H Design Patterns & Frameworks

H.1 Overview

Design is hard.
One way to avoid the act of design is to reuse existing designs
[Ralph Johnson]

- Design patterns
- Class libraries
- Frameworks — What they are, How they work, Benefits
- Types of Frameworks
- CORBA & Frameworks
- Java & Frameworks

H.3 Design Patterns

Each pattern describes a problem which occurs over and over again in our environment,
and then describes the core of the solution to that problem,
in such a way that you can use this solution a million times over,
whithout ever doing it the same way twice
[Christopher Alexander, 1977 —
talking about patterns in buildings and towns]

- Design patterns represent *solutions* to *problems* that arise when developing software within a particular *context*
- Patterns capture the static and dynamic *structure* and *collaboration* among key *participants* in software designs
- Classes / class libraries = reuse of implementations
  Patterns = reuse of designs

H.2 References

EJB99. Enterprise JavaBeans Overview. http://java.sun.com/products/ejb/index.html

GHJ+97. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

JoF88. R.E Johnson, B. Foote: Designing Reusable Classes, *Journal of Object-Oriented Programming*, June 1988, ftp://st.cs.uiuc.edu/pub/papers/frameworks/designing-reusable-classes.ps.

Joh92. R.E Johnson, Documenting Frameworks Using Patterns, OOPSLA '92 Proceedings, 1992, ftp://st.cs.uiuc.edu/pub/patterns/papers/documenting-frameworks.ps

Joh93. R.E Johnson, How to Design Frameworks, Tutorial Notes, OOPSLA '93, Washington, 1993, ftp://st.cs.uiuc.edu/pub/papers/frameworks/OOPSLA93-frmwk-tut.ps

MoM97. Thomas Mowbray, Raphael Malveau. *CORBA Design Patterns*. Wiley Computer Publishing, 1997.

OHE96. Robert Orfali, Dan Harkey, Jeri Edwards: *The Essential Distributed Objects Survival Guide*, John Wiley, New York, 1996.

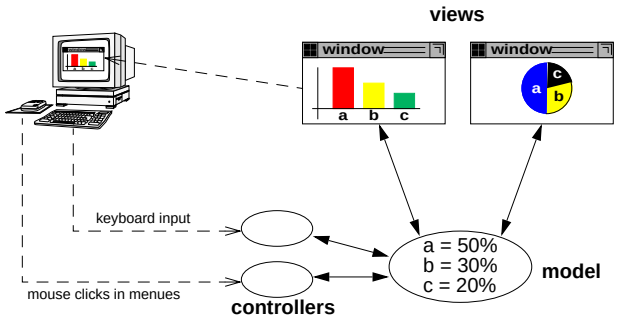
SFO98. San Francisco Project Overview. http://http://www.software.ibm.com/ad/sanfrancisco/

Vli98. John Vlissides. *Pattern Hatching: Design Patterns Applied*, Addison-Wesley (Software Patterns Series), 1998.

Patterns@WWW. The patterns homepage: http://hillside.net/patterns/

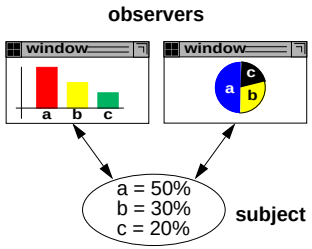
1 Example: Smalltalk's Model/View/Controller

- Basic concept for building user interfaces in Smalltalk
  - ➤ Model: the application object
  - ➤ View: screen presentation of application object
  - ➤ Controller: takes user input and modifies application object



1 Example: Smalltalk's Model/View/Controller (2)

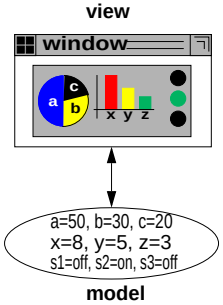
- **Observer pattern**
  - view decoupled from model
  - view = observer
  - model = subject
  - subscribe/notify protocol
  - observer independent from subject



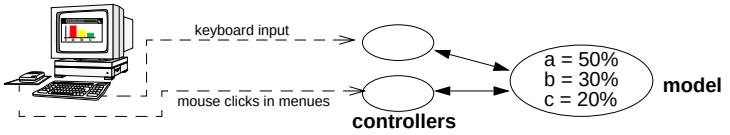
1 Example: Smalltalk's Model/View/Controller (3)

- MVC allows nested views
  - a nested view is also a view (`CompositeView` = subclass of `View`)

- **Composite design pattern**
  - class hierarchy
  - primitive objects (e.g. line, circle, polygon)
  - compatible composite objects that assemble the primitive objects into more complex objects (e.g. picture)
  - composite object may be used instead of primitive object



1 Example: Smalltalk's Model/View/Controller (4)



- User input handled by separate controller object that translates it into method invocations at model

- **Strategy pattern**
  - strategy object represents an algorithm
  - may be replaced statically or dynamically — independent from model
  - strategy objects may encapsulate different variants of an algorithm
  - strategy objects may encapsulate complex data structures

2 Elements of a Design Pattern

- Pattern name
- Problem
  - describes the problem and its context
- Solution
  - describes
    - the elements,
    - their relationships,
    - responsibilies,
    - collaborations
- Consequences
  - results
  - trade-offs

  (space, time, language- and implementation issues, …)

3 Design Pattern Space

Table 1:

		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter (class)	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

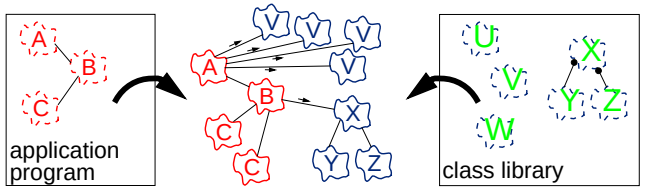
2 What is a Framework?

- Framework = design for a set of applications
  - ➥ design of a set of objects that collaborate to carry out a set of responsibilities
  - ➥ a way to reuse high-level design
- Framework =
  set of classes
  + rules how the objects play together
  + definition of the interfaces in the game (how can I join)
  + definition of interfaces to the game (interaction with the outside world)
  + definition of the goals of the game
- Compared with a hardware board
  - ◆ the board = instance of the Framework
  - ◆ ICs = objects
  - ◆ backplane = ORB

H.4 Frameworks

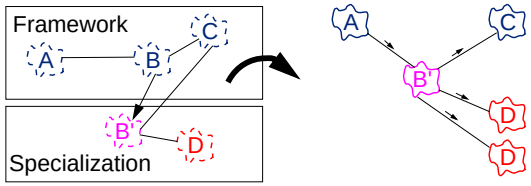
1 Classes and Class Libraries

- Class = design for a set of objects
- Class library
  - ◆ Collection of classes
  - ◆ Flow of control: application objects → library objects



2 What is a Framework? (2)

- Frameworks
  - ◆ are an application or application skeleton
  - ◆ application developer may
    – add – substitute – modify
    components
  - ◆ Flow of control:
    framework → application object → framework
    "Don't call us, we'll call you" (Hollywood principle)

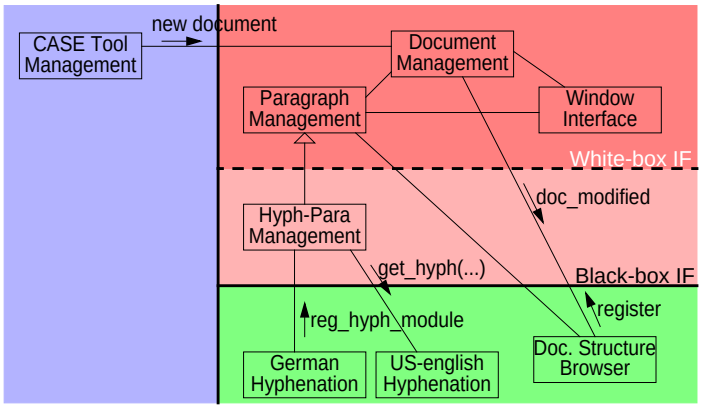


3 Frameworks — How?

- Two sorts of interfaces, two ways for customization
- Client API
  - ➤ external interface of the framework (how can other applications interact with the framework)
  - ➤ described in IDL
- Framework API (Black-box Interface for customization)
  - ➤ internal interface of the framework (how can new components interact with the rest of the framework + how does the framework interact with the new components)
  - ➤ interface described in IDL, protocol for registration & notification
- Subclasses of framework components (White-box Interface for customization)
  - ➤ customization + replacement of components of the framework
  - ➤ polymorphism guarantees interoperation

4 Example

- Framework for document processing



5 Benefits

- Prefabricated infrastructure
  - ➤ reduces coding, debugging & testing
- Architectural guidance
  - ➤ software is wired and ready to go
  - ➤ you just have to plug in your extensions
- Less monolithic applications
  - ➤ small pieces of applications are plugged into existing frameworks
  - ➤ existing frameworks are plugged together
- Foundation for a software components industry
  - ➤ Well-designed general frameworks are the basis for problem-specific solutions
- Reduced maintenance
  - ➤ Frameworks provide the bulk of (hopefully well-tested) code

H.5 Types of Frameworks

1 Application Frameworks

- Expertise applicable to a wide variety of programs
  - ➤ graphical user interfaces

2 Domain Frameworks

- Expertise in a particular problem domain
  - ➤ manufacturing control
  - ➤ document processing

3 Support Frameworks

- System-level services
  - ➤ file systems, device interaction, ...

H.6 CORBA & Frameworks

- Main goal of CORBA
  - ➥ infrastructure for business objects
- ★ Business objects
  - ➤ a representation of a thing active in the business domain
  - ➤ includes
    - – business name and definition
    - – attributes, behavior, relationship, constraints
  - ➤ examples:
    - – a person (customer), a place, a concept (invoice, contract), …
  - ➥ may be used in unpredictable combinations
  - ➥ is independent of specific applications
  - ◆ represents a "everyday life entity" → exists in the "end user's world"
  - ◆ in contrast: entities that make sense only to information systems

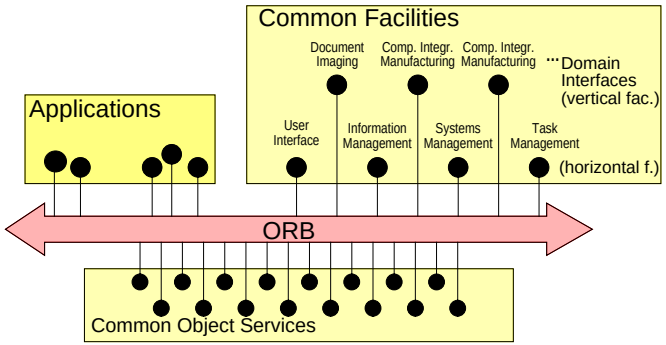
H.7 Java & Frameworks

1 IBM SanFrancisco

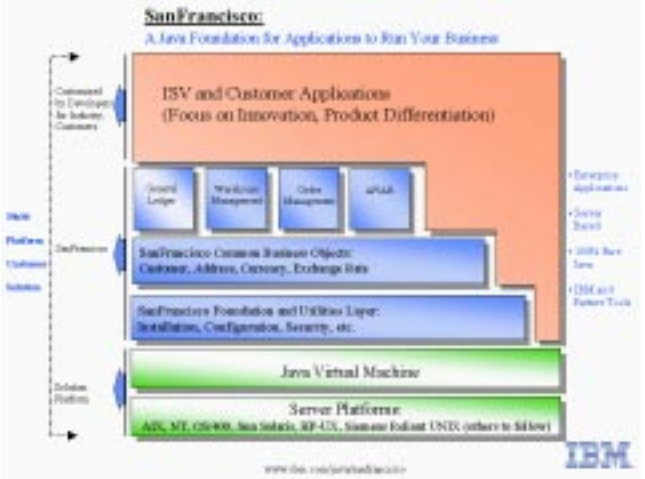
- Commercial application framework
  - ➤ automated business management systems
  - ➤ components for
    - – General Ledger
    - – Order Management
    - – Warehouse Management
- Based on Java

H.6 CORBA & Frameworks (2)

- CORBA Facilities
  - ➤ provide services for business objects
  - ➤ are built on top of Common Object Services
  - ➤ extend the "primitive" COS



1 IBM SanFrancisco (2)



2 Java Frameworks

- Java Media Framework
  - ➤ audio and video device control
- Lightweight UI Framework
  - ➤ Customizable user interface environment
- General Administrative Framework

★ Fundamental concepts for building Frameworks with Java:
- ➥ Interfaces to describe type conformance
- ➥ Component technology: Java Beans
- ➥ Enterprise JavaBeans

3 Enterprise JavaBeans

- Standard multitier component architecture
  for reusable server components
  - ➥ infrastructure for distributed frameworks
- Application server
  - ➤ execution environment for application components / business objects
  - ➤ most parts of an application's logic moved from clients to server
- Multitier applications
  - ◆ client/server = 2-tier
    - ➤ presentation logic + business logic + data manipulation logic on client
    - ➤ database on server
  - ◆ thin client + distributed server components = multitier
    - ➤ presentation logic on client
      - ➥ thin client
    - ➤ business logic + data manipulation logic = separate server components