Users of modern WebApps expect that robust content will be coupled with sophisticated functionality.

The advanced functionality will allow users to:
1) magnify their understanding of WebApp content,
2) characterize (see) content in different ways,
3) personalize their interaction with WebApp, and
4) provide added value to their website visit.

Functional design of WebApps is almost always component-based and compartmentalized (tagged, labeled).

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 (incompatibility) in Web browsers.
Security of Web Applications: Statistics

http://projects.webappsec.org/Web-Application-Security-Statistics#OverallData

Functionality Categories (1)

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; for example, Webster system)
Functionality Categories (2)

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 interfering)

http://www.smartwebby.com/web_site_design/dreamweaver_behaviors.asp

WebApp Functional Design: Functionality Levels and Design Tasks

- **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; it has strong relations with interaction design.

- **Application-level functionality** represents a lower-level design of internal functionality that may not be directly visible to users; it has strong relations with information design (content object design).

- 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.
Functional Design is based on Analysis Model

### Analysis Model

**1. Scenario-based models:**
- use-cases – text;
- use-cases – diagrams;
- activity diagrams;
- swim lane diagrams;

**2. Class models:**
- class diagrams;
- analysis packages;
- collaboration diagrams;

**3. Behavioral models:**
- state transition diagrams;
- sequence diagrams;

**4. Data-flow models:**
- data-flow diagrams;
- ERD diagrams;
- control-flow diagrams;
- processing narratives;

Functional Design: Modular Approach

- **Functional design** is a paradigm used to simplify the design of software applications.

- A functional design assures that **each modular part of software application has only one responsibility (one function)** and performs that responsibility with the minimum of side effects on other parts. In other words, functionally-designed modules tend to have low coupling.
Partitioning the WebApp

Motto: “Divide and Concur”

“Horizontal” and “vertical” partitioning are required

Vertical Partitioning

1. define separate branches of the module hierarchy for each major function
2. use control modules to coordinate communication between functions

---

function 1
(for ex: GUI)

function 2
(for ex: online database)

function 3
(for ex: security login, password)
Horizontal Partitioning: Factoring

1. decision making and work are stratified (or, separated by levels)

2. decision making modules reside at the top of the architecture

Functional Design: Main Components
Functional Design: Main Components

Getting Started (1):
Webster’s Use-Case Diagram
(use case scenarios)
Getting Started -- Outcomes (2): Webster’s Sequence (Swim lane) Diagrams

Behavioral Modeling

The behavioral model indicates how WebApp will respond to external events.

To create the model, the Web engineer must perform the following steps:

- Identify events that drive the interaction sequence and understand how these events relate to specific objects.
- Create a sequence of events for each use-case.
- Build a state transition diagram (STD) for the system.
- Review the behavioral model to verify accuracy and consistency.
State transition diagrams (STD) - one means of representing WebApp’s Behavioral Model.

**Stable States of the WebApp**

- **State**: a set of observable circumstances that characterizes the behavior of a system at a given time.
- **State transition**: the movement from one state to another.
- **Event**: an occurrence that causes the system to exhibit some predictable form of behavior.
- **Action**: process that occurs as a consequence of making a transition.
An Example: Research Seminar Registration STD

Top-level STD in UML format

An STD with details for the “Enrollment” state in UML format


An Example of Webster’s State Transition Diagram

User Functionality Design:
a set of detailed
- Collaboration/Interaction Diagrams,
- Swim lane Diagrams,
- Activity Diagrams,
- Sequence Diagrams,
- State Transition Diagrams
User Interaction Support: Detailed hierarchical structure of a GUI to provide designated functions

**Functional Modeling**

Following are the main functions:

1. Money Transaction
2. Money Deposit
3. Money Withdrawal
4. Alert System (E-mail Mobile)
5. Security
6. E-check
7. Online Bill Payment
8. Wire Transfer
9. Online Statements
10. Account Activity
11. Corporate Banking
12. Agricultural/Anat Rural Banking

**Application Functionality Design:**

- a set of detailed
- Class Diagrams
- Detailed Activity Diagrams
- Detailed Sequence Diagrams
- Detailed State Diagrams
Class Diagrams

- Class Diagrams are actively used on Class-based Modeling
- Represents objects system manipulates, operations applied to objects, and collaborations occurring between classes
- Elements of class model include: classes, data objects, attributes, operations, collaboration diagrams, packages, etc.
- Examine the problem statement and try to find nouns that fit the following categories and produce or consume information (i.e. grammatical parse)
  - External entities (systems, devices, people)
  - Things (e.g. reports, displays, letters, signals)
  - Events occurring during system operation
  - Roles (e.g. manager, engineer, salesperson)
  - Organizational units (e.g. division, group, team)
  - Places
  - Structures (e.g. sensors, vehicles, computers)
Class Diagrams: Examples

The upper part holds the name of the class.

The middle part contains the attributes of the class.

The bottom part gives the methods or operations the class can take or undertake.

Class Diagram – "Student" Class:
An example of the initial conceptual class diagram

Source: http://www.agilemodeling.com/artifacts/classDiagram.htm
Webster's Class Diagrams

**Grades**
- Course ID
- Course Name
- Section
- Description
- Semester Hour
- Grade
- Institutional

**Sem. Schedule**
- Course ID
- Course Name
- section
- Description
- Room Location
- Meeting Days
- Start Time
- End Time
- Sem. Hours
- Seat Available
- Waiting List
- Instructor Name

**Local Address**
- ID
- Street Address 1
- Street Address 2
- City
- State
- Zip
- Phone

**Home Address**
- ID
- Address Line 1
- Address Line 2
- Address Line 3
- Address Line 4
- Country
- Phone

**Bradley Registration #**
- ID
- Old Term
- New Term

**Other**
- Advisor
- Major
- Total Hours
- Class
- Class Rank
- GPA

**Degree Audit**
- Student ID
- any(id)
- Major
- Minor
- Hours Completed

**Hold**
- Student ID
- Hold Type
- Department

**Getting Started -- Outcomes (4):**
Webster's Data Flow Diagram
(Context Diagram)
Getting Started – Outcomes (4):
Online Banking Level-0
Data Flow Diagram (Context Diagram)

Functional Modeling
Following are the main functions:
1. Money Transaction
2. Money Deposit
3. Money Withdrawal
4. Alert System (Email, Mobile)
5. Security
6. E-Check
7. Online Bill Payment
8. Wire Transfer
9. Online Statements
10. Account Activity
11. Corporate Banking
12. Agricultural and Rural Banking

Application Tasks Support --
Outcomes are based on project’s evolution
(sketch, draft 1, draft 2, draft 3, etc.):

Webster’s Activity (Swim lane) Diagram:
(an example of a very detailed diagram)
Webster’s Detailed Activity Diagram: Page 3

Functional Architecture Design: a set of detailed
- Component Diagrams
- Deployment Diagrams
Component Diagrams

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component.

This illustrates the service consumer - service provider relationship between the two components.

Detailed functional modeling for WebApps is usually only carried out for those components that are extremely complex or extremely critical.
An Example: Webster Software System – Component-Based Design

System Level
(Webster System)

Level of Subsystems (Domains)
(Databases, GUI, Security, HELP, etc.)

Level of Elements or Components
<tables, forms, queries, reports, macros and modules, …)

Level of Sub-elements, Details
(for ex., attributes)
(ID, First Name, Last Name, DOB, YOA, status, …)

Examples of Tools for Component-Based Design
(Tools for Quick Database Design)

- phpMyAdmin
- ORBADA
- VOLCANO
- JOSE
- PIGEON PLANNER
- TIMECLOCK
- INGRES
- DELPHI
- OPENOFFICE BASE
- DB4OBJECTS
<table>
<thead>
<tr>
<th>#</th>
<th>Tool's name</th>
<th>Tool's Brief Description</th>
<th>Tool's main functions</th>
<th>Web address to download this tool</th>
<th>Required technical platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>phpMyAdmin</td>
<td>PhpMyAdmin is a tool written in PHP intended to handle the administration of MySQL over the Web.</td>
<td>It can create/drop/alter tables, delete/edit/add fields, execute any SQL statement, manage keys on fields.</td>
<td>LINK</td>
<td>You need PHP 5.2.0 or newer MySQL 5.0 or newer Web browser with cookies.</td>
</tr>
<tr>
<td>2</td>
<td>ORBADA</td>
<td>ORBADA is a database, SQL, query client using JDBC drivers.</td>
<td>It is a tool to administrate and manage database structure. It is for Oracle, SQLite, Firebird, HSQLDB, DerbyDB, MySQL and</td>
<td>LINK</td>
<td>JRE Windows/linux Drivers for databases</td>
</tr>
<tr>
<td>3</td>
<td>VOLCANO</td>
<td>Open source, universal database program.</td>
<td>User can make his own type of database and export it. Using own database files, *.dbf. Can use only with this program. Operating with databases in special GUI.</td>
<td>LINK</td>
<td>Windows xp or higher</td>
</tr>
<tr>
<td>4</td>
<td>JOSE</td>
<td>jose is a graphical Chess tool</td>
<td>You can store chess games in a database. You can view and edit games (including variations and comments). You can play against a plugged-in chess engine and use it for analysis.</td>
<td>LINK</td>
<td>All 32-bit MS Windows 95/98/NT/2000/XP, All POSIX (Linux/BSD/UNIX-like OSes), OS A, Linux, Win2K, WinXP, JRE</td>
</tr>
<tr>
<td>5</td>
<td>PIGEON PLANNER</td>
<td>Pigeon Planner aims to be a simple to use pigeon database</td>
<td>Add, edit, remove pigeons and view their details such as name, image, sex, colour, pedigree, relatives and results.</td>
<td>LINK</td>
<td>OS Independent (Written in an interpreted language)</td>
</tr>
<tr>
<td>6</td>
<td><strong>TIMECLOCK</strong></td>
<td>Timeclock is a simple yet effective time clock system. It allows you to track all employee time as well as upcoming vacations and more, it can also replace manual sign-in sheets and such.</td>
<td>LINK</td>
<td>atleast PHP 4.1.x, with mysql support — MySQL — Webserver — Web browser</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>INGRES</strong></td>
<td>Ingres is an opensource sql relational database management system intended to support large commercial and government applications. Ingres Corporation controls the development of Ingres and makes certified binaries available for download.</td>
<td>LINK</td>
<td>32-bit MS Windows (95/98/NT/2000/XP), Linux, Solaris is JRE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><strong>DELPHI</strong></td>
<td>It is an opensource database tool. It creates, administrates and alters database.</td>
<td>LINK</td>
<td>32-bit MS Windows (95/98), All 32-bit MS Windows (95/98/NT/2000/XP)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><strong>OPENOFFICE BASE</strong></td>
<td>BASE is a fully featured desktop database management system, designed to meet the needs of a broad array of users, from just tracking your personal CD collection, to producing a corporate monthly departmental sales report. BASE offers wizards to help users new to database design (or just new to BASE) to create Tables, Queries, Forms and Reports, along with a set of predefined table definitions for tracking Assets, Customers, Sales Orders, Invoices and much more.</td>
<td>LINK</td>
<td>Windows/Linux</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>DB4OBJECTS</strong></td>
<td>db4o (database for objects) is an embeddable open source object database. It is a tool to create and manage database.</td>
<td>LINK</td>
<td>32-bit MS Windows (95/98/NT/2000/XP), Linux, Solaris is JRE</td>
<td></td>
</tr>
</tbody>
</table>
3 Best Tools for Quick Database Design

- **phpMyAdmin**
  - browse and drop databases, tables, views, columns and indexes
  - create, copy, drop, rename and alter databases, tables, columns and indexes
  - load text files into tables
  - create and read dumps of tables
  - This tool offers GUI for maintaining users and creating databases and tables with out executing a single line of sql statements
  - Database can be managed from remote client using web browser.
  - Can execute batch files with sql statements.

- **ORBADA**
  - Running SQL queries
  - Viewing database objects
  - Export to CSV, Excel, DBF, XML, HTML, SQL (INSERT)
  - Editing stored procedures/functions
  - Many wizards
  - Easier for developers

- **VOLCANO**
  - create Database using GUI
  - Edit database tables
  - Delete tables or entities
<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>phpMyAdmin</td>
<td>It is web based - it runs on any server capable of handling PHP. Simple.</td>
<td>Some limitations in handling very large databases (this should not affect you unless you have tens of thousands of records).</td>
</tr>
<tr>
<td></td>
<td>Visual(no need to remember line commands). Browser based (no need for extra software). Widely available. Free</td>
<td>Somewhat slower than other systems such as Shell.</td>
</tr>
<tr>
<td>ORBADA</td>
<td>Can connect to any database like mysql, oracle, sqlite. Easy to browse data and edit them.</td>
<td>Requires drivers and connection strings to connect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No remote access</td>
</tr>
<tr>
<td>VOLCANO</td>
<td>Easy to create small databases. Data is stored in files which can be transferred. Easy to retrieve data.</td>
<td>Supports only 3 data types. Doesn't support large data. No drivers support</td>
</tr>
</tbody>
</table>

**phpMyAdmin (Lab and Demo)**
Overall Functional Architecture (Functional Model) as an Outcome of Functional design

Functional Architecture (Functional Model): Outcomes

- We got answers for 2 key questions:
  1. How do we partition the functionality into components that have clearly defined roles and interfaces?
  2. Where does each functional component exist, and what does it interact with?
Topic 11

WebE - Functional Design

Additional Information

Web Browser Incompatibility:
An Example with InterLabs Web-Lecturing Technology

Video/Audio

Text
PPT slides
Pictures

Email
Bulletin B.
Chat
Whiteboard
Video-conf.
Audio-conf.

Web-based Animation

Web-based programming and simulation
Getting Started: SafeHomeAssured.com Example

- 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: SafeHomeAssured.com Example

- 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.
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.

---

**Functional Modeling**

Following are the main functions:

1) Money Transaction
2) Money Deposit
3) Money Withdrawal
4) Alert System (E-mail Module)
5) Security
6) E-check
7) Online Bill Payment
8) Wire Transfer
9) Online Statements
10) Account Activity
11) Corporate Banking
12) Authorize.Net E-Check Banking

**Detailed hierarchical structure of a GUI to provide designated functions**
Deployment Diagrams
(Physical Architecture Model)

Deployment diagram depicts a static view of the run-time WebApp configuration of network processing nodes and the components that run on those nodes.

In other words, deployment diagrams show the hardware for your WebApp, the software that is installed on that hardware, and the middleware used to connect the disparate machines to one another.