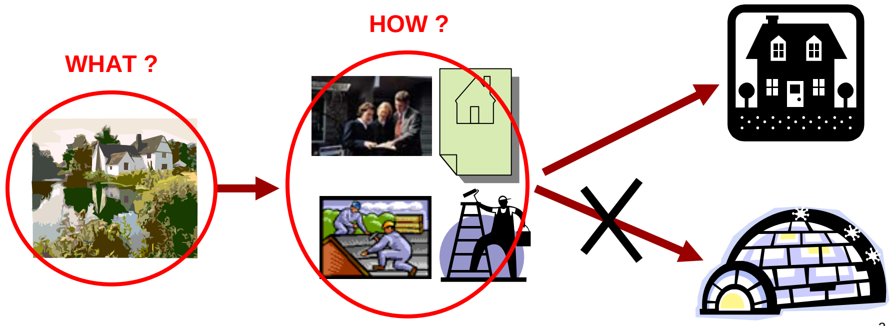
Topic 08

Web Applications Design: Components, Goals, Concepts

Web Design: What Is It About?

- **Web Design** - concepts, principles, and methods that are required (needed) to transform
  an understanding of WHAT the WebApp should do
  into
  a representation of HOW the WebApp should do.

- From **WebApp Analysis** Model to **WebApp Design Model**



WebApp Design: Art vs Engineering

- Jakob Nielsen states: “There are essentially two basic approaches to design:
  1. the artistic ideal of expressing yourself, and
  2. 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.”

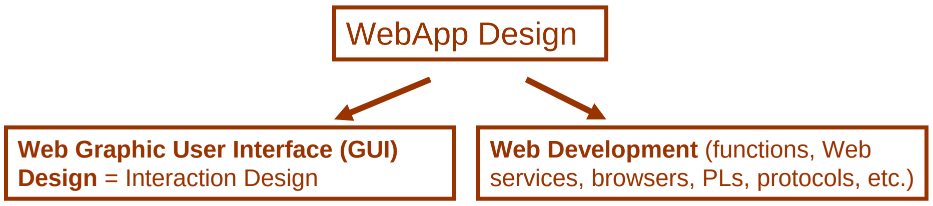
- However --
  - when WebApp content and functions are complex and very complex,
  - when the size of the WebApp encompasses hundreds of content objects, functions, use scenarios, etc. ,
  - when multiple (hundreds and thousands) people become involved in the design, and
  - when expected number of users is high and very high (thousands and millions),
  - when the success of the WebApp will have a direct impact on the success of the multi-million dollar business and/or users,

  WebApp design cannot and should not be taken lightly.

Conclusion: **Design of complex WebApp is a part of well- thought and well-structured process – software engineering.**

3

WebApp Design: Approaches


WebApp Design
Web Graphic User Interface (GUI) Design = Interaction Design
Web Development (functions, Web services, browsers, PLs, protocols, etc.)

With growing specialization in the information technology field, there is **a strong tendency to draw a clear line** between
a) Web graphic design, and
b) Web development (Web programming).

Web graphic interface design is a kind of **graphic design** intended for development and styling of objects of the Internet's information environment to provide them with high-end consumer features and aesthetic qualities.

Web development emphasizes the **functional features** of WebApp.

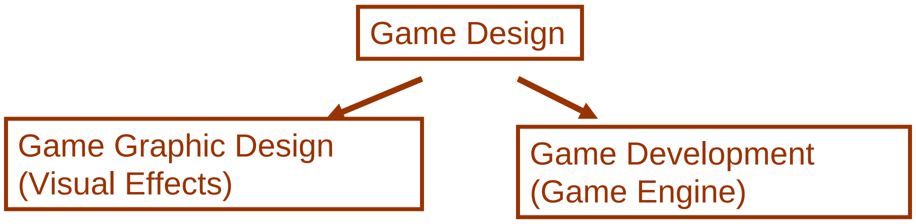
Web Design = Web Graphic Interface Design + Web Development

4

WebApp Design =

WebApp Graphic Design + WebApp Development

Analogy:
Computer Game Design Approaches

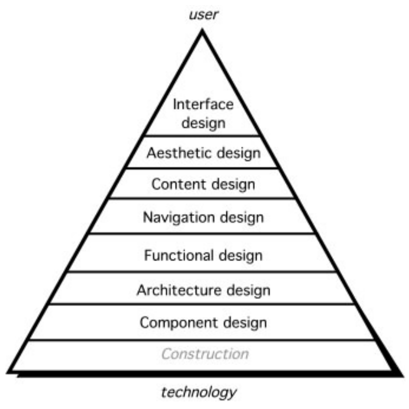

Game Design
Game Graphic Design (Visual Effects)
Game Development (Game Engine)

5

Web Applications Design: Components

6

Main Components of WebApp Design (WebApp Design Pyramid)



Interaction Design (interface design and aesthetic design = layout) = Interaction Model (GUI)

Information Design (content design and navigation design) = Information Model

Functional Design (overall behavior and functionality) = Functional Model

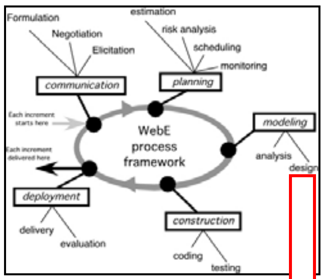
Technical Design (architectural design and component design) = Architectural or Structural Model

Additional useful components:

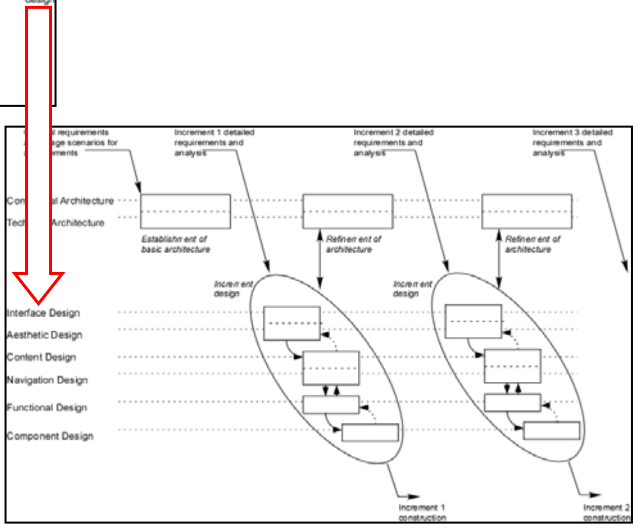
*) Design Patterns or Templates

**) Design and development Technologies and Tools

7



WebEng Framework Activities and WebApp Design Process (increment-based)



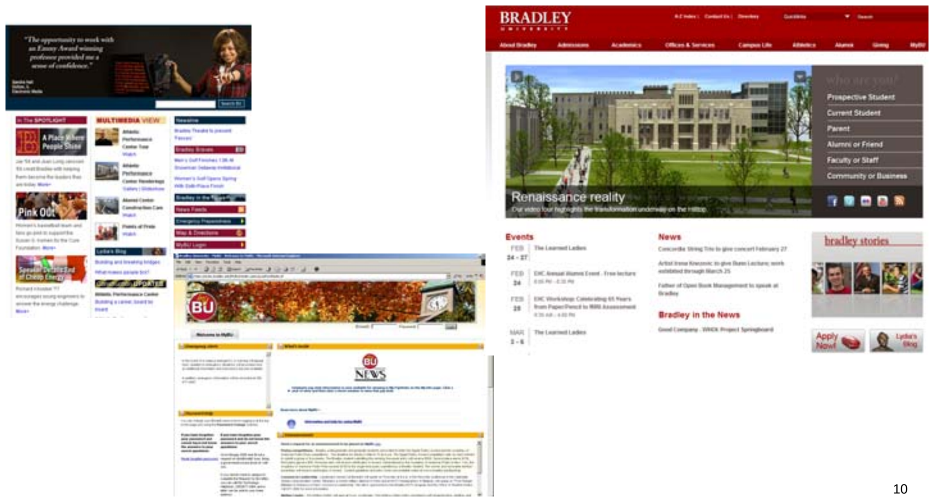
8

Web Applications - Graphic User Interface Design:

Goals

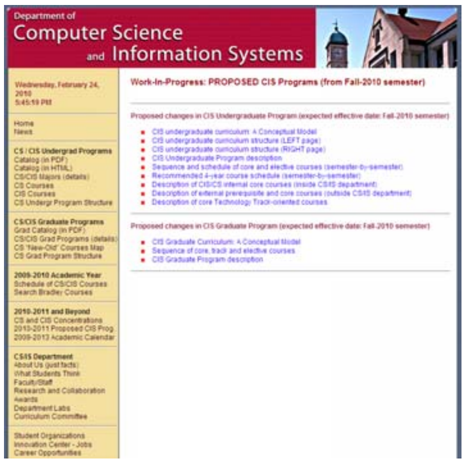
WebApp Design Goals

- **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.



WebApp Design Goals

- **Navigability.** Users should be able to understand how to move about the WebApp without having to search for navigation links or instructions.



Department of Computer Science and Information Systems

Wednesday, February 24, 2010 5:45:19 PM

Home
News

CS / CIS Undergrad Programs
Catalog (in PDF)
Catalog (in HTML)
CS/CIS Majors (details)
CS Courses
CIS Courses
CS Undergr Program Structure

CS/CIS Graduate Programs
Grad Catalog (in PDF)
CS/CIS Grad Programs (details)
CS 'New-Old' Courses Map
CS Grad Program Structure

2009-2010 Academic Year
Schedule of CS/CIS Courses
Search Bradley Courses

2010-2011 and Beyond
CS and CIS Concentrations
2010-2011 Proposed CIS Prog
2009-2013 Academic Calendar

CS/IS Department
About Us (just facts)
What Students Think
Faculty/Staff
Research and Collaboration
Awards
Department Labs
Curriculum Committee

Student Organizations
Innovation Center - Jobs
Career Opportunities

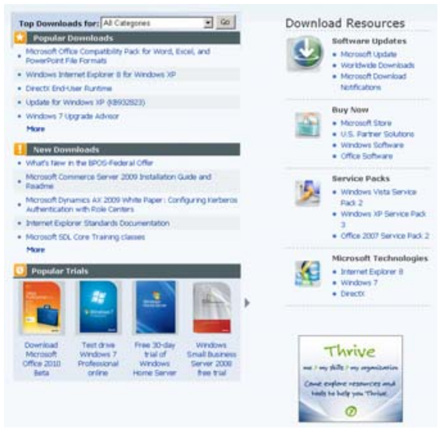
Work-In-Progress: PROPOSED CIS Programs (from Fall-2010 semester)

Proposed changes in CIS Undergraduate Program (expected effective date: Fall-2010 semester)

- CIS undergraduate curriculum: A Conceptual Model
- CIS undergraduate curriculum structure (LEFT page)
- CIS undergraduate curriculum structure (RIGHT page)
- CIS Undergraduate Program description
- Sequence and schedule of core and elective courses (semester-by-semester)
- Recommended 4-year course schedule (semester-by-semester)
- Description of CIS/CS internal core courses (inside CS/IS department)
- Description of external prerequisite and core courses (outside CS/IS department)
- Description of core Technology Track-oriented courses

Proposed changes in CIS Graduate Program (expected effective date: Fall-2010 semester)

- CIS Graduate Curriculum: A Conceptual Model
- Sequence of core, track and elective courses
- CIS Graduate Program description



Top Downloads for: All Categories Go

Popular Downloads
- Microsoft Office Compatibility Pack for Word, Excel, and PowerPoint File Formats
- Windows Internet Explorer 8 for Windows XP
- DirectX End-User Runtime
- Update for Windows XP (KB932823)
- Windows 7 Upgrade Advisor

More

New Downloads
- What's New in the BPOS-Federal Offer
- Microsoft Commerce Server 2009 Installation Guide and Readme
- Microsoft Dynamics AX 2009 White Paper : Configuring Kerberos Authentication with Role Centers
- Internet Explorer Standards Documentation
- Microsoft SDL Core Training classes

More

Popular Trials

Download Microsoft Office 2010 Beta | Test drive Windows 7 Professional online | Free 30-day trial of Windows Home Server | Windows Small Business Server 2008 free trial

Download Resources

Software Updates
- Microsoft Update
- Worldwide Downloads
- Microsoft Download Notifications

Buy Now
- Microsoft Store
- U.S. Partner Solutions
- Windows Software
- Office Software

Service Packs
- Windows Vista Service Pack 2
- Windows XP Service Pack 3
- Office 2007 Service Pack 2

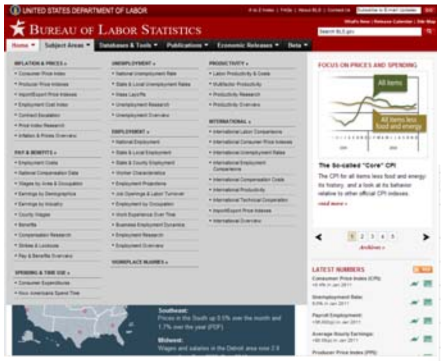
Microsoft Technologies
- Internet Explorer 8
- Windows 7
- DirectX

Thrive
me / my skills / my organization
Come explore resources and tools to help you Thrive.

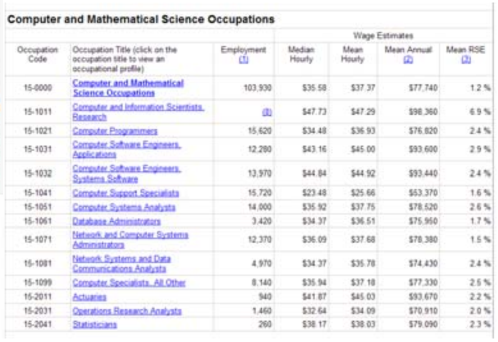
11

WebApp Design Goals

- **Identity.** The aesthetic, interface, and navigational design of a WebApp must be consistent with the application domain for which it is to be built.


UNITED STATES DEPARTMENT OF LABOR
Bureau of Labor Statistics
Home
Subject Areas
Databases & Tools
Publications
Economic Releases
Beta
FOCUS ON PRICES AND SPENDING
All items
All items less food and energy
The So-called "Core" CPI
LATEST NUMBERS

http://www.bls.gov/

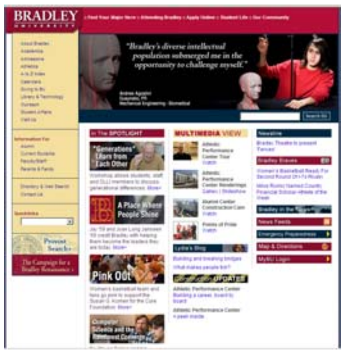


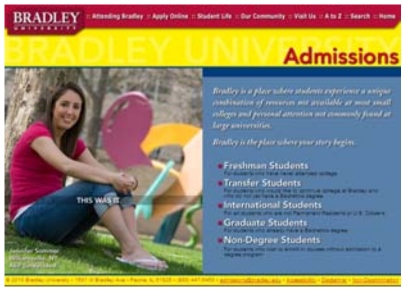
Computer and Mathematical Science Occupations

Occupation Code	Occupation Title (click on the occupation title to view an occupational profile)	Employment (1)	Wage Estimates			
			Median Hourly	Mean Hourly	Mean Annual (2)	Mean RSE (3)
15-0000	**Computer and Mathematical Science Occupations**	103,930	$35.58	$37.37	$77,740	1.2 %
15-1011	Computer and Information Scientists, Research	(8)	$47.73	$47.29	$98,360	6.9 %
15-1021	Computer Programmers	15,620	$34.48	$36.93	$76,820	2.4 %
15-1031	Computer Software Engineers, Applications	12,280	$43.16	$45.00	$93,600	2.9 %
15-1032	Computer Software Engineers, Systems Software	13,970	$44.84	$44.92	$93,440	2.4 %
15-1041	Computer Support Specialists	15,720	$23.48	$25.66	$53,370	1.6 %
15-1051	Computer Systems Analysts	14,000	$35.92	$37.75	$78,520	2.6 %
15-1061	Database Administrators	3,420	$34.37	$36.51	$75,950	1.7 %
15-1071	Network and Computer Systems Administrators	12,370	$36.09	$37.68	$78,380	1.5 %
15-1081	Network Systems and Data Communications Analysts	4,970	$34.37	$35.78	$74,430	2.4 %
15-1099	Computer Specialists, All Other	8,140	$35.94	$37.18	$77,330	2.5 %
15-2011	Actuaries	940	$41.87	$45.03	$93,670	2.2 %
15-2031	Operations Research Analysts	1,460	$32.64	$34.09	$70,910	2.0 %
15-2041	Statisticians	260	$38.17	$38.03	$79,090	2.3 %

WebApp Design Goals

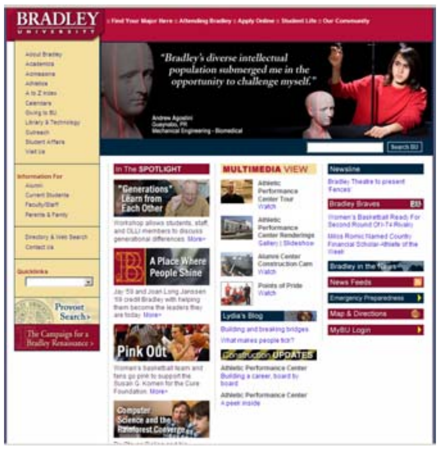
- **Consistency.**
  - Content should be constructed consistently
  - Graphic design (aesthetics) should present a consistent look
  - Architectural design should establish templates that lead to a consistent hypermedia navigation
  - Navigation mechanisms should be used consistently

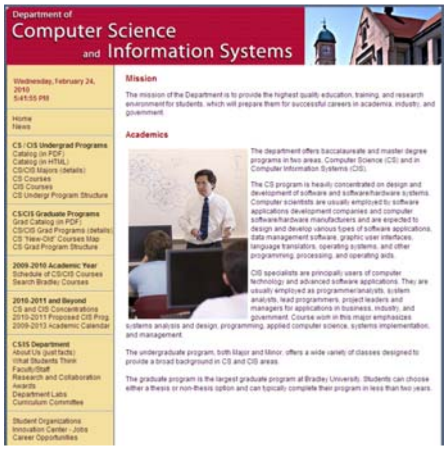

BRADLEY
"Bradley's diverse intellectual population submerged me in the opportunity to challenge myself."
In The SPOTLIGHT
MULTIMEDIA VIEW
Newsline
Pink Out



13

WebApp Design Goals: Do you see a consistency?

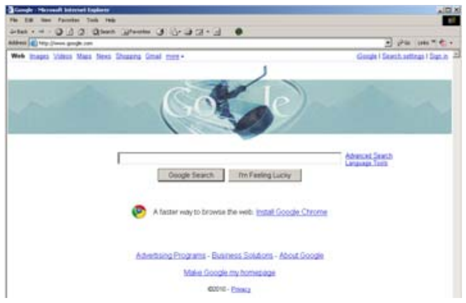

BRADLEY
"Bradley's diverse intellectual population submerged me in the opportunity to challenge myself."
In The SPOTLIGHT
MULTIMEDIA VIEW
Newsline
A Place Where People Shine
Pink Out

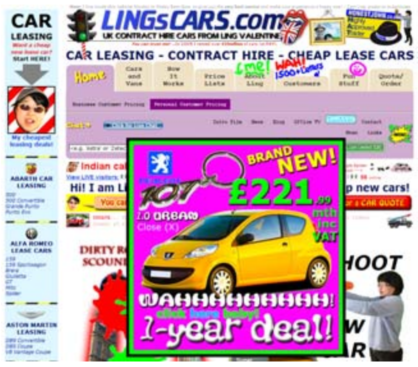


14

WebApp Design Goals: Simplicity

- **Simplicity.** Rather than *feature-bloat*, it is better to strive for moderation and simplicity (simple interface, simple navigation, simple structure, simple hierarchical model, relatively simple functions, etc.).





15

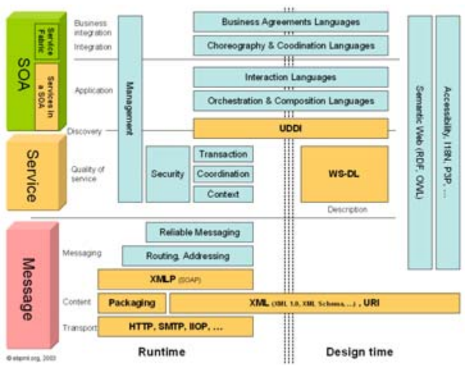
Web Applications – Functionality Development:

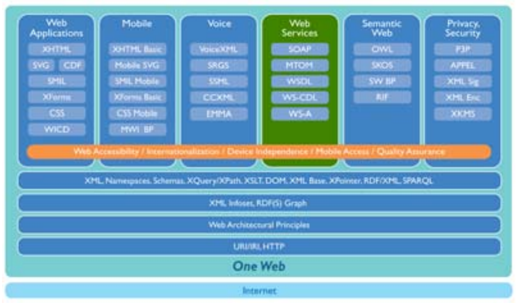
Goals

16

WebApp Development Goals

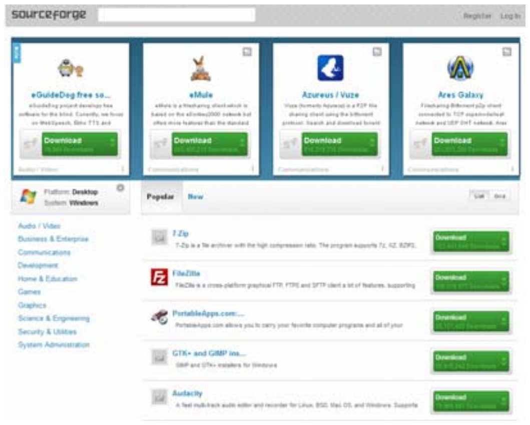
- **Main functions to be implemented** (search, retrieve, calculate, translate, convert, links to other WebApps, voice, Web services, Web security, etc.)
- **Web technologies/platforms to be used** (AJAX, Mash-Up, Web2.0, semantic Web,, etc.)
- **Programming Languages to be used** (Java, PHP, XML, DHTML, etc.)
- **Web services to be used and implemented.** Web services or Application





WebApp Design Goals

- **Robustness.** The user expects robust content and functions that are relevant to the user's needs.



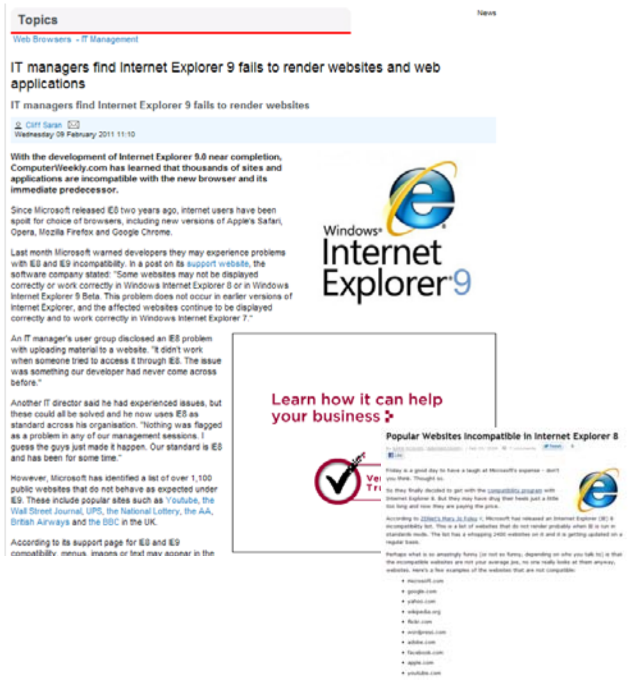
WebApp Development Goals

- **Security.** Web application frameworks may come with authentication and authorization frameworks, that enable the web server to identify the users of the application, and restrict access to functions based on some defined criteria.
- **Database access and mapping.** Many web application frameworks create a unified API to a database backend, enabling web applications to work with a variety of databases with no code changes, and allowing programmers to work with higher-level concepts.
- **Cashing.** Web caching is the caching of web documents in order to reduce bandwidth usage, server load, and perceived "lag".

19

WebApp Design Goals

- **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 of them.



Topics

News

Web Browsers - IT Management

IT managers find Internet Explorer 9 fails to render websites and web applications

IT managers find Internet Explorer 9 fails to render websites

Cliff Saran
Wednesday 09 February 2011 11:10

With the development of Internet Explorer 9.0 near completion, ComputerWeekly.com has learned that thousands of sites and applications are incompatible with the new browser and its immediate predecessor.

Since Microsoft released IE8 two years ago, internet users have been spoilt for choice of browsers, including new versions of Apple's Safari, Opera, Mozilla Firefox and Google Chrome.

Last month Microsoft warned developers they may experience problems with IE8 and IE9 incompatibility. In a post on its support website, the software company stated: "Some websites may not be displayed correctly or work correctly in Windows Internet Explorer 8 or in Windows Internet Explorer 9 Beta. This problem does not occur in earlier versions of Internet Explorer, and the affected websites continue to be displayed correctly and to work correctly in Windows Internet Explorer 7."

An IT manager's user group disclosed an IE8 problem with uploading material to a website. "It didn't work when someone tried to access it through IE8. The issue was something our developer had never come across before."

Another IT director said he had experienced issues, but these could all be solved and he now uses IE8 as standard across his organisation. "Nothing was flagged as a problem in any of our management sessions. I guess the guys just made it happen. Our standard is IE8 and has been for some time."

However, Microsoft has identified a list of over 1,100 public websites that do not behave as expected under IE9. These include popular sites such as Youtube, the Wall Street Journal, UPS, the National Lottery, the AA, British Airways and the BBC in the UK.

According to its support page for IE8 and IE9 compatibility, menus, images or text may appear in the

Windows Internet Explorer 9
Learn how it can help your business

Popular Websites Incompatible in Internet Explorer 8

Friday is a good day to have a laugh at Microsoft's expense - don't you think. Thought so.

So they finally decided to get with the compatibility program with Internet Explorer 8. But they may have drug their heels just a little too long and now they are paying the price.

According to ZDNet's Mary Jo Foley, Microsoft has released an Internet Explorer (IE) 8 incompatibility list. This is a list of websites that do not render probably when IE is run in standards mode. The list has a whopping 2400 websites on it and it is getting updated on a regular basis.

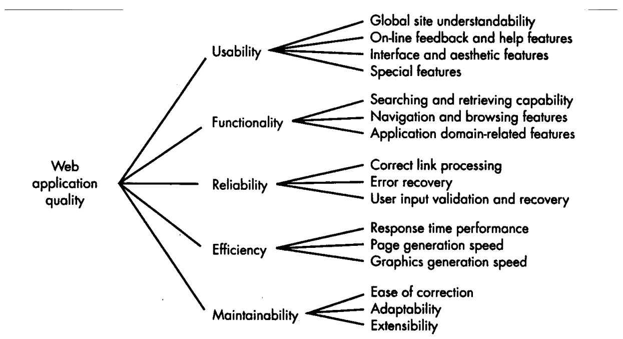
Perhaps what is so amazingly funny (or not so funny, depending on who you talk to) is that the incompatible websites are not your average joe, no one really looks at them anyway, websites. Here's a few examples of the websites that are not compatible:

- microsoft.com
- google.com
- yahoo.com
- wikipedia.org
- flickr.com
- wordpress.com
- adobe.com
- facebook.com
- apple.com
- youtube.com

20

Web Applications Development and Quality Issues

WebApp Design and Quality Issues


Web application quality
Usability
Global site understandability
On-line feedback and help features
Interface and aesthetic features
Special features
Functionality
Searching and retrieving capability
Navigation and browsing features
Application domain-related features
Reliability
Correct link processing
Error recovery
User input validation and recovery
Efficiency
Response time performance
Page generation speed
Graphics generation speed
Maintainability
Ease of correction
Adaptability
Extensibility

From end user's point of view:

- Performance
- Response time and latency (delay)
- Content
- WebApp versions (upgrades? How quickly?)
- Structural (how well parts of WebApp hold together?)

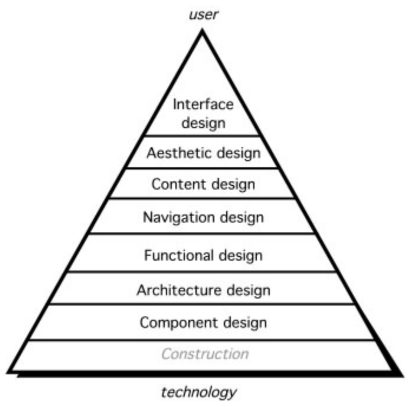
Software Design Concepts

Software Design: Fundamental Concepts (in CS590)

1. Abstraction	still a model of reality (data, procedure, control, etc.); errors or loss of details may occur
2. Architecture	the overall structure of the software
3. Patterns	”conveys the essence” of a proven design solution (best cases, re-usable designs)
4. Modularity	compartmentalization of data and function
5. Information Hiding	controlled interfaces
6. Functional Independence	single-minded function and low coupling
7. Refinement	elaboration of detail for all abstractions
8. Re-factoring	a reorganization technique that simplifies the design

In the next several lectures:

Main Components of WebApp Design = Design Models



Interaction Design (interface design and aesthetic design = layout) = **Interaction Model (GUI)**

Information Design (content design and navigation design) = **Information Model**

Functional Design (overall behavior and functionality) = **Functional Model**

Technical Design (architectural design and component design) = **Architectural or Structural Model**

Additional useful components:

*) Design Patterns or Templates

**) Design and development Technologies and Tools

25

Web Applications Design: Components, Goals, Concepts

Additional information.

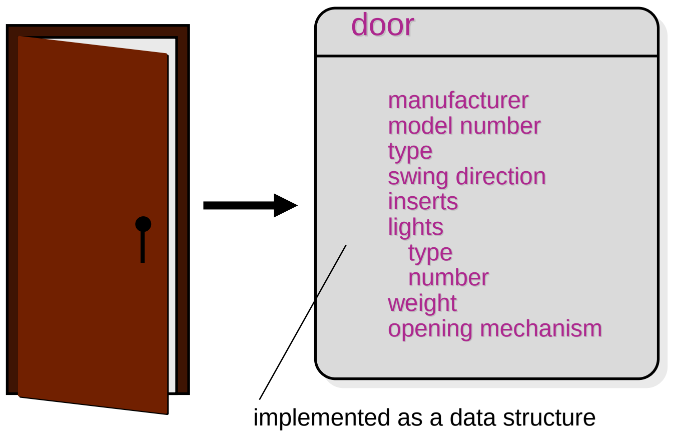
26

Software Design: Fundamental Concepts

1. Abstraction	data, procedure, control
2. Architecture	the overall structure of the software
3. Patterns	”conveys the essence” of a proven design solution (best cases, re-usable designs))
4. Modularity	compartmentalization of data and function
5. Information Hiding	controlled interfaces
6. Functional Independence	single-minded function and low coupling
7. Refinement	elaboration of detail for all abstractions
8. Re-factoring	a reorganization technique that simplifies the design

27

1.1. Data Abstraction

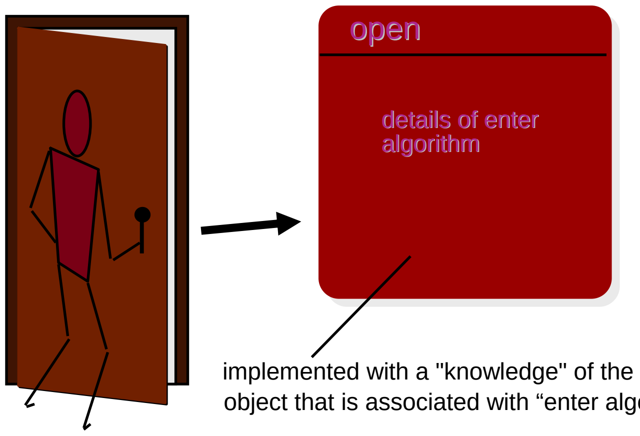


Abstraction - allows designers to simplify a problem and focus on solving a problem without being concerned about irrelevant lower level details

(an example: a class – a named collection of data objects)

28

1.2. Procedural Abstraction



implemented with a "knowledge" of the object that is associated with “enter algorithm”

Abstraction - allows designers to simplify a problem and focus on solving a problem without being concerned about irrelevant lower level details

(en example: procedural abstraction: a subroutine – a named sequence of events)

29

2. Architecture

“The overall structure of the software and the ways in which that structure provides conceptual integrity for a system.”

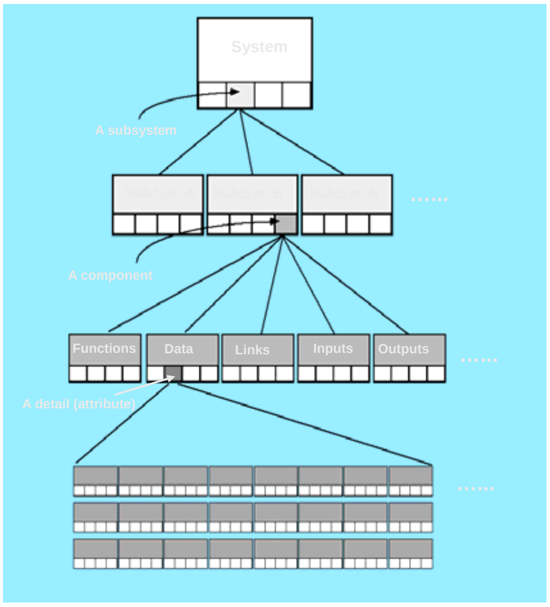
Structural properties. This aspect of the architectural design representation defines the components of a system (e.g., modules, objects, filters) and the manner in which those components are packaged and interact with one another. For example, objects are packaged to encapsulate both data and the processing that manipulates the data and interact via the invocation of methods.

Extra-functional properties. The architectural design description should address how the design architecture achieves requirements for performance, capacity, reliability, security, adaptability, and other system characteristics.

Families of related systems (reusable architectural building blocks). The architectural design should draw upon repeatable patterns that are commonly encountered in the design of families of similar systems. In essence, the design should have the ability to reuse architectural building blocks.

30

System Engineering: A Structural View



System Level
(World View)

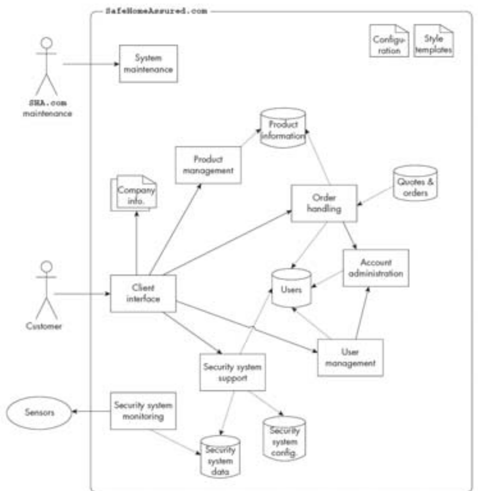
Level of Subsystems (Domains)
(Subsystem or Domain View)

Level of Elements or Components
(Element or Component View)

Level of Sub-elements,
Details (for ex., attributes)
(Detail View)

31

Conceptual (Logic) Architecture of the *SafeHomeAssured.com* WebApp



32

3. Patterns

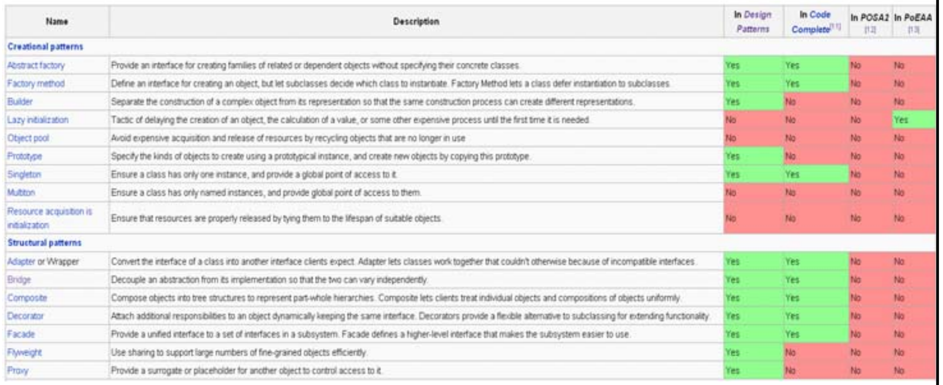
Design Pattern Template:

Pattern name	describes the essence of the pattern in a short but expressive name
Intent	describes the pattern and its functions
Also-known-as	lists any synonyms for the pattern
Motivation	provides an example of the problem
Applicability	notes specific design situations in which the pattern is applicable
Structure	describes the classes that are required to implement the pattern
Participants	describes the responsibilities of the classes that are required to implement the pattern
Collaborations	describes how the participants collaborate to carry out their responsibilities
Consequences	describes the "design forces" that affect the pattern and the potential trade-offs that must be considered when the pattern is implemented
Related patterns	cross-references related design patterns

*) Patterns-based SE, patterns-based analysis, patterns-based software development, pattern languages, etc. – still a lot of research needed, a lot of Ph.D. dissertations

33

3a. Patterns (details)



Name	Description	In Design Patterns	In Code Complete[11]	In POSA2 [12]	In PoEAA [13]
Creational patterns					
Abstract factory	Provide an interface for creating families of related or dependent objects without specifying their concrete classes.	Yes	Yes	No	No
Factory method	Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.	Yes	Yes	No	No
Builder	Separate the construction of a complex object from its representation so that the same construction process can create different representations.	Yes	No	No	No
Lazy initialization	Tactic of delaying the creation of an object, the calculation of a value, or some other expensive process until the first time it is needed.	No	No	No	Yes
Object pool	Avoid expensive acquisition and release of resources by recycling objects that are no longer in use	No	No	No	No
Prototype	Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.	Yes	No	No	No
Singleton	Ensure a class has only one instance, and provide a global point of access to it.	Yes	Yes	No	No
Multiton	Ensure a class has only named instances, and provide global point of access to them.	No	No	No	No
Resource acquisition is initialization	Ensure that resources are properly released by tying them to the lifespan of suitable objects.	No	No	No	No
Structural patterns					
Adapter or Wrapper	Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.	Yes	Yes	No	No
Bridge	Decouple an abstraction from its implementation so that the two can vary independently.	Yes	Yes	No	No
Composite	Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.	Yes	Yes	No	No
Decorator	Attach additional responsibilities to an object dynamically keeping the same interface. Decorators provide a flexible alternative to subclassing for extending functionality.	Yes	Yes	No	No
Facade	Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.	Yes	Yes	No	No
Flyweight	Use sharing to support large numbers of fine-grained objects efficiently.	Yes	No	No	No
Proxy	Provide a surrogate or placeholder for another object to control access to it.	Yes	No	No	No

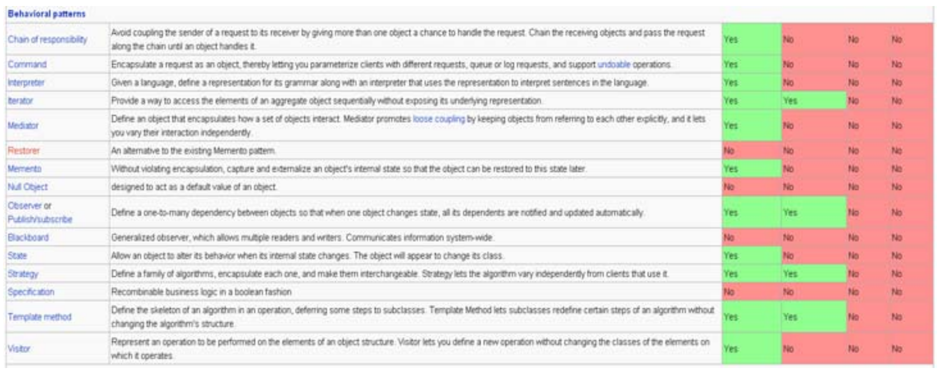
A design pattern is a general reusable solution to a commonly occurring problem in software design.

A design pattern is not a finished design that can be transformed directly into code.

It is a description or template for how to solve a problem that can be used in many different situations.

34

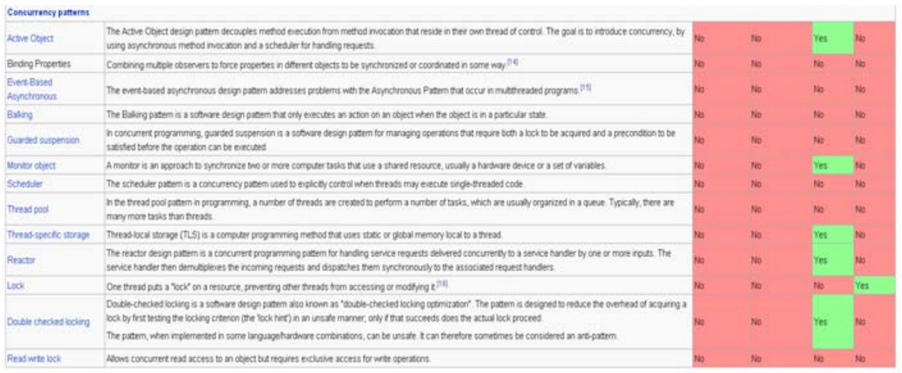
3a. Patterns (details)



Behavioral patterns					
Chain of responsibility	Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.	Yes	No	No	No
Command	Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.	Yes	No	No	No
Interpreter	Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.	Yes	No	No	No
Iterator	Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.	Yes	Yes	No	No
Mediator	Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.	Yes	No	No	No
Restorer	An alternative to the existing Memento pattern.	No	No	No	No
Memento	Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.	Yes	No	No	No
Null Object	designed to act as a default value of an object.	No	No	No	No
Observer or Publish/subscribe	Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.	Yes	Yes	No	No
Blackboard	Generalized observer, which allows multiple readers and writers. Communicates information system-wide.	No	No	No	No
State	Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.	Yes	No	No	No
Strategy	Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.	Yes	Yes	No	No
Specification	Recombinable business logic in a boolean fashion	No	No	No	No
Template method	Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.	Yes	Yes	No	No
Visitor	Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.	Yes	No	No	No

35

3a. Patterns (details)

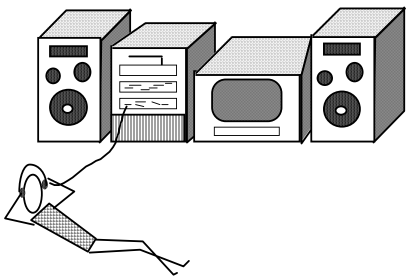


Concurrency patterns					
Active Object	The Active Object design pattern decouples method execution from method invocation that reside in their own thread of control. The goal is to introduce concurrency, by using asynchronous method invocation and a scheduler for handling requests.	No	No	Yes	No
Binding Properties	Combining multiple observers to force properties in different objects to be synchronized or coordinated in some way.[14]	No	No	No	No
Event-Based Asynchronous	The event-based asynchronous design pattern addresses problems with the Asynchronous Pattern that occur in multithreaded programs.[15]	No	No	No	No
Balking	The Balking pattern is a software design pattern that only executes an action on an object when the object is in a particular state.	No	No	No	No
Guarded suspension	In concurrent programming, guarded suspension is a software design pattern for managing operations that require both a lock to be acquired and a precondition to be satisfied before the operation can be executed.	No	No	No	No
Monitor object	A monitor is an approach to synchronize two or more computer tasks that use a shared resource, usually a hardware device or a set of variables.	No	No	Yes	No
Scheduler	The scheduler pattern is a concurrency pattern used to explicitly control when threads may execute single-threaded code.	No	No	No	No
Thread pool	In the thread pool pattern in programming, a number of threads are created to perform a number of tasks, which are usually organized in a queue. Typically, there are many more tasks than threads.	No	No	No	No
Thread-specific storage	Thread-local storage (TLS) is a computer programming method that uses static or global memory local to a thread.	No	No	Yes	No
Reactor	The reactor design pattern is a concurrent programming pattern for handling service requests delivered concurrently to a service handler by one or more inputs. The service handler then demultiplexes the incoming requests and dispatches them synchronously to the associated request handlers.	No	No	Yes	No
Lock	One thread puts a "lock" on a resource, preventing other threads from accessing or modifying it.[16]	No	No	No	Yes
Double checked locking	Double-checked locking is a software design pattern also known as "double-checked locking optimization". The pattern is designed to reduce the overhead of acquiring a lock by first testing the locking criterion (the 'lock hint') in an unsafe manner; only if that succeeds does the actual lock proceed. The pattern, when implemented in some language/hardware combinations, can be unsafe. It can therefore sometimes be considered an anti-pattern.	No	No	Yes	No
Read write lock	Allows concurrent read access to an object but requires exclusive access for write operations.	No	No	No	No

36

4. Modular Design
(easier to build, easier to change, easier to fix…)

easier to build, easier to change, easier to fix ..



*) 1969, Boeing 747

*) modern cars (standard modules)

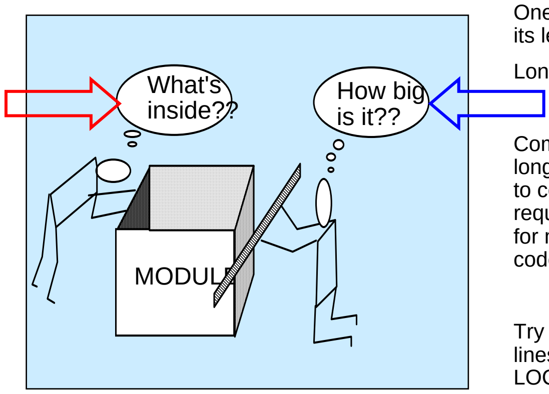
*) reusable learning objects (modules of online courses)

Microsoft Design Center:

Industrialization and modular object-oriented software development has theoretically made it easy to tear things down and replace them.

37

Sizing Modules: Two Views



One important feature of a module is its length.

Longer is not better.

Compilation time is expensive; the longer the module, the longer it takes to compile. Most program changes that require a recompilation of a module are for minor changes, not involving lots of code lines.

Try to restrict your modules to 400 lines of code; optimal length: 60-300 LOC per module

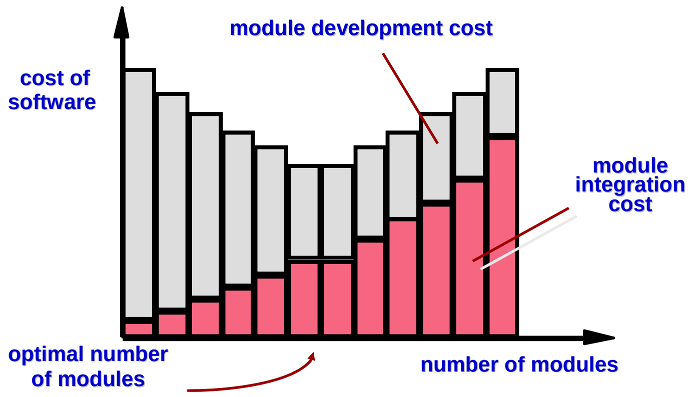
If the module needs to be longer, split it into multiple modules, all with a related name.

Source: http://www.sltf.com/ssm/ssm.htm#Modules

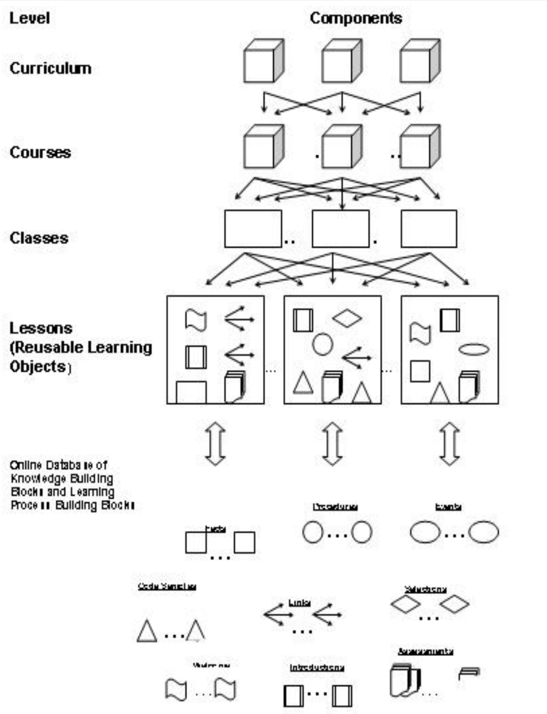
38

Software Modularity: Trade-offs

What is the "right" number of modules for a specific software design?



39



Level
Components
Curriculum
Courses
Classes
Lessons (Reusable Learning Objects)
Online Databases of Knowledge Building Blocks and Learning Process Building Blocks
Facts
Procedures
Events
Code Samples
Links
Solutions
Assignments
Visuals
Introductions

Modular Approach to Learning Content Design:

System Level

Level of Subsystems (Domains)

Level of Components

Level of elements

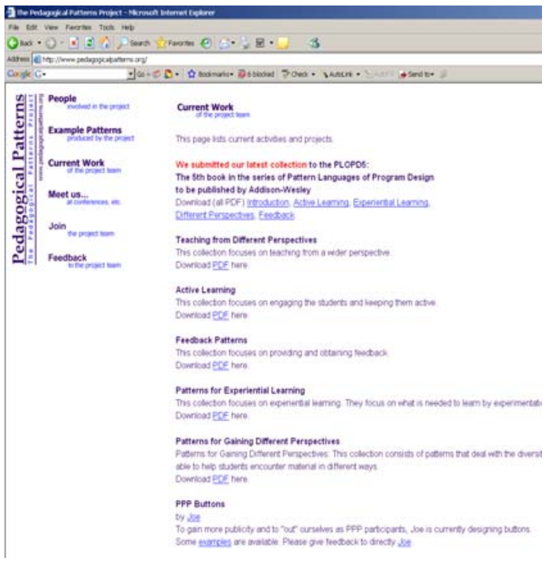
40

BTW: Pedagogical Patterns (details)

Pedagogical Patterns are high-level patterns that have been recognized in many areas of training and pedagogy such as group work, software design, human computer interaction, education and others.

The concept is an extension of pattern languages. In both cases, the patterns seek to foster best practices of teaching.

You may think about patterns as some kind of metadata – a top-level description of data about data.


The Pedagogical Patterns Project - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.pedagogicalpatterns.org/
Pedagogical Patterns
The Pedagogical Patterns Project
People involved in the project
Example Patterns produced by the project
Current Work of the project team
Meet us... at conferences, etc.
Join the project team
Feedback to the project team

Current Work
of the project team

This page lists current activities and projects.

We submitted our latest collection to the PLOPD5:
The 5th book in the series of Pattern Languages of Program Design to be published by Addison-Wesley
Download (all PDF) Introduction, Active Learning, Experiential Learning, Different Perspectives, Feedback

Teaching from Different Perspectives
This collection focuses on teaching from a wider perspective.
Download PDF here.

Active Learning
This collection focuses on engaging the students and keeping them active.
Download PDF here.

Feedback Patterns
This collection focuses on providing and obtaining feedback.
Download PDF here.

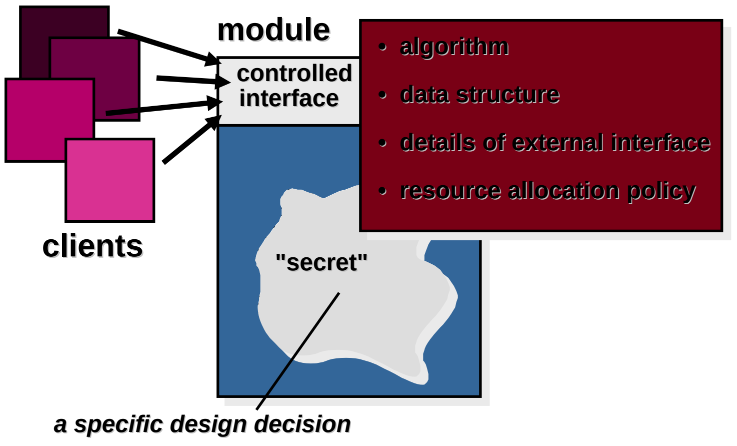
Patterns for Experiential Learning
This collection focuses on experiential learning. They focus on what is needed to learn by experimentati
Download PDF here.

Patterns for Gaining Different Perspectives
Patterns for Gaining Different Perspectives: This collection consists of patterns that deal with the diversi
able to help students encounter material in different ways.
Download PDF here.

PPP Buttons
by Joe
To gain more publicity and to "out" ourselves as PPP participants, Joe is currently designing buttons.
Some examples are available. Please give feedback to directly Joe.

41

5. Information Hiding (or, Encapsulation in Programming)



- algorithm
- data structure
- details of external interface
- resource allocation policy

42

Why Information Hiding?

- leads to encapsulation—an attribute of high quality design
- results in higher quality software
- limits the global impact of local design decisions
- emphasizes communication through well-structured and controlled interfaces
- reduces the likelihood of “side effects”
- discourages the use of global data

Advantages of Encapsulation in Programming

- Large software programs may be split into manageable modules
- Implementation details are hidden (isolated); as a result, no need to waste your time
- Subprograms and programs become more portable, and, probably, re-usable by external users
- Development time is shortened due to well-written, well-structured, well-tested sub-programs (functions, lists, files, etc.)

6. Functional Independence

- Functional independence is achieved by developing **modules with "single-minded" function** and an "aversion" (disliking) to excessive interaction with other modules.
- ***Cohesion*** is an indication of the relative functional strength of a module.
  - A cohesive module performs a single task, requiring little interaction with other components in other parts of a program. Stated simply, a cohesive module should (ideally) do just one thing.
- ***Coupling*** is an indication of the relative interdependence among modules.
  - Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across the interface.

System Concepts

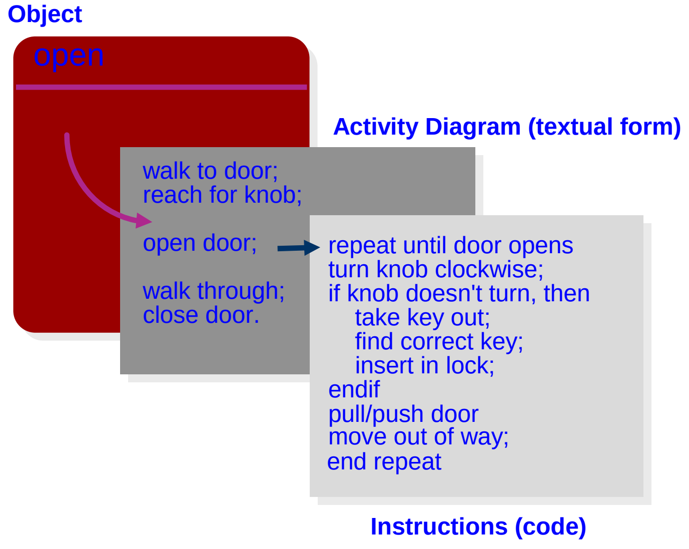
Once we have recognized something as a system, **how do we understand the system?**

Important system concepts include:

Modularity	is dividing a system into parts/chunks/modules of relatively uniform size.
Decomposition	is the process of breaking down a system into its component parts.
Coupling	is the extent to which subsystems are dependent on each other.
Cohesion	is the extent to which a system or a subsystem performs a single function.
Open system:	a system that interacts freely with its environment, taking input and returning output.
Closed system:	a system that is cut off from its environment and does not interact with it.

45

7. Stepwise Refinement



```
walk to door;
reach for knob;

open door;

walk through;
close door.
```

```
repeat until door opens
turn knob clockwise;
if knob doesn't turn, then
    take key out;
    find correct key;
    insert in lock;
endif
pull/push door
move out of way;
end repeat
```

46

WebApp Design: Main Components

- The WebApp Design Model encompasses
  1. content,
  2. aesthetics (artistic features),
  3. interface,
  4. architecture (structural components),
  5. navigation (functions), and
  6. component-level (objects) 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 **WebApp Design Model** will lead to an effective implementation should be **continuously assessed**

47

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**.

48

Developing the architecture of WebApp

- 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 of WebApp

- 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.

Languages for Logic Modeling of WebApp

- HDM - W2000
- RMM
- OOHDM
- ARANEUS
- STRUDEL
- TIRAMISU
- WebML
- Hera
- UML Web Application Extension
- UML-based Web Engineering (UWE)
- ACE
- WebArchitect
- OO-H

51

Technical Architecture

- Shows how the conceptual architecture can be mapped into specific technical components
- Any decision made about how one component might map into the technical architecture will affect the decisions about other components
  - For example, the WebE team may choose to design **SafeHomeAssured.com** in a way that stores product information as XML files. Later, the team discovers that the content management system doesn't easily support access to XML content, but rather assumes that the content will be stored in a conventional relational database. One component of the technical architecture conflicts with constraints imposed by another component.

52