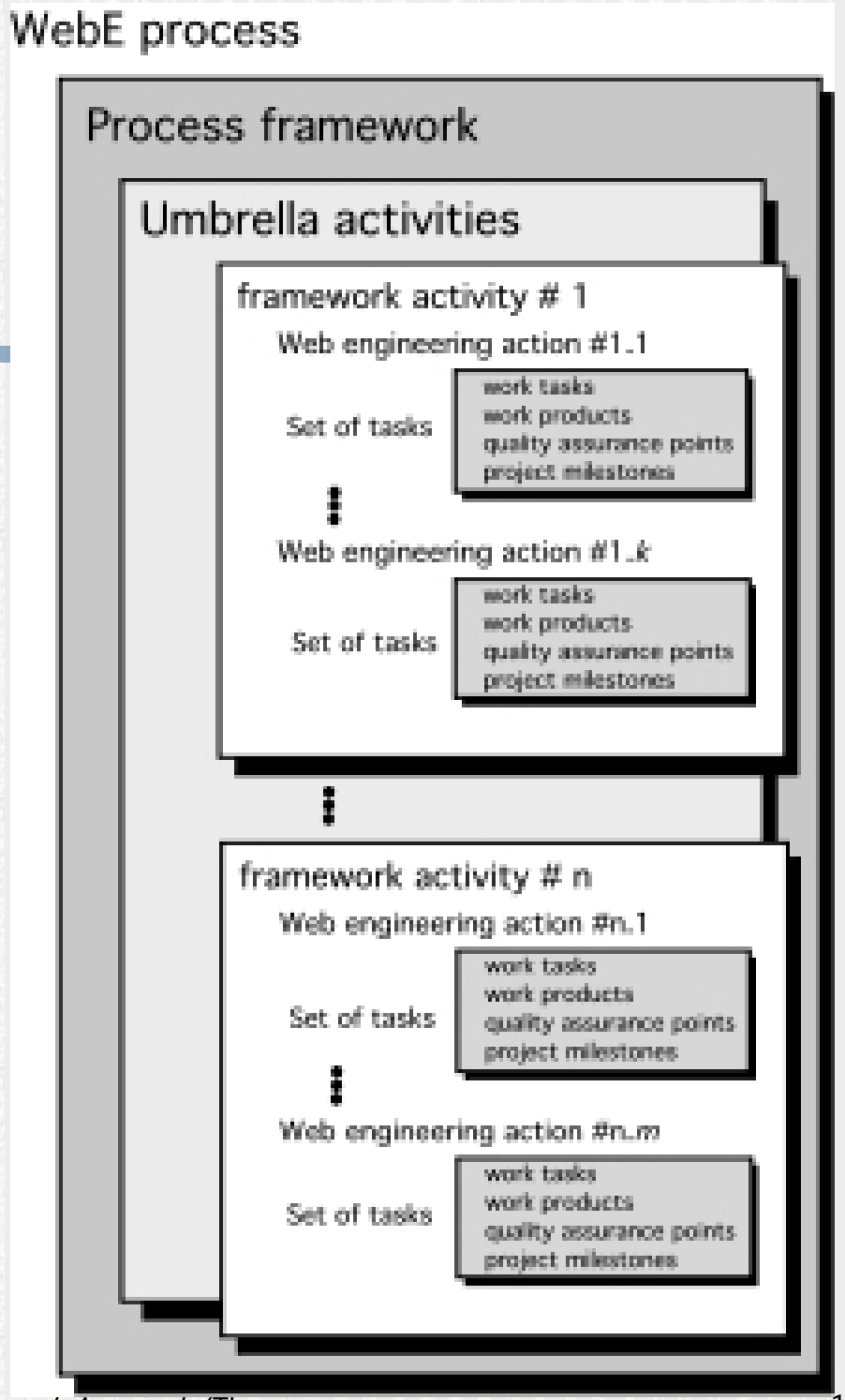
What is an Agile Process?

- Agile Web engineering combines a philosophy and a set of development guidelines. The philosophy encourages:
  - Customer satisfaction
  - Early incremental delivery of the WebApp
  - Small, highly motivated project teams
  - Informal methods
  - Minimal work products
  - Overall development simplicity
- An agile process stresses delivery over analysis and design and also active and continuous communication between developers and customers.

What is a WebE Framework?

- Framework
  - A set of activities that will *always* be performed for *every* Web Engineering project – though the nature of the activities might vary to suit the project
  - Each framework activity is composed of a set of actions
  - Actions encompass
    - Work tasks
    - Work products
    - Quality assurance points
    - Project milestones
  - A framework also has a set of “umbrella activities”

A Generic Framework



The WebE Framework: Activities

- Communication
- Planning
- Modeling
- Construction
- Deployment

The WebE Framework: Activities

- **Communication.** Involves heavy interaction and collaboration with the customer (and other stakeholders) and encompasses requirements gathering and other related activities.
- **Planning.** Establishes an incremental plan for the WebE work.
- **Modeling.** Encompasses the creation of models that assist the developer and the customer to better understand WebApp requirements and the design
- **Construction.** Combines both the generation of HTML, XML, Java, and similar code with testing that is required to uncover errors in the code.
- **Deployment.** Delivers a WebApp increment to the customer who evaluates it and provides feedback based on the evaluation.

Adapting the Framework

- Adapt
  - to the problem
  - to the project
  - to the team
  - to the organizational culture
  - to adapt throughout the project as circumstances change!

Adapting the Framework

- Adaptation leads to,
  - Overall flow of activities, actions, and tasks and the interdependencies among them
  - Degree to which **work tasks are defined** within each framework activity
  - Degree to which **work products are identified** and required
  - Manner in which **quality assurance** activities are applied
  - Manner in which **project tracking and control activities** are applied
  - Overall **degree of detail** and rigor with which the process is described
  - Degree to which **customers and other stakeholders** are involved with the project
  - Level of **autonomy given to the software project team**
  - Degree to which **team organization and roles are prescribed**

Underlying Agility Principles

1. **Highest priority is to satisfy the customer** through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness **continuous change for the customer's competitive advantage**.
3. **Deliver working software increments frequently**, from as often as every few days to every few months, with a preference to the shorter timescales.

Underlying Agility Principles

4. Business people and developers **must work together** daily throughout the project.
5. Build projects around motivated people. Give them needed **environment and support**, and trust them to get the job done.
6. The most efficient and effective method of **conveying information** to and within a development team is face-to-face conversation.

Underlying Agility Principles

7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence and good design** enhances agility.

Underlying Agility Principles

10. **Simplicity**—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At **regular intervals, the team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Web Engineering = Software Engineering ?

- Software engineering principles, concepts, and methods can be applied to Web development, but their application **requires a somewhat different approach** than their use during the development of conventional software based systems.
- Software engineering is a layered technology

Software Engineering Layers

- Quality: Foster a continuous process improvement culture
- Process: The glue that holds the technology layers together
  - Work products (e.g., models and documents) are produced, milestones are established, quality is ensured, and change is properly managed
- Methods: Provide the technical how-to's
  - Communication, requirements analysis, design modeling, program construction, testing, and support.
- Tools: Support for the process and the methods

Web Engineering

- Web Engineering differed from Software Engineering…
  - WebE framework must be defined within a process that:

    (1) embraces change,

    (2) encourages the creativity and independence of development staff and strong interaction with WebApp stakeholders,

    (3) builds systems using small development teams, and

    (4) emphasizes incremental development using short development cycles

WebE Methods

- Encompasses a set of technical tasks that enable a Web engineer to understand, characterize, and then build a high-quality WebApp
  1. Communication methods
  2. Requirements analysis methods
  3. Design methods
  4. Construction methods
  5. Testing methods

WebE Methods

1. Communication methods
   - Define the approach used to facilitate communication between Web engineers and all other **WebApp stakeholders** (e.g., end users, business clients, problem domain experts, content designers, team leaders, project managers)
   - Communication techniques are important during requirements gathering and whenever a WebApp increment is to be evaluated
2. Requirements analysis methods
   - Provides understanding the deliverable content of a WebApp, functions for the end user, and the navigation modes of interaction for each class of user

WebE Methods

3. Design methods
   - Design techniques for WebApp content, application and information architecture, interface design, and navigation structure
4. Construction methods
   - Set of languages, tools, and related technology to create WebApp
5. Testing methods
   - Testing component-level and architectural issues
   - Navigation testing
   - Usability testing
   - Security testing
   - Configuration testing

WebE Methods

- Other than these are
  - Project management techniques
    - Estimation
    - Scheduling
    - Risk analysis
  - Software configuration management techniques
  - Review techniques

Industry-Quality WebApps

Characteristics

- Take the time to understand 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.
- *Always d* evelop a project plan, even if it's 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 reinvent when you can reuse.
- Don't rely on early users to debug the WebApp—design and use comprehensive tests before releasing the system.

Industry-Quality WebApps

Characteristics :

1. Take the time to understand business needs and product objectives, even if the details of the WebApp are vague
   - Many WebApp developers erroneously believe that vague requirements (which are quite common) relieve them from the need to be sure that the system they are about to engineer has a legitimate business purpose
   - The end result is (too often) good technical work that results in the wrong system being built for the wrong reasons and for the wrong audience
   - If stakeholders cannot describe a business need for the WebApp, proceed with extreme caution
   - If stakeholders struggle to identify a set of clear objectives for the product (WebApp), do not proceed until they can

Industry-Quality WebApps

Characteristics :

2. Describe how users will interact with the WebApp using a scenario based approach
   - Stakeholders should be convinced to develop scenarios (Chapters 4, 5, and 7) that reflect how various users will interact with the WebApp
   - These scenarios can then be used:
     (1) for project planning and tracking,
     (2) to guide analysis and design modeling, and
     (3) as important input for the design of tests

Industry-Quality WebApps

Characteristics :

3. Develop a project plan, even if it's very brief
   - Base the plan (Chapter 5) on a process framework that is acceptable to all stakeholders
   - Because project time lines are very short, use a "fine" granularity for schedule-project should be scheduled and tracked on a daily basis
   - Many WebApp developers erroneously believe that vague requirements (quite common) relieve them from the need to be sure that the system they are about to engineer has a legitimate business purpose
     - The end result is (too often) good technical work that results in the wrong system being built for the wrong reasons and for the wrong audience
     - If stakeholders cannot describe a business need for the WebApp, proceed with extreme caution

Industry-Quality WebApps

Characteristics :

4. Spend some time modeling what it is that you're going to build

- Generally, comprehensive analysis and design documentation is *not developed* as a part of Web engineering work
- However, well-targeted graphical models (Chapters 6 through 12) can and do illuminate important engineering issues

Industry-Quality WebApps

Characteristics :

5. Review the models for consistency and quality
   - Pair walkthroughs and other types of reviews (Chapter 5) should be conducted throughout a WebE project
   - The time spent on reviews pays important dividends because it often eliminates rework and results in a high-quality WebApp—thereby increasing customer satisfaction

Industry-Quality WebApps

Characteristics :

6. Use tools and technology that enable you to construct the system with as many reusable components as possible
   - A wide array of WebApp tools is available for virtually every aspect of the WebApp construction (Chapter 14)
   - Many of these tools enable a Web engineer to build significant portions of the application using reusable components

Industry-Quality WebApps

Characteristics :

7. Don’t reinvent when you can reuse
   - A wide range of design patterns have been developed for WebApps
   - These patterns allow a WebE team to develop architectural, navigation, and component-level details quickly using proven templates (See Chapter 13 for a detailed discussion)

Industry-Quality WebApps

Characteristics :

8. Don’t rely on early users to debug the WebApp—design comprehensive tests and execute them before releasing the system
   - Users of a WebApp will often give it one chance. If it fails to perform, they move elsewhere—never to return
   - It is for this reason that “test first, then deploy” should be an overriding philosophy, even if deadlines must be stretched
   - See Chapter 15 for details

Chapter 3

- *The WebE Process*

Chapter 3: *The WebE Process*

- The process must be agile and adaptable, but it must also be *incremental*
- Why incremental?
  - Requirements evolve over time
  - Changes will occur frequently (and always at inconvenient times
  - Time lines are short
- Incremental delivery allows you to manage this change!

Incremental Delivery

Repeat the development cycle for each increment!

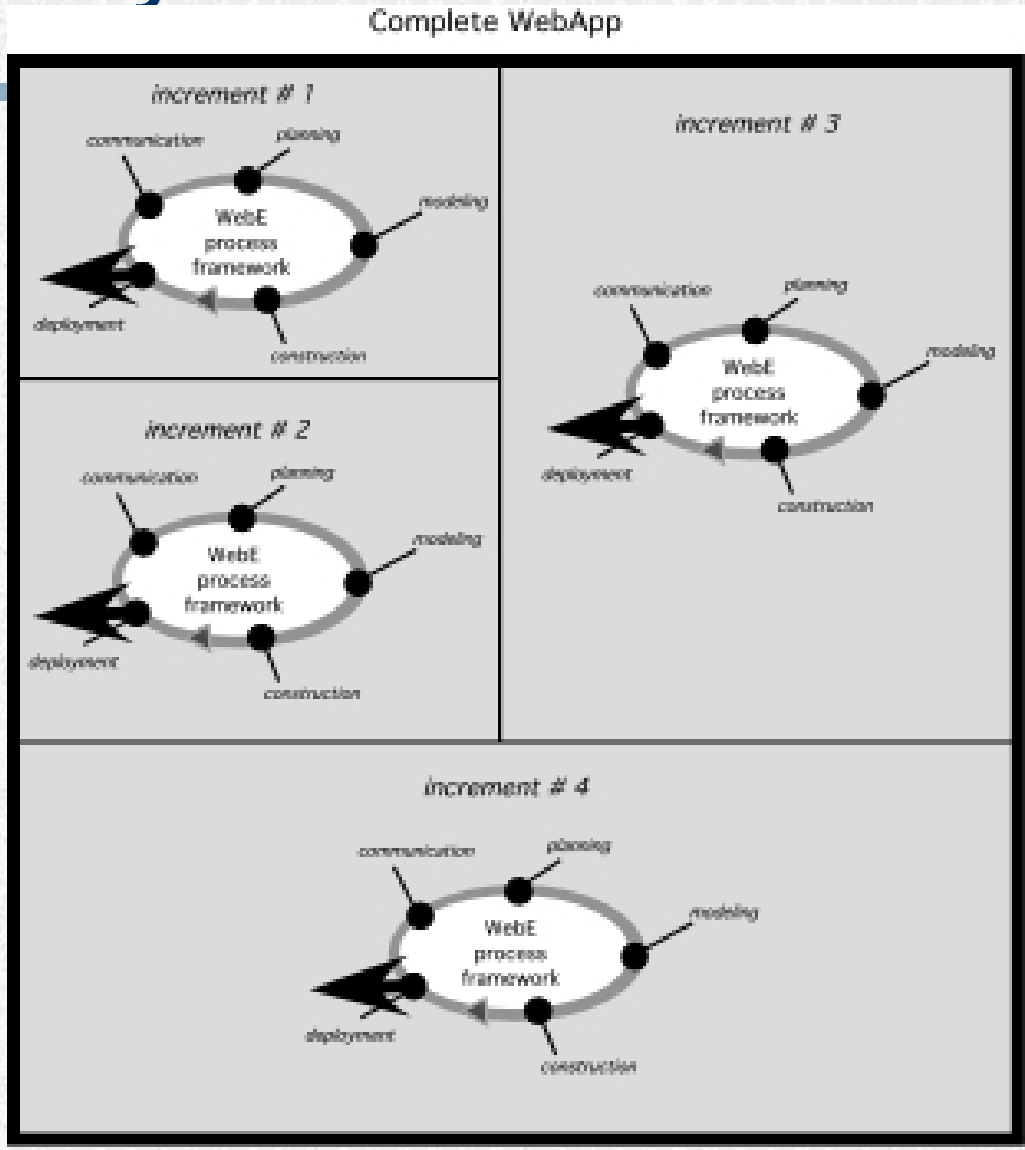
-Communication

-Planning

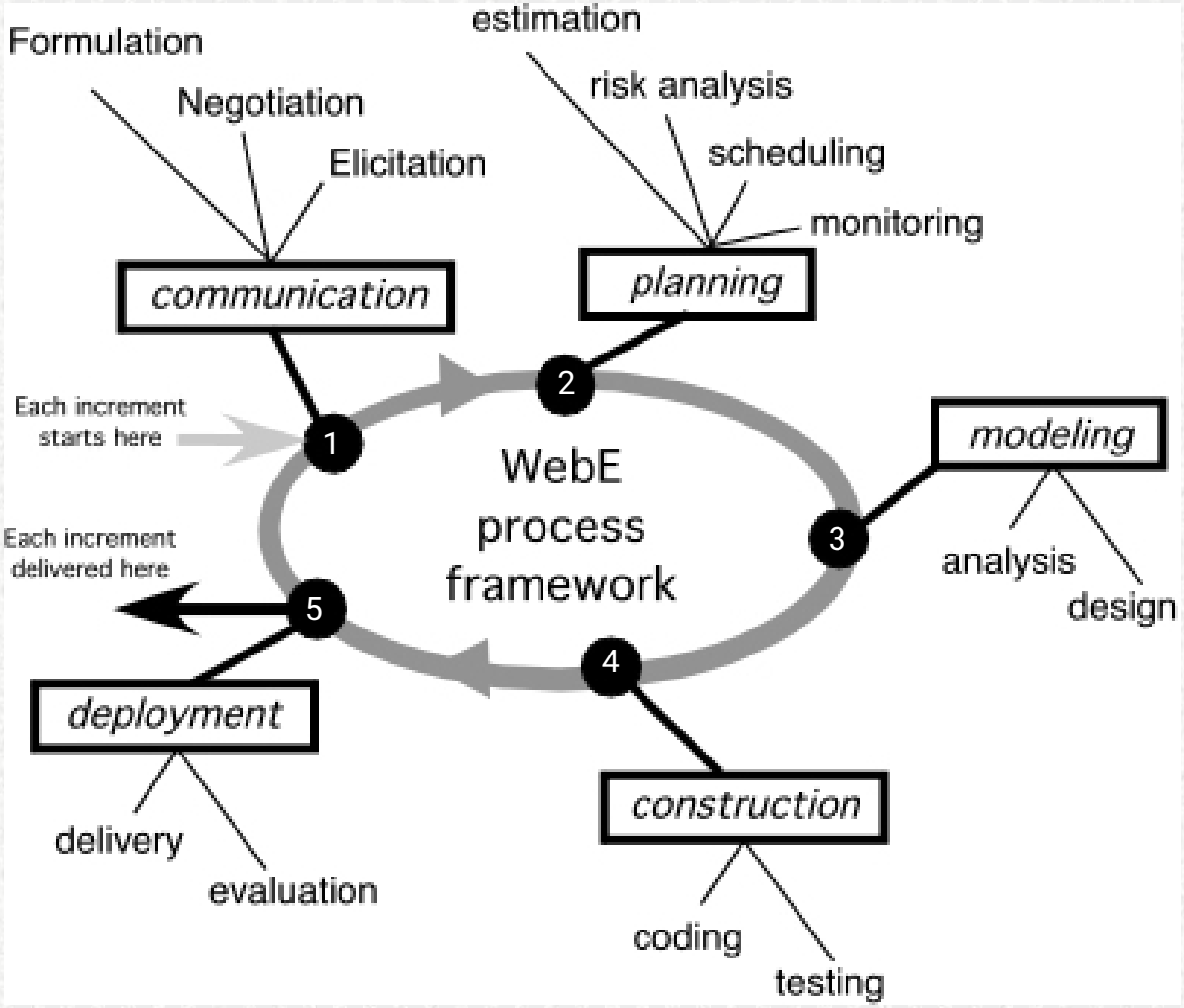
-Modeling

-Construction

-Deployment



WebE Process Activities & Actions



WebE Process: Communication

- *Communication is the activity that establishes the "destination" for a WebApp* project
- For a simple destination, there are a relatively small number of informal actions and tasks required to be sure you know where you're going
- If the destination is more difficult to describe, you'll need to refine the communication activity with more care

WebE Process: Communication

- Objectives:
  - Identify business stakeholders
  - Identify user categories
  - Formulate the business context
  - Define key business goals and objectives for the WebApp
  - Identify the problem to Solve, Input/Output for the End User, functionality required to manipulate data
  - Define informational and applicative goals for classes of content are to be provided to end users, and how dynamic is the content; that is, how often does it change…
  - Gather requirements about content, interaction metaphor, Computational functions, navigation schema
  - Develop usage scenarios

WebE Process: Planning

- Activity requiring the following to model, construct, and deploy the increment
  - Estimation of effort and time required to deploy the increment
    - In terms of person-days and time (calendar days)
    - Resources (people, hardware, software) required to do the work
  - Assessment of risks associated with the delivery of the increment high-probability, high-impact risks to be mitigated
  - Development schedule for the increment for tasks to be allocated along the time line and to establish intermediate milestones
  - Establishment of
    - Work products (written scenarios, sketches, models, documents) to be produced as a consequence of each framework activity
    - Quality assurance approach

WebE Process: Modeling

- *An activity* that creates one or more conceptual representations of some aspect of the WebApp to be built
- A*conceptual representation-one or more of the following* forms:
  1. Written documents
  2. Sketches
  3. Schematic diagrams
  4. Graphical models
  5. Written scenarios
  6. Paper or executable prototypes
  7. Executable code
- Two Web engineering actions occur during modeling:
  - *Analysis*
  - *Design*

WebE Process: Analysis Modeling

- This model focuses on WebApp content, modes of interaction (including navigation), functionality, and the technical configuration of the WebApp
- From the communication,
  - If information exists and is complete, → no need for analysis modeling for the increment
  - If the information is incomplete or implies a degree of complexity → demands further examination, proceed to the analysis modeling tasks

WebE Process: Analysis Modeling

- The following tasks are to be done during an analysis model:
  - Represent WebApp content, decide static (do not change based on user type/input) and dynamic (generated based on user type/input)
  - Identify content form and style of each content class and relationships among them
  - Refine and extend user scenarios for input, steps and to check consistency and of enough detail
  - Create an interaction model for complex scenarios showing the relationship between user and each task and user actions transition from one state to another
  - Refine interface requirements for menus, layout, navigation
  - Identify functions and user input for function
  - Define constraints and performance requirements and privacy policy
  - Identify database requirements, content interface for the database

WebE Process: Design Modeling

- The goal is to produce a model or representation
- If the increment is well understood and very easy to construct, the only design model might be a simple sketch
- If, on the other hand, the increment is more complex, a more detailed design model may be created

WebE Process: Design Modeling

- The model can consider some/all aspects of WebApp design:
  - Interface design
  - Aesthetic design
  - Content design
  - Navigation design
  - Architecture design
  - Component design

WebE Process: Design Modeling

- The model can consider some/all aspects of WebApp design:
  - Interface design
    - Describes the structure and organization of the user interface
    - Includes a representation of screen layout, a definition of the modes of interaction, and a description of navigation mechanisms
  - Aesthetic design
    - Aka graphic design, describes the “look and feel” of the WebApp
    - Includes color schemes, geometric layout, text size, font and placement, the use of graphics, and related aesthetic decisions
  - Content design
    - Defines the layout, structure, and outline for all content
    - Establishes the relationships among content objects

WebE Process: Design Modeling

- The model can consider some/all aspects of WebApp design:
  - Navigation design
    - Represents the navigational flow among content objects and functions
  - Architecture design
    - Identifies the overall hypermedia structure for the WebApp
  - Component design
    - Develops the detailed processing logic required to implement functional components that implement a complete WebApp function

WebE Process: Design Modeling

- The following to consider to develop a design model:
  - For each usage scenario, design the
    - Interaction tasks and subtasks to be represented as part of the interface
    - Required interface control mechanisms (e.g., links, buttons, menus)
    - Control mechanisms positions on a Web page
  - Design the aesthetic for the WebApp user categories by considering consistent
    - Page layout , color and form, navigation mechanisms positions and representation, all logos, graphics, images, and backgrounds implemented
  - Design the content/databases and data structures required to implement functionality or to display content
  - Design appropriate security and privacy mechanisms

WebE Process: Construction

- As construction proceeds, you can perform two WebE actions:
  - Code generation
  - Testing
- The following tasks help you plan the code generation action:
  - Build and/or acquire all content, and integrate the content into the WebApp architecture
  - Select the appropriate tool set for the generation of HTML code
  - Implement each page layout, function, form, and navigation capability
  - Implement all forms, scripts, and database interfaces and computation functions for the client/server side
  - Address configuration issues of browsers, plug-ins, and operating system environments for both the client/server sides

WebE Process: Construction

- Once the WebApp has been constructed, it must be tested
- Testing begins with a relatively narrow focus and then continues to exercise a broader view of the WebApp
- The following tasks to plan the testing action :
  - Test all WebApp components (content and function)
  - Test navigation

WebE Process: Deploy

- The following tasks are considered to deploy the WebApp increment:
  - Deliver the WebApp increment to a server at a predefined domain
  - Establish an online feedback mechanism for end users
  - Evaluate end-user interaction, assess lessons learned and consider all end-user feedback and make modifications to the WebApp increment as required

Ch.7 Analysis Modeling of WebApps

- Modeling Language (ML)
  - Incorporates a set of notations, terms, and/or symbols, as well as the rules for establishing associations between them
  - 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)

Ch.7 Analysis Modeling of WebApps

- Many software tools exist for the construction of WebApps, relatively few have been developed specifically for analysis
- Four categories of tools can be used for analysis:
  1. **UML tools**
  2. **Prototyping tools**
  3. **Issue tracking tools**
  4. **Content management tools**

Ch.7 Analysis Modeling of WebApps

1. **Unified Modeling Language (UML) tools**
   - Used to create analysis models
   - Widely used in the software engineering community
2. **Prototyping tools**
   - Virtually any WebApp construction tool (e.g., Adobe *GoLive) can be used to create an operational prototype*
   - Allow fast layout, integration of content, and the development of rough aesthetics, all appropriate for the creation of a quick prototype
3. **Issue tracking tools**
   - Used to record and track the resolution of issues that arise out of interpretation of the emerging analysis models
4. **Content management tools**
   - Used to model the nature and structure of content objects of the WebApp

Case Study

- You've been approached by CPI Corporation, a company that builds, markets, sells, and monitors security systems for homes and small businesses. CPI has no Web presence and wants to roll out a "significant" website that will coincide with the introduction of a new line of security sensors and a set of radically new Web-based services. They want your help in the development of the WebApp, which is called SafeHomeAssured.com, and at the same time for you to assist them as they create new Web services that will increase their market share.
- CPI has engineered a compact, wireless sensor controller that will become the core element in a new line of commercial and residential security systems that it intends to call *SafeHome.*

Case Study

- The features (content and function) to be implemented:
  - Information about CPI and its products and people
  - Specifications for all security hardware components, including pictures, technical descriptions, installation instructions, pricing, and other pertinent information
  - Security system design assistance that enables a customer to specify a living or business space (e.g., rooms, doors, windows) and then get semi automated layout assistance for a security system
  - e-Commerce capability that enables a customer to order security hardware and monitoring services
    - This capability will be coupled to backroom systems that support a customer purchase

Case Study

- The following features (content and function) will be implemented:
  - Customer monitoring via the Internet to enable a homeowner or business person to use video to monitor a space in real time
  - Customer account access capability
  - Customer service access capability including specialized in-house functionality
  - Technical service staff access capability including specialized in-house functionality
  - In addition, CPI wants to abandon a brick-and-mortar sales strategy (i.e., salespeople, store fronts) and move toward a twenty-first century paradigm. The company wants to sell exclusively via the Web.

UML

- Allows the “modeling”of actors
- But in UML an actor strictly represents the idealization of a particular interaction with the system—rather than a user
- The same user may participate in different interactions (and therefore be represented as different actors), but that user will carry his or her context across these roles
- It is important to model the user context and not just the individual interactions
  - To achieve these, both the different users (using the «user» actors) and the different types of interactions in which those users participate (using the «interaction» actors)

UML-Case Study Actors

- For SafeHomeAssured.com there are three main types of users, who can carry out four different types of interactions:

(1) Guests

- Can browse the public information on products and download specs, as well as purchase products (but the system won't remember their preferences)

(2) Registered users

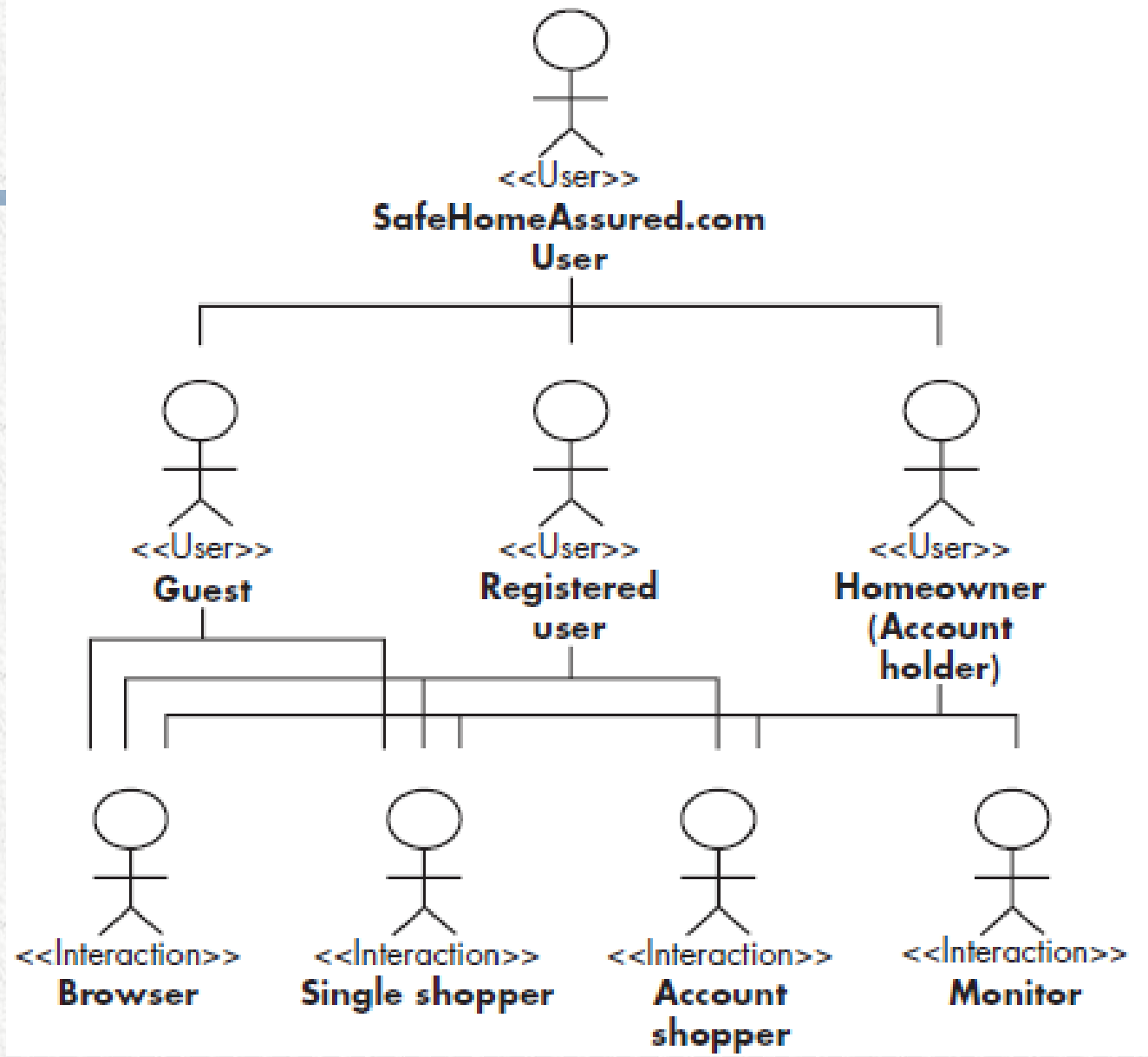
- Have access to the same functionality as guests, but the system will remember their preferences as well as allow them to purchase on an account

(3) Registered homeowners

- Can browse information and buy products, but can also have their system monitored online

- For each «user» actor → need to understand the nature of the user category—its context, needs, and behaviours

UML-Actors


<<User>>
SafeHomeAssured.com User
<<User>>
Guest
<<User>>
Registered user
<<User>>
Homeowner (Account holder)
<<Interaction>>
Browser
<<Interaction>>
Single shopper
<<Interaction>>
Account shopper
<<Interaction>>
Monitor

UML-Conversation

- The vast majority of WebApps enable a “conversation” between an end user and application functionality, content, and behavior
- This conversation can be described using an *interaction model that can be composed of one or more of the following* elements: (1) use cases, (2) sequence diagrams, (3) state diagrams,and/or (4) user interface prototypes
- In addition to these representations, the interaction is also represented within the context of the navigation model → “Relationship-Navigation Analysis”

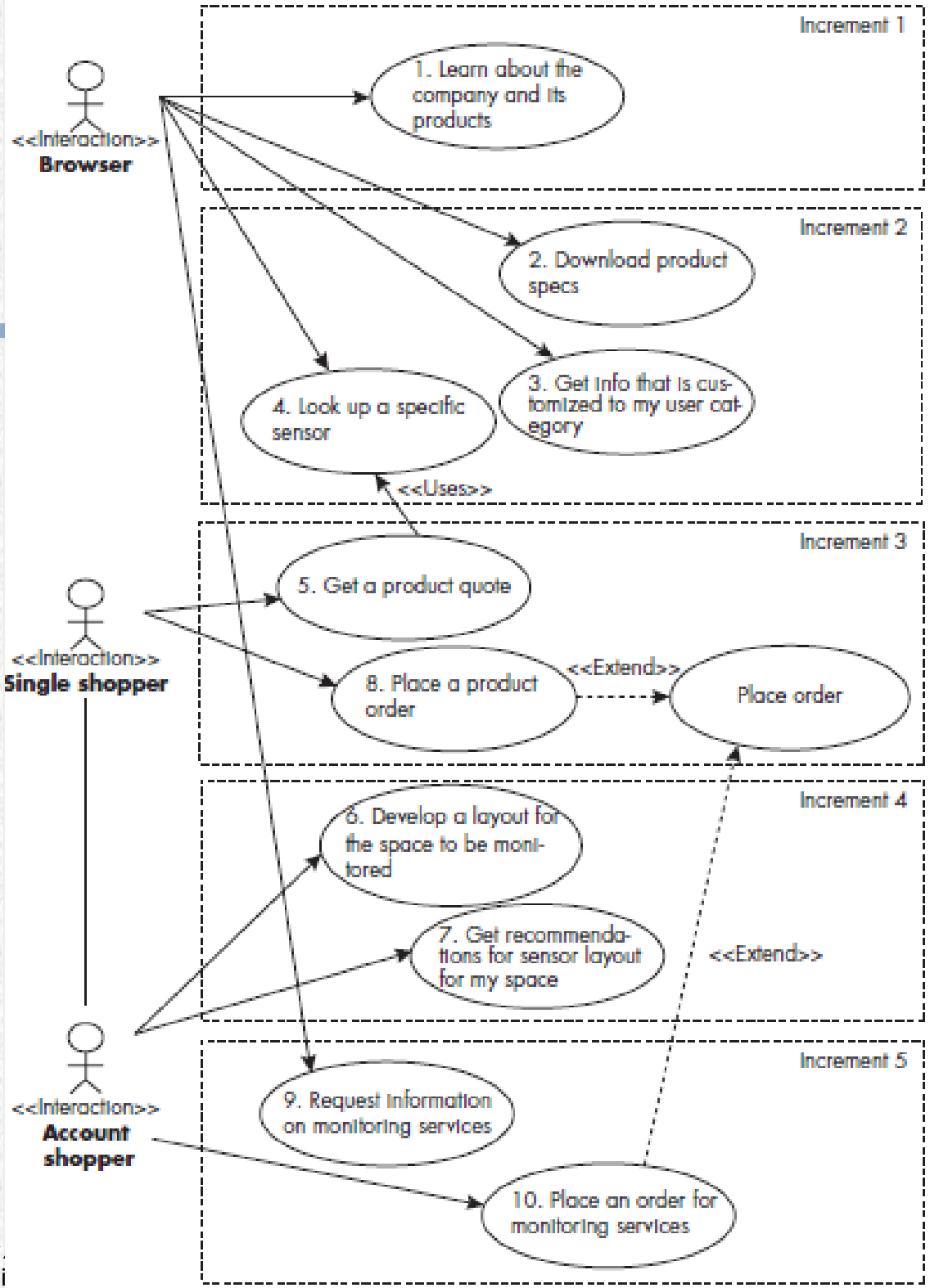
Use Cases

- The dominant element of the interaction model for WebApps
- Not uncommon to describe 100 or more use cases when large, complex WebApps are analyzed, designed, and constructed
- However, a relatively small percentage of the use cases describes the major interactions between end-user categories (actors) and the system
- Other use cases refine the interactions, providing the analysis detail necessary to guide the design and construction

Use Cases

- In many instances, a set of use cases is sufficient to describe the interaction at an analysis level
- However, when the interaction sequence is complex and involves multiple analysis classes or many tasks, it is sometimes worthwhile to depict it using a more rigorous diagrammatic form

UseCase Diagram



Sequence Diagram

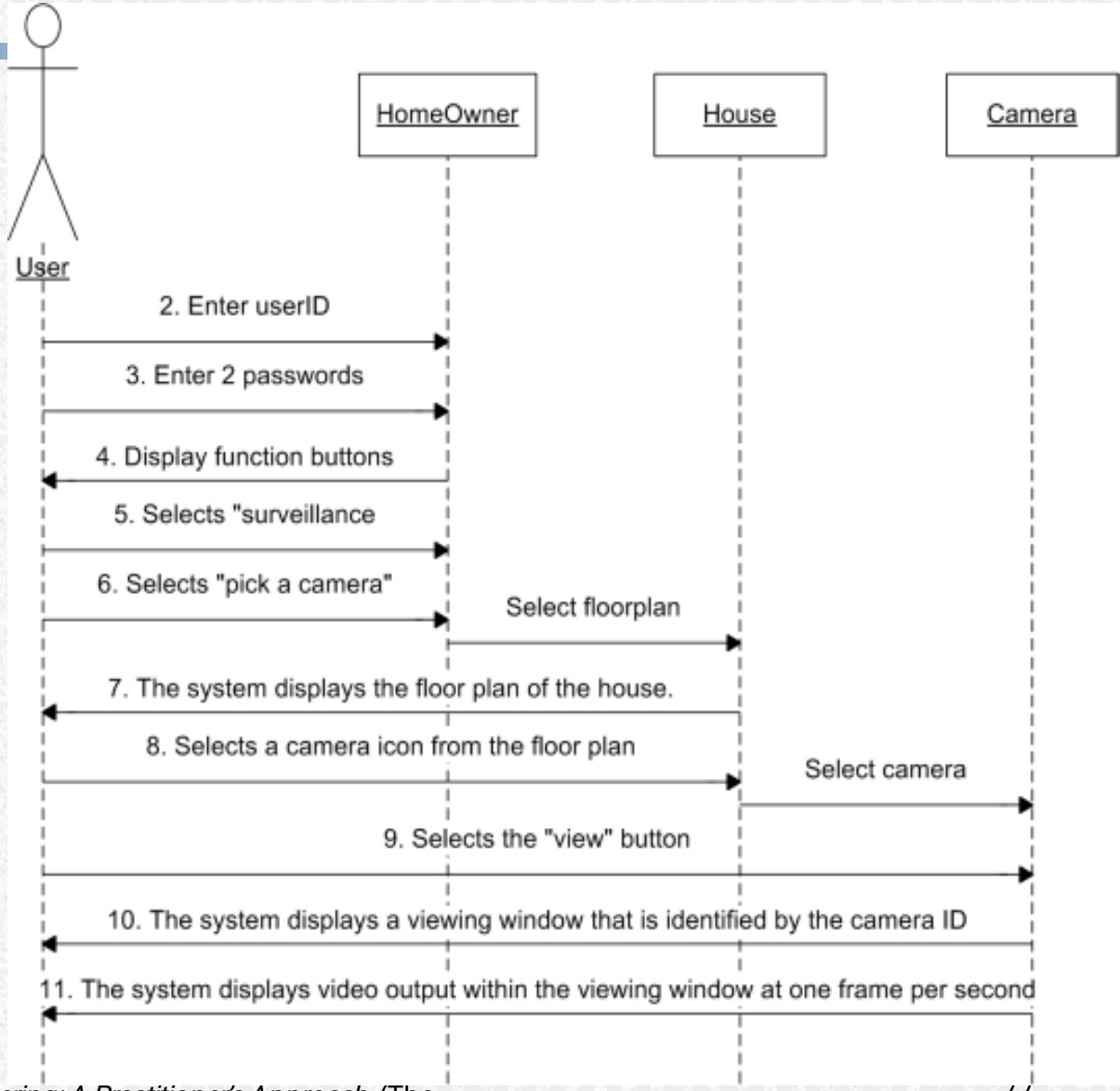
- Shorthand representation of the manner in which user actions (the dynamic elements of a system defined by use cases) collaborate with analysis classes (the structural elements of a system)
- Since analysis classes are extracted from use case descriptions, there is a need to ensure that traceability exists between the classes that have been defined and the use cases that describe system interaction
- The merging of dynamic and structural elements of the [analysis] model is the key link in the traceability of the model and should be taken very seriously

Sequence Diagram

- The vertical axis of the diagram
  - Depicts actions that are defined within the use case
- The horizontal axis
  - Identifies the analysis classes that are used as the use case proceeds
- For example:
  - A user logs in to the WebApp, navigates the relevant options, and selects a camera
  - The video from that camera is then displayed
- The movement across and down the sequence diagram ties each analysis class to use case actions
  - If a use case action is missing from the diagram, you should reevaluate the description of analysis classes to determine if one or more classes are missing

Sequence Diagram

- For the Access camera surveillance via the Internet use case
- Sequence diagrams can be created for each use case and the analysis classes defined for the use case



State Diagram

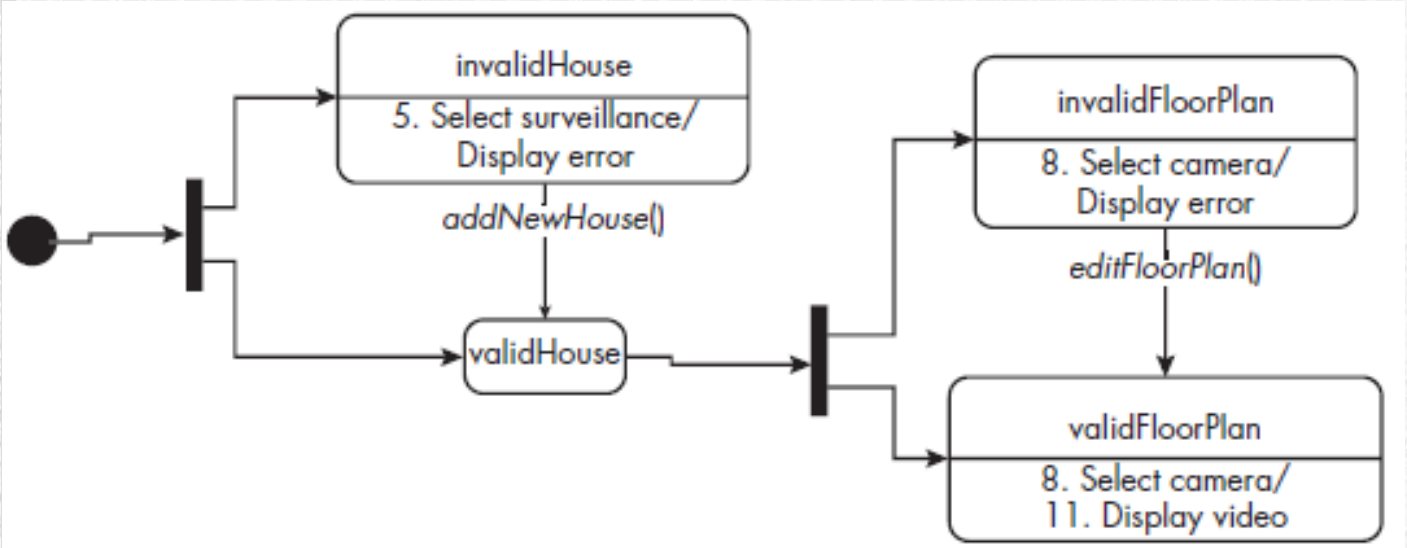
- Provides another representation of the dynamic behavior of the WebApp as an interaction occurs
- Like most modeling representations used in Web engineering, the state diagram can be represented at different levels of abstraction
- State diagrams are most useful when a user interaction triggers a change in the state of the WebApp—and hence changes the way in which it might react to a user

State Diagram

- A *change of state occurs when the user observes a new mode of behavior for* the WebApp
- For example:
  - When a user who is browsing product descriptions requests a product quote, the request triggers a change of state from *a product overview* state to *a product quotation state*
- *The WebApp look and feel observed by the* user may also change as a consequence

State Diagram

- Often these changes of state will relate to interconnections between different use cases
- For example, the Access camera surveillance via the Internet use case involves selecting a camera from a floor plan—but this is only possible if that user has previously configured (or had SafeHomeAssured.com configure) a floor plan



Use Case+Sequence Dia+State Dia?

- **Needed all three to Fully Describe the Interaction Model?**
- Because use cases, sequence diagrams, and state diagrams all present related information, it is reasonable to ask why all three are necessary. But, in some cases, they are not.
- Use cases
  - May be sufficient in some situations, particularly if you have documented the exceptions thoroughly
  - However, use cases provide a rather one dimensional view of the interaction
- Sequence diagrams
  - Present a second dimension that is more procedural (dynamic) in nature
- State diagrams
  - Provide a third dimension that is more behavioral and contains information about potential navigation pathways that is not provided by use cases or the sequence diagrams

Use Case+Sequence Dia+State Dia?

- When all three dimensions are used, omissions or inconsistencies that might escape discovery in one dimension become obvious when a second (or third) dimension is examined
- It is for this reason that large complex WebApps can benefit from an interaction model that encompasses all three representations

Use Case+Sequence Dia+State Dia?

- Key factors affecting whether or not use cases, sequence diagrams, and state diagrams are all necessary include the complexity of the overall increment, the consequences of an error for users

Ch. 8 WebApp design

WebApp design

- A combination of art and engineering
- To be effective, a good design must accommodate each of the requirements established by stakeholders during the communication activity
- But it must also accommodate the inevitable changes to requirements that will occur throughout the design and into construction and delivery

WebApp design

- Design begins by focusing on the way in which a user will interact with an application and then moves to consider the functions and information content that will be required to meet stakeholders' needs
- It then proceeds to physical design, in which the logical elements are mapped into a representation that can be implemented as part of the WebApp
- The primary output of the design activity is a design model that encompasses interface descriptions, aesthetics, content, navigation, architecture, and component-level design issues

WebApp design

- Generic WebApp design goals:
  - simplicity, consistency, identity, robustness, navigability, visual appeal, and compatibility.
- A variety of quality frameworks can be applied to WebApp design
- A user-centric quality model provides a solid indication of the degree to which end users like the technology that the WebApp delivers
- WebApp designers can apply a Design IQ Checklist that serves to assess the importance of various quality criteria and the degree to which the WebE team has met each of the criteria

WebApp design

- The design process is an agile, iterative collection of design actions that are applied to each WebApp increment as it is created
- A design pyramid can be used to describe a set of design actions that are performed (with varying degrees of emphasis) for each WebApp increment
- The design actions include interface design, aesthetic design, content design, navigation design, functional design, architecture design, and component design
- Each of the design actions represents a challenge for the WebE team

Relation-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

Chapter 8 *WebApp Design*

- Jakob Nielsen [Nie00] states: "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."
- Even today, some proponents of agile software development use WebApps as poster children for the development of applications based on "limited design."
  - However --
    - When content and function are complex
    - when the size of the WebApp encompasses hundreds of content objects, functions, and analysis classes
    - when multiple people become involved in the design; and
    - when the success of the WebApp will have a direct impact on the success of the business,
    - design cannot and should not be taken lightly.

WebApp Design

- The design model encompasses content, aesthetics, architecture, interface, navigation, and component-level design issues.
- The design model provides sufficient information for the WebE team to construct the final WebApp
- alternative solutions are considered, and the degree to which the current design model will lead to an effective implementation is also assessed

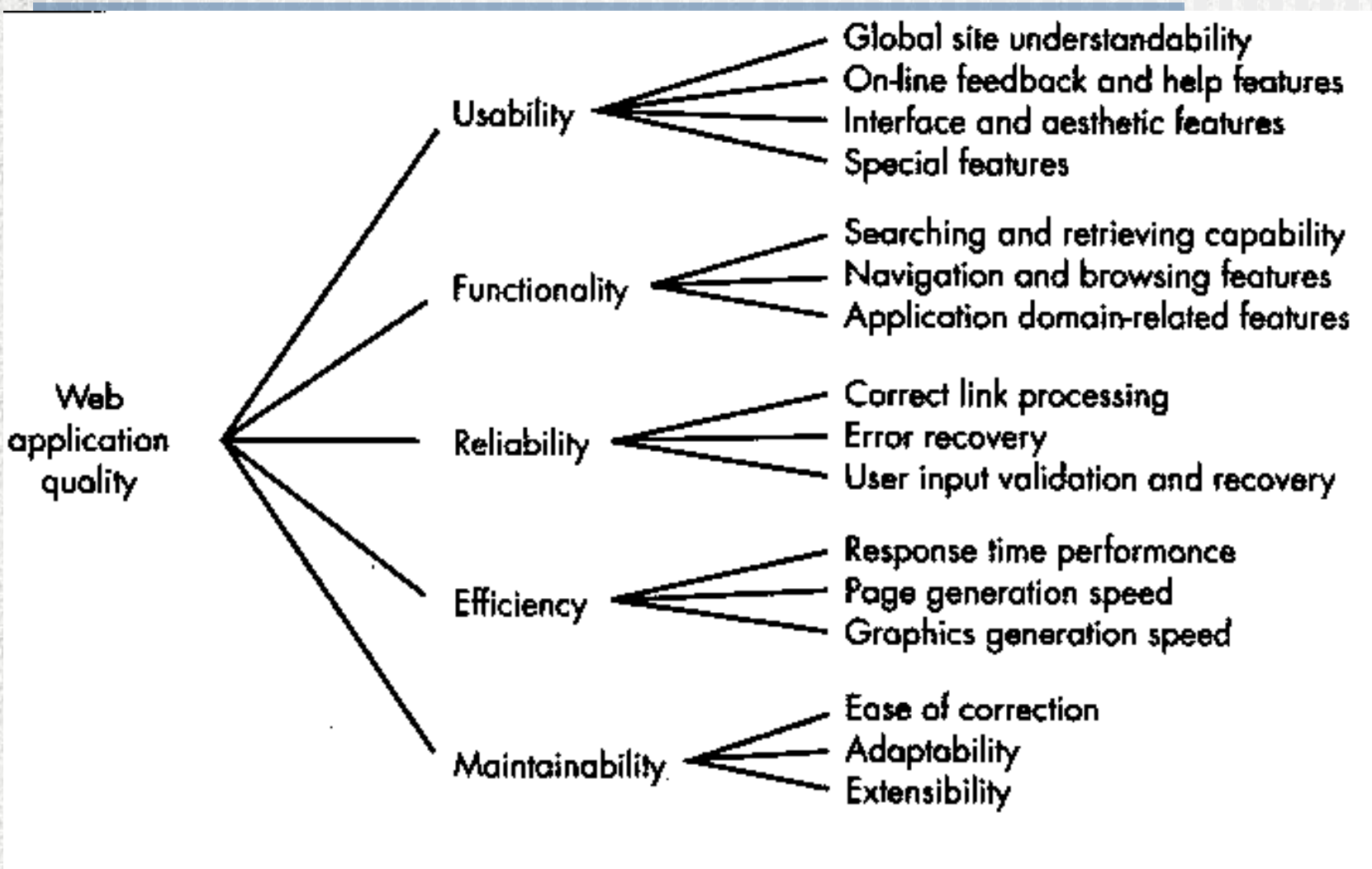
Design Goals - I

- **Simplicity.** Although it may seem old-fashioned, the aphorism "all things in moderation" applies to WebApps. Rather than *feature-bloat*, it is better to strive for moderation and simplicity.
- **Consistency.**
  - Content should be constructed consistently
  - Graphic design (aesthetics) should present a consistent look
  - Architectural design should establish templates that lead to a consistent hypermedia navigation
  - Navigation mechanisms should be used consistently
- **Identity.** The aesthetic, interface, and navigational design of a WebApp must be consistent with the application domain for which it is to be built.

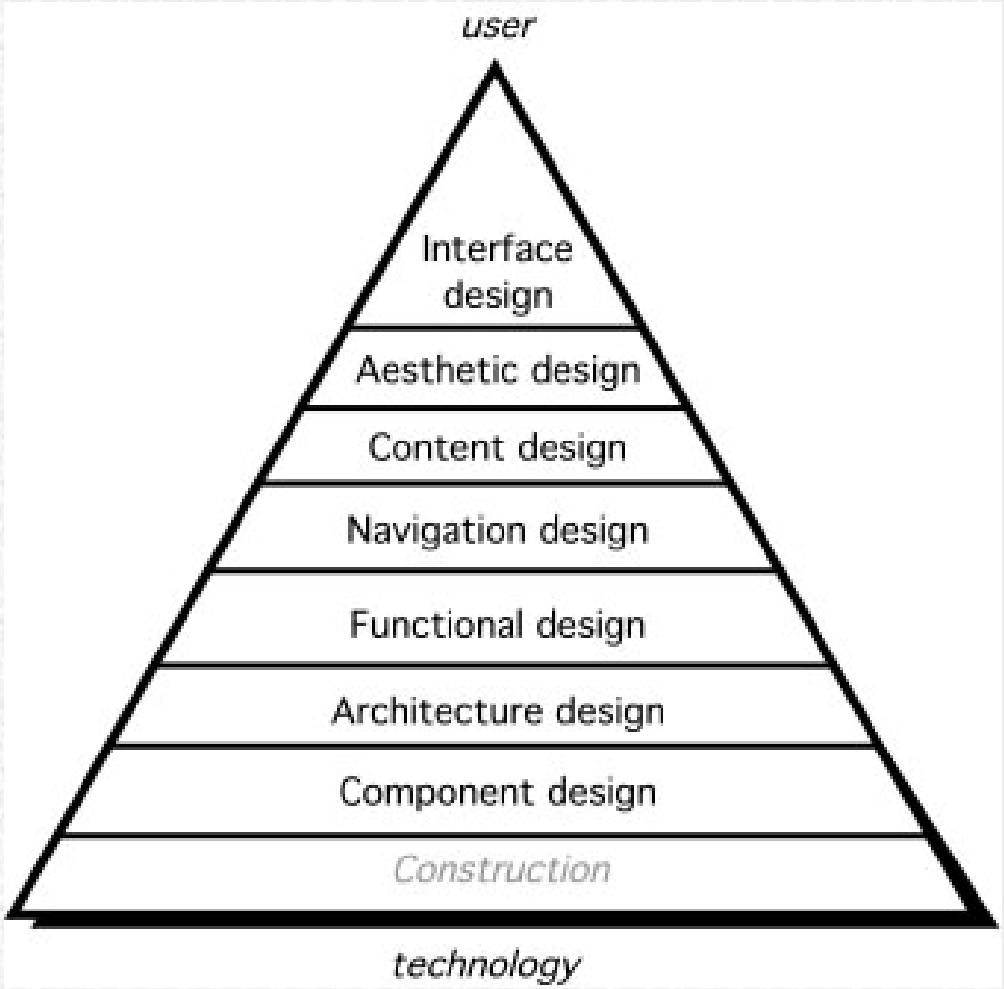
Design Goals - II

- **Robustness.** The user expects robust content and functions that are relevant to the user's needs.
- **Navigability.** users should be able to understand how to move about the WebApp without having to search for navigation links or instructions.
- **Visual appeal.** design characteristics (e.g., the look and feel of content, interface layout, color coordination, the balance of text, graphics and other media, and navigation mechanisms) contribute to visual appeal.
- **Compatibility.** Most WebApps will be used in a variety of environments (e.g., different hardware, Internet connection types, operating systems, and browsers) and must be designed to be compatible with each

Design & WebApp Quality



Design Actions



Conceptual Architecture

- Provides an overall structure for the WebApp design
  - Affects all later increments – so important for it to developed in the context of the full set of *likely* increments
- Represents the major functional and information components for the WebApp and describes how these will fit together
  - Depends on the nature of the WebApp, but in every case, it should ensure a sound integration between the WebApp information and the WebApp functionality.

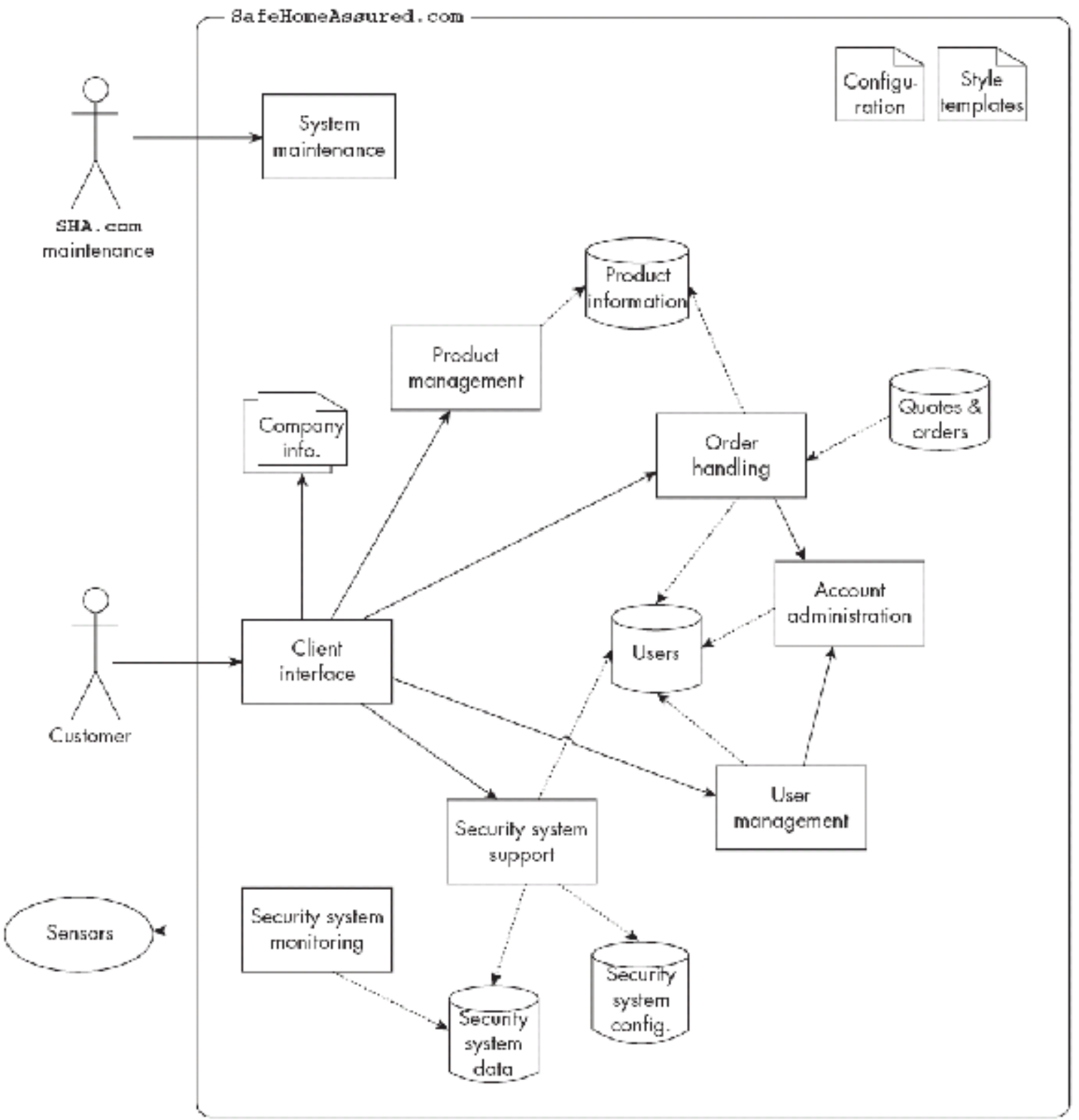
Developing the architecture-I

- How do we achieve an effective balance between information and functionality in the conceptual architecture?
- A good place to start is with workflows or functional scenarios (which are an expression of the system functionality) and information flows
- As a simple example, consider the following set of key functionalities for **SafeHomeAssured.com**
  - 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)

Developing the architecture-II

- From these key functionalities we can identify the following partial list of*functional subsystems:*
  - **UserManagement.** Manages all user functions, including user registration, authentication and profiling, user-specific content, and interface adaptation and customization.
  - **ProductManagement**. Handles all product information, including pricing models and content management.
  - **OrderHandling**. Supports the management of customers' orders.
  - **AccountAdministration**. Manages customers' accounts, including invoicing and payment.
  - **SecuritySystemSupport**. Manages users' space layout models and recommends security layouts.
  - **SecuritySystemMonitoring**. Monitors customers' security systems and handles security events.
- And, of course, there are overall management subsystems:
  - **ClientInterface.** Provides the interface between users and the other subsystems, as required to satisfy users needs.
  - **SystemMaintenance**. Provides maintenance functionality, such as database cleaning.

Architecture



Chapter 9 *Interaction Design*

- Design an interface to answer three generic questions:
  - *Where am I?* The interface should (1) provide an indication of the WebApp that has been accessed and (2) inform users of their location in the content hierarchy.
  - *What can I do now?* The interface should always help users understand their 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. Hence, it must provide a "map" (implemented in a way that is easy to understand) of where users have been and what paths they may take to move elsewhere within the WebApp.

Design Principles (Tognozzi) - I

- **Anticipation.** *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) should be consistent throughout the WebApp.*
- **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.*
- **Flexibility.** *The interface should be flexible enough to enable some users to accomplish tasks directly and others to explore the WebApp in a somewhat random fashion.*
- **Focus.** *The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.*

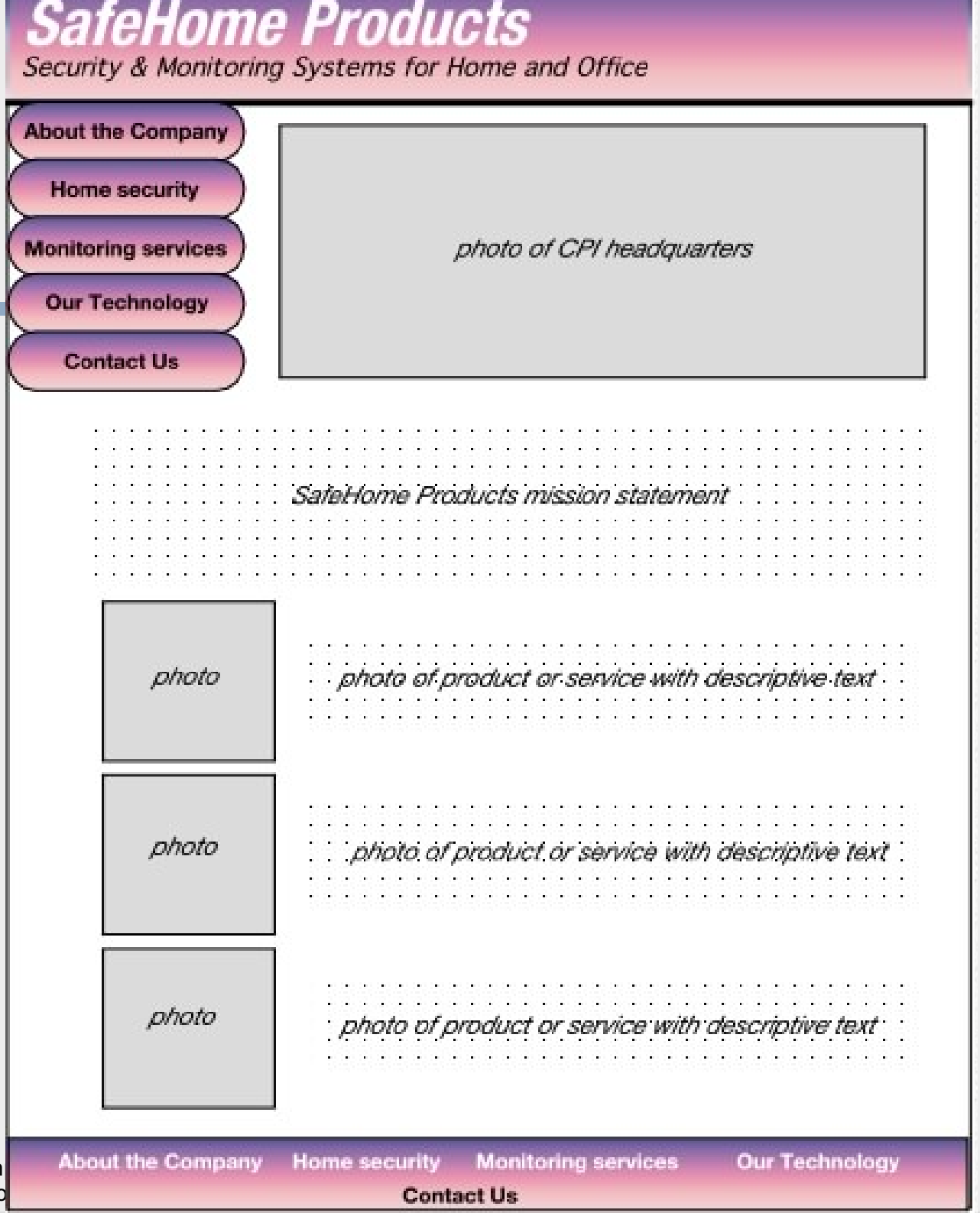
Design Principles (Tognozzi) - II

- **Fitt's law.** *"The time to acquire a target is a function of the distance to and size of the target"*
- **User Interface Objects.** *A vast library of reusable human interface objects (and patterns) has been developed for WebApps.*
- **Latency reduction.** *Rather than making the user wait for some internal operation to complete (e.g., downloading a complex graphical image), the WebApp should use multitasking 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.*
- **Metaphors.** *An interface that uses an interaction metaphor is easier to learn and easier to use, as long as the metaphor is appropriate for the application and the user.*

Design Principles (Tognozzi) - III

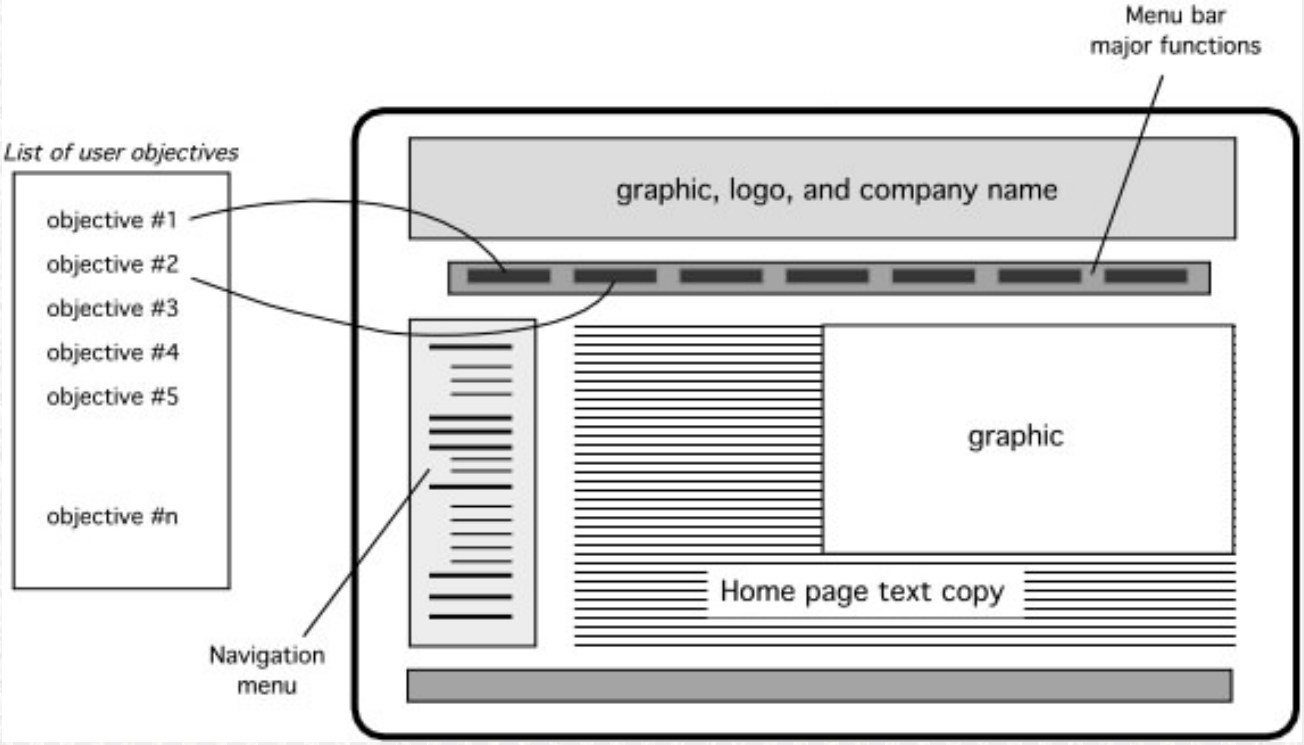
- **Maintain work product integrity.** *A work product (e.g., a form completed by the user, a user-specified list) must be automatically saved so that it will not be lost if an error occurs.*
- **Readability.** *All information presented through the interface should be readable by young and old.*
- **Track state.** *When appropriate, the state of user interactions should be tracked and stored so that users can log off and return later to pick up where they 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"*

Preliminary Page Layout



Interface Design Workflow - I

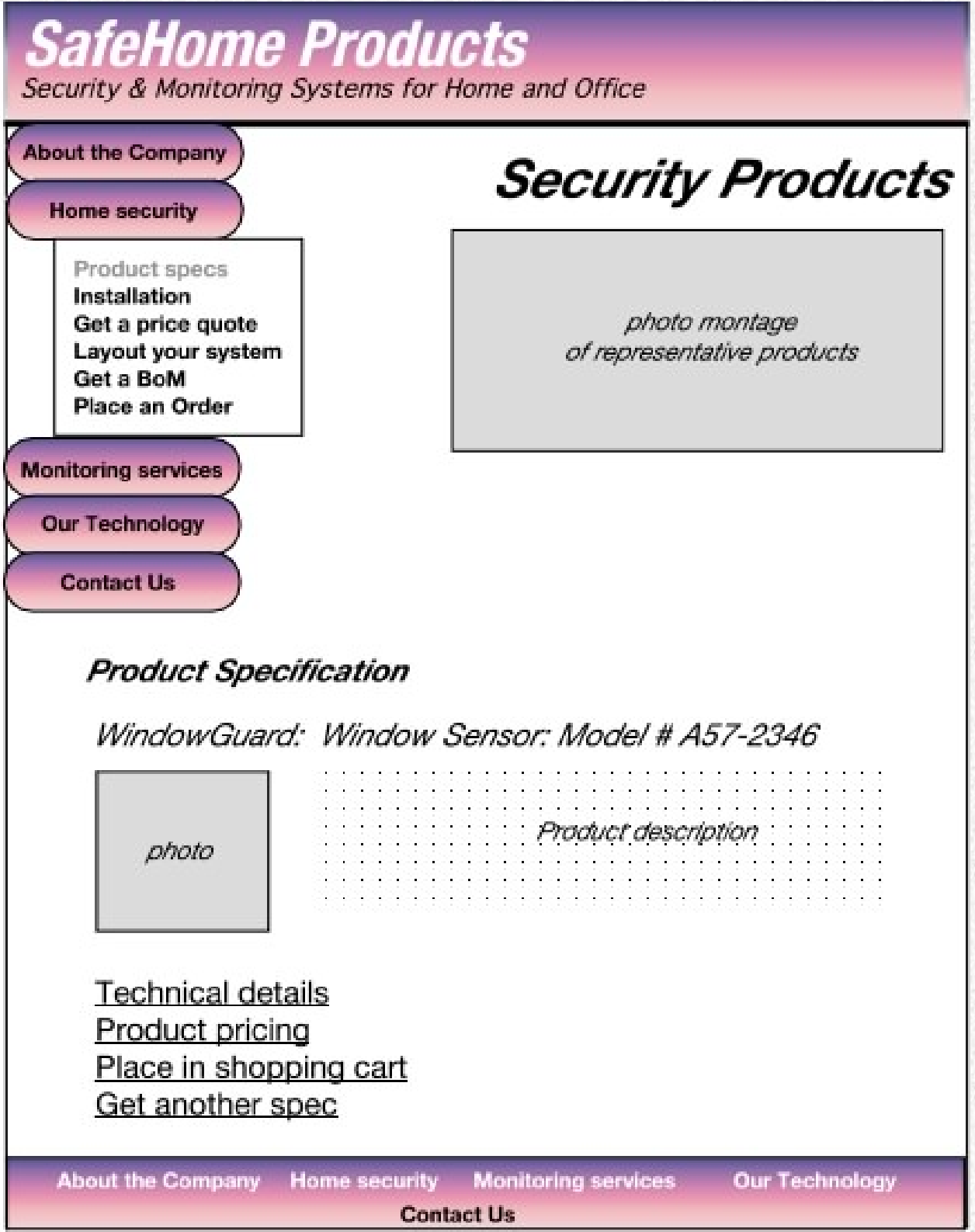
- Review user characteristics and categories, user tasks, use cases, and related information contained in the analysis model and refine as required.
- Develop a rough design prototype of the WebApp interface layout.
- Map user objectives into specific interface actions.



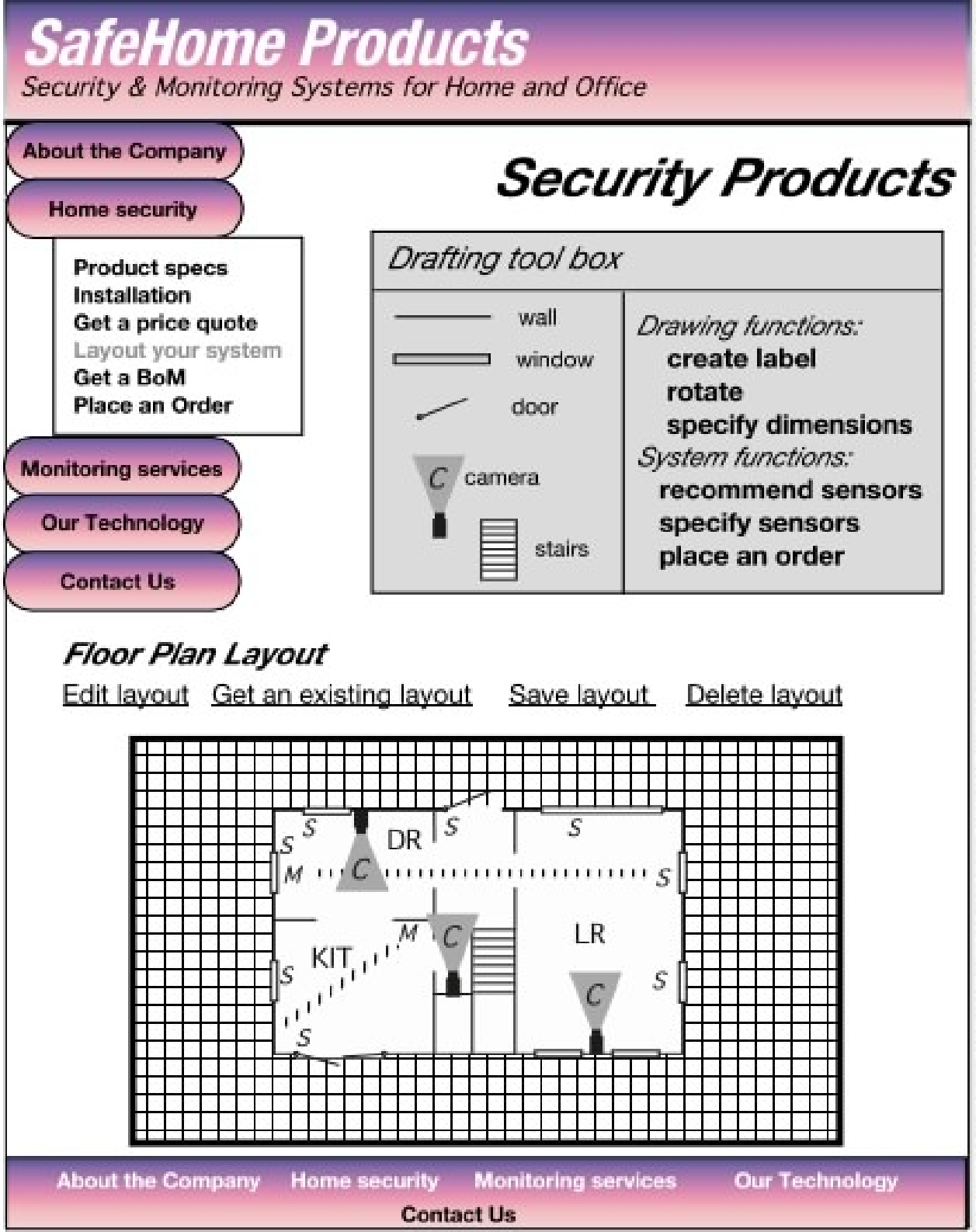
Interface Design Workflow - II

- Define a set of user tasks that are associated with each action.
- Develop screen images for each interface action.
- Refine interface layout and screen images using input from aesthetic design.
- 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 of the interface.
- Describe the interface layout for each state.
- Pair walkthroughs (Chapter 5) should be conducted throughout all these design tasks and should focus on usability.

Elaborate the design



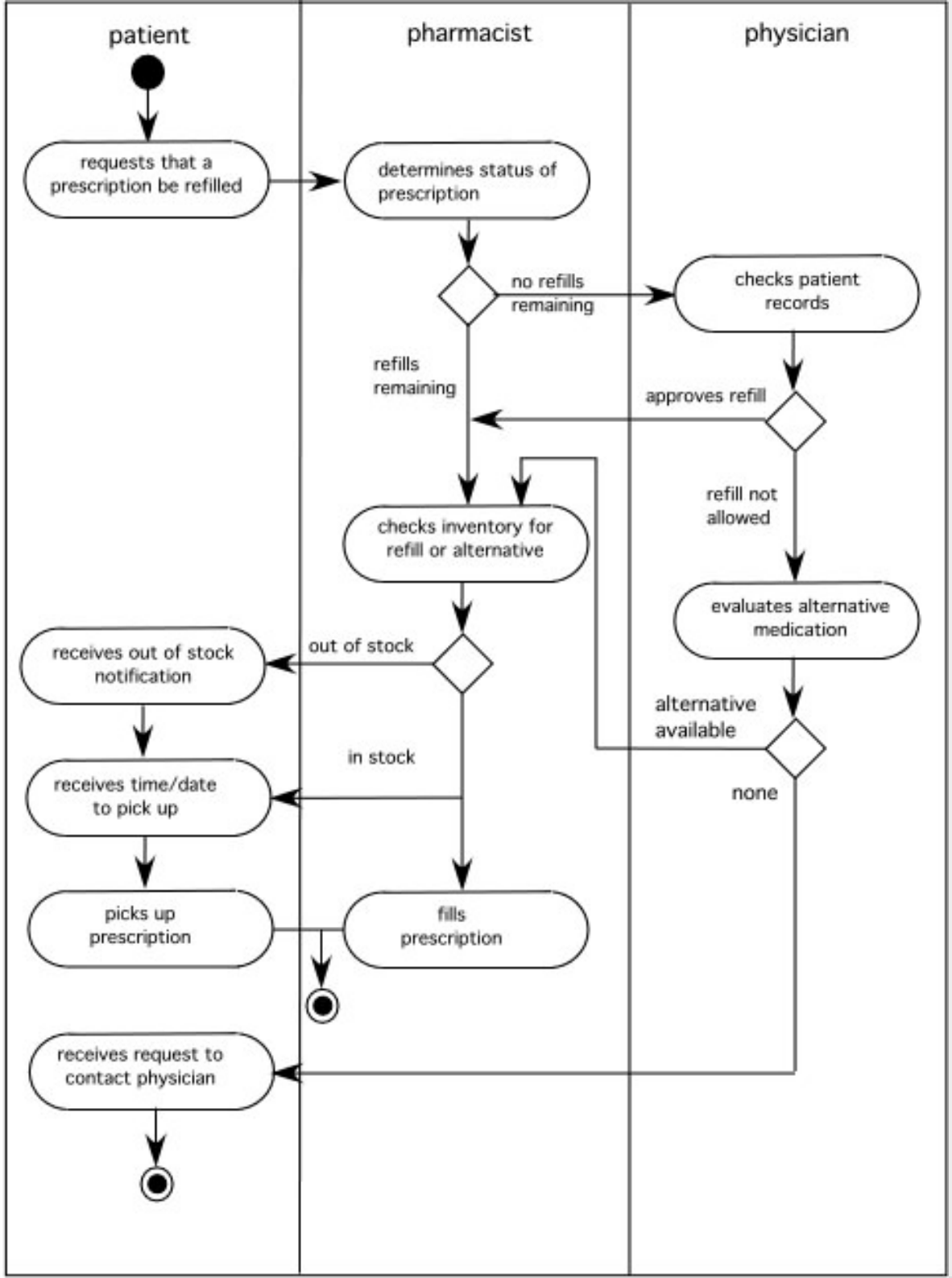
Elaborate the Design



Different Users in Different Roles

The swimlane diagram:

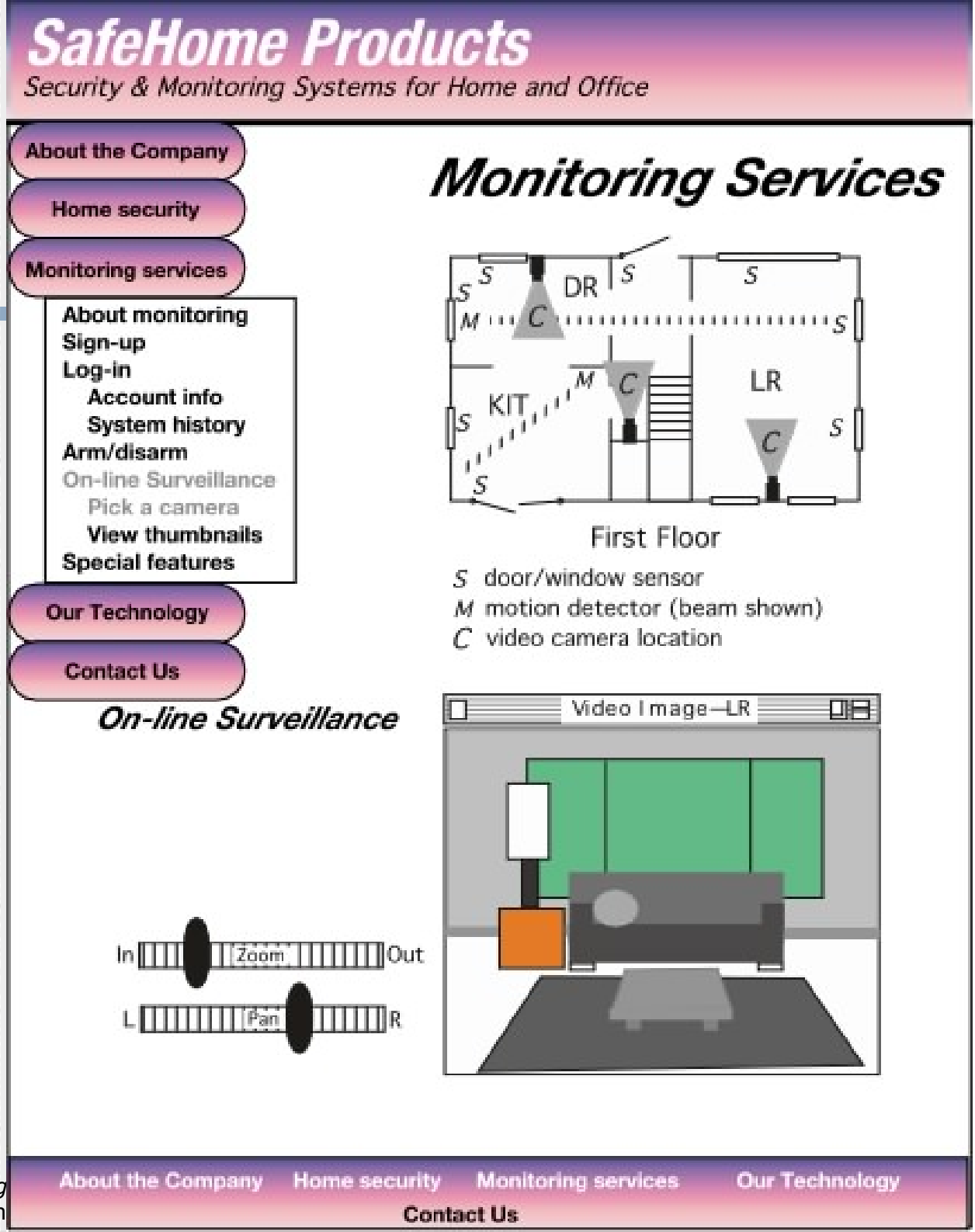
Captures workflows and shows interactions between different users



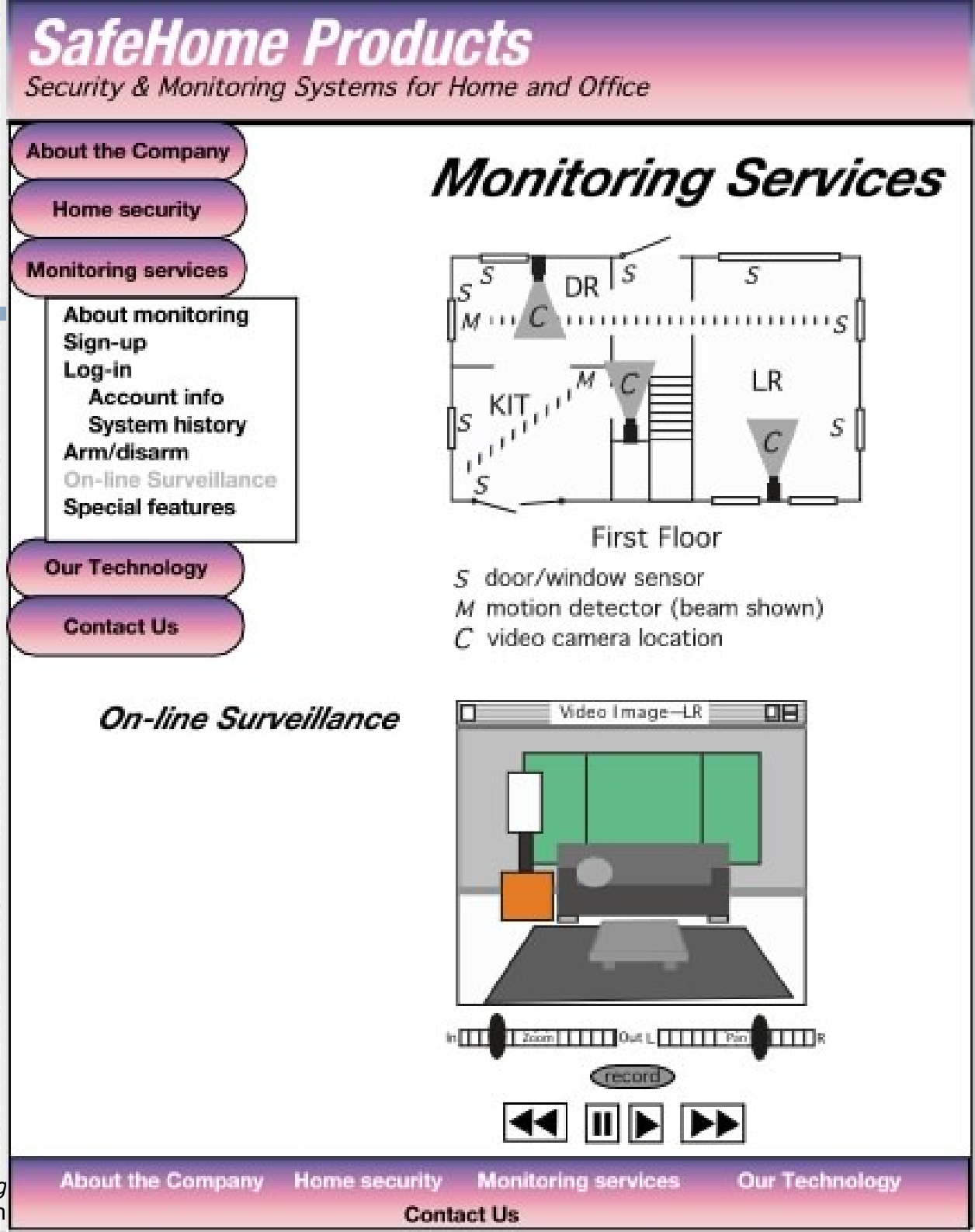
Translating Actions and Objects

- *From the use case on pp. 213 - 214*
  - *Accesses* the **SafeHome** system
  - *Enters* an **ID** and **Password** to allow remote access
  - *Displays* **FloorPlan** and **SensorLocations**
  - *Displays* **VideoCameraLocations** on floor plan
  - *Selects* **VideoCamera** for viewing
  - *Views* **VideoImages** (four frames per second)
  - *Pans* or *zooms* the **VideoCamera**
- Based on these actions and objects we create a design layout -->

Design Layout



Revising the Layout



Aesthetic Design - I

- Don’t be afraid of white space
- Emphasize content
- Organize layout elements from top-left to bottom-right.
- Don’t extend your real estate with the scrolling bar.
- Consider resolution and browser window size when designing the layout
- Design the layout for freedom of navigation.
- Don’t assume that the layout will be consistent across different display devices and browsers.
- If you use photos, make them small format with the option to enlarge
- If you want a cohesive layout, look, and feel across all WebApp pages, use a cascading style sheet (CSS).

Other Design Issues

- Response time
- “Help” facilities
- Error handling
- Accessibility
- Internationalization

Managing Risk

- Many problems can arise during a project
- Risk management focuses on understanding and managing these problems
  - Identify the risk; Assess its probability of occurrence; Estimate its impact; Establish a contingency plan
- Considers risk at two different levels of granularity

  (1) the impact of risk on the entire WebApp project, and

  (2) the impact of risk on the current WebApp increment
- Typical risks:
  - Is the time timeframe defined and reasonable?
  - Will the increments provide ongoing value for end users?
  - How will requests for change impact delivery schedules?
  - Is the available technology appropriate for the job?
  - Does the team have the right mix of skills to build this increment?

Identifying Risks

- Address the fundamental question: “What can go wrong?”
- Each team member is asked to make a list of risks
  - *People risks*
    - Potential problems that can be directly traced to some human action or failing
  - *Product risks*
    - Potential problems associated with WebApp content, functions, constraints, or performance
  - *Process risks*
    - Problems that are tied to the framework actions and tasks that have been chosen by the team

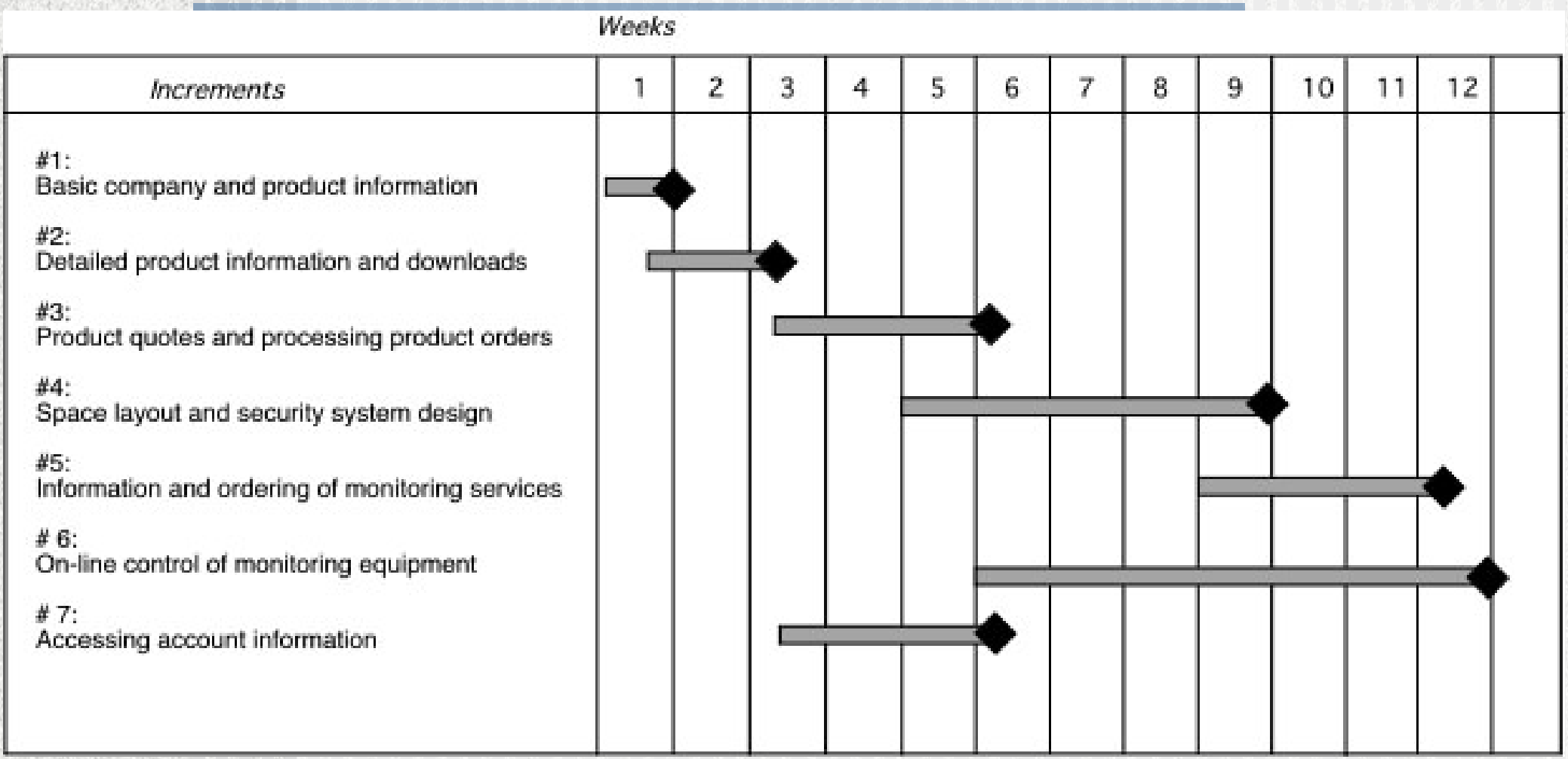
Risk Analysis

Risks	probability	impact
People		
Little XML experience on team	80%	3
Stakeholders uncooperative	60%	2
Senior manager may change mid-stream	40%	1
Product		
Informational content may be outdated	50%	2
Algorithms may not be adequately defined	80%	3
Security for WebApp more difficult than expected	80%	3
Database integration more difficult than expected	40%	3
Space def. capability more difficult than expected	70%	3
Process		
Not enough emphasis on communication	60%	2
Too many analysis tasks (too much time spent)	30%	1
Not enough emphasis on navigation design	40%	2
⋮	⋮	⋮

Developing a Schedule

- How do projects fall behind schedule?
  - *One day at a time*
- Approach:
  - List all WebE actions and tasks for an increment
  - Build a network that depicts interdependencies
  - Identify tasks that are critical
  - Track progress (especially critical tasks)
- The WebApp schedule evolves over time
- During the first iteration a macroscopic schedule is developed
  - Identify all increments and dates on which each will be deployed
- For each subsequent increment
  - The entry for the increment on the macroscopic schedule is refined into a detailed schedule

The Schedule



Estimating Time and Effort

- Two viable (and quick) approaches to detailed estimation
- *Usage scenario–based estimation*
  - Examines the usage scenarios for the increment, compares them to previous data on average effort (E
  - Adjust estimates based on perceived complexity
- *Product-process table*
  - First column lists, for all major WebE actions, content objects and functions for an increment
  - Subsequent columns lists estimates of effort (in person-days) required to perform each of the main WebE actions for each content object and function.
- Warning! The relationship between effort, people and time is NOT linear.

End…