Indefinitely Evolvable Architectures: Event-Based Systems

Ted Faison

Faison Computing Inc.

ted.faison@computer.org

EBSs have Superior Non-Functional Characteristics

- Manageability - Independent teams
- Maintainability – Independent changes
- Deployability – Independent updates
- Testability – Independent parts
- Verifiability – Independent implementations
- Flexibility – Independent designs

EBSs are cheaper and faster to build

- EBSs are cheaper

  Lots of small and independent parts are cheaper to build than fewer large and dependent parts

- EBSs are easier to build

  Development teams can work largely in parallel, due to the independence between parts. Final integration is much easier.

- EBSs can evolve indefinitely

  The parts are small and independent, so changes in the system requirements tend to have much smaller impacts on the individual parts. Changes often require only changing the system wiring and adding new parts.

Important EBS Definitions

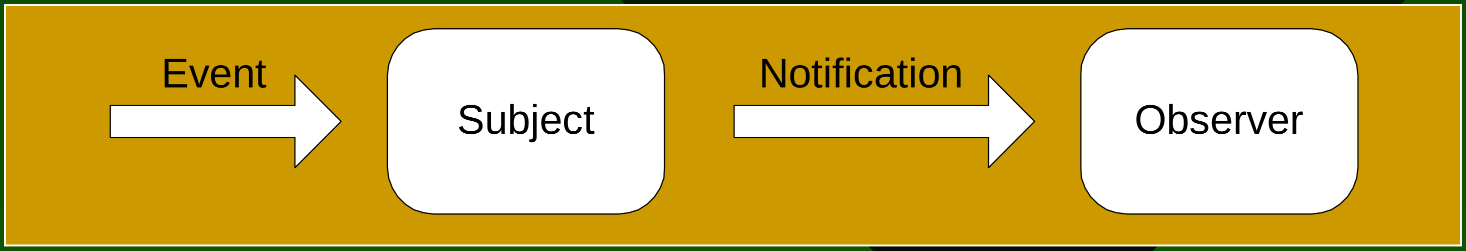
- Event

  A detectable occurrence

- Notification

  Messages triggered by events

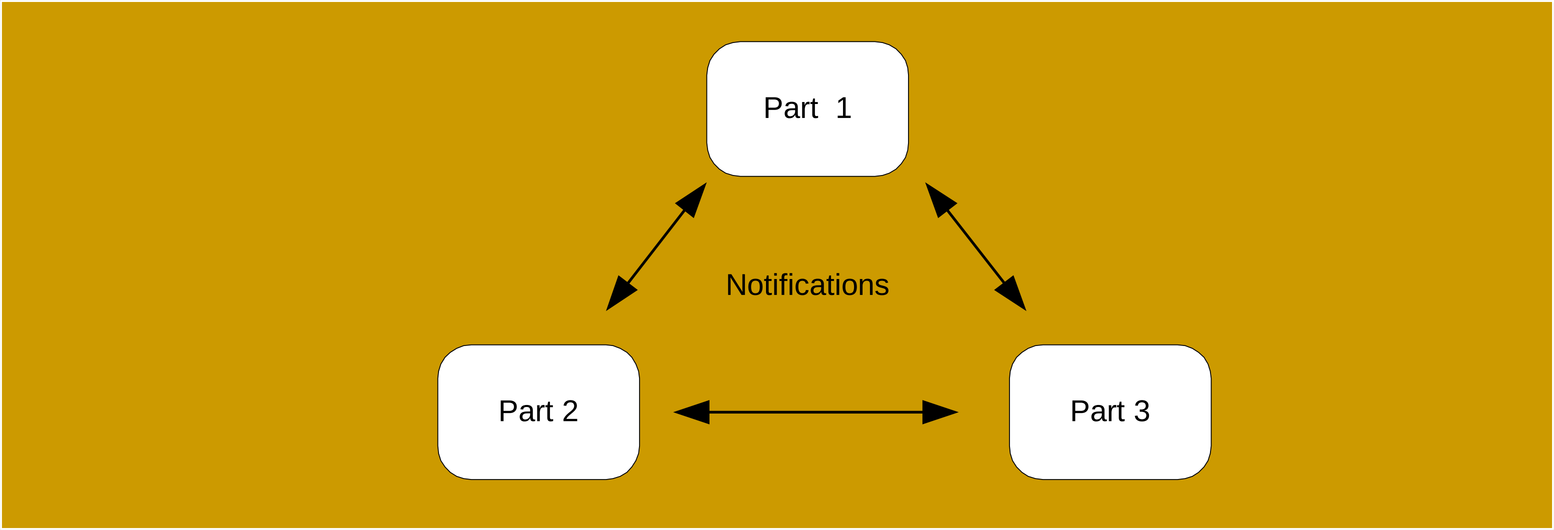
- The Observer design pattern



What is an EBS?

- It’s all about the system connectivity

The constituent parts interact primarily or solely via notifications



What is an EBS?

- It’s all about coupling (and how to avoid it)
  - Static Coupling

    Occurs at compile-time

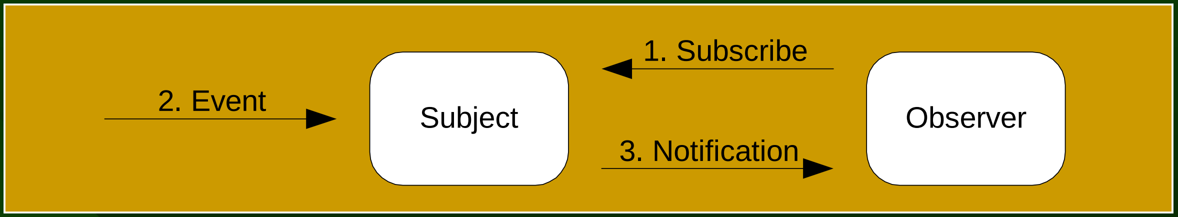
    Greatly affects development teams
  - Dynamic Coupling

    Occurs at run-time

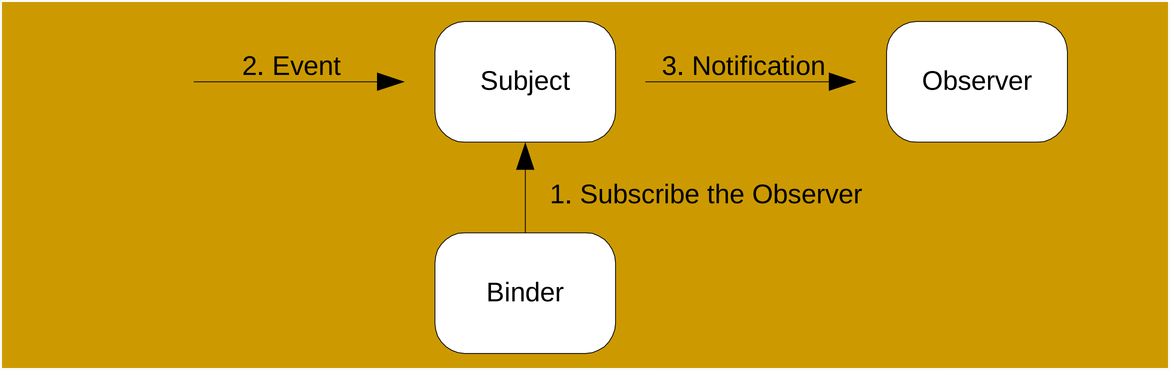
    Has little affect on development teams

Improving the Observer pattern

- Self-subscribing Observers are coupled to Subjects



- Binders decouple the parts



Firing Events

(aka sending notifications)

- Sending Messages
- Using Procedure Calls
  - Typed calls

    Introduce type coupling, which is static

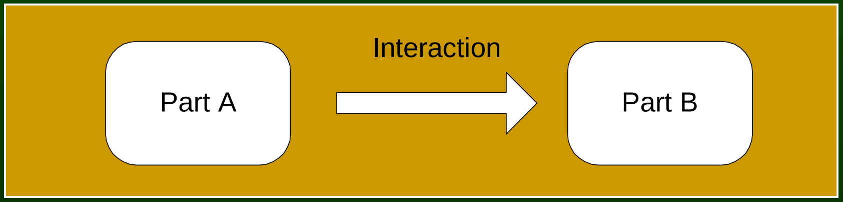
    Example: myTypedReference.DoSomething( )

  - Untyped calls

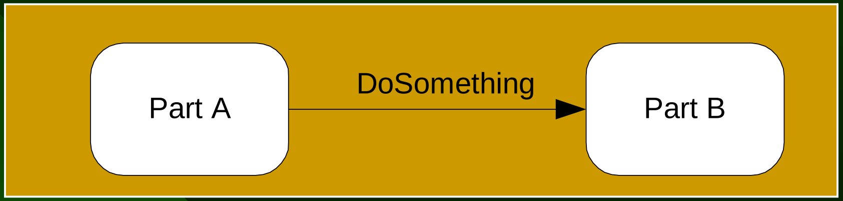
    Introduce signature coupling, which is dynamic

    Example : myMethodReference.Execute( )

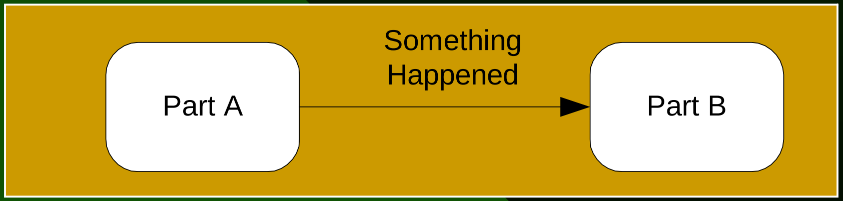
Interaction dynamics: active and reactive patterns



- Active interactions



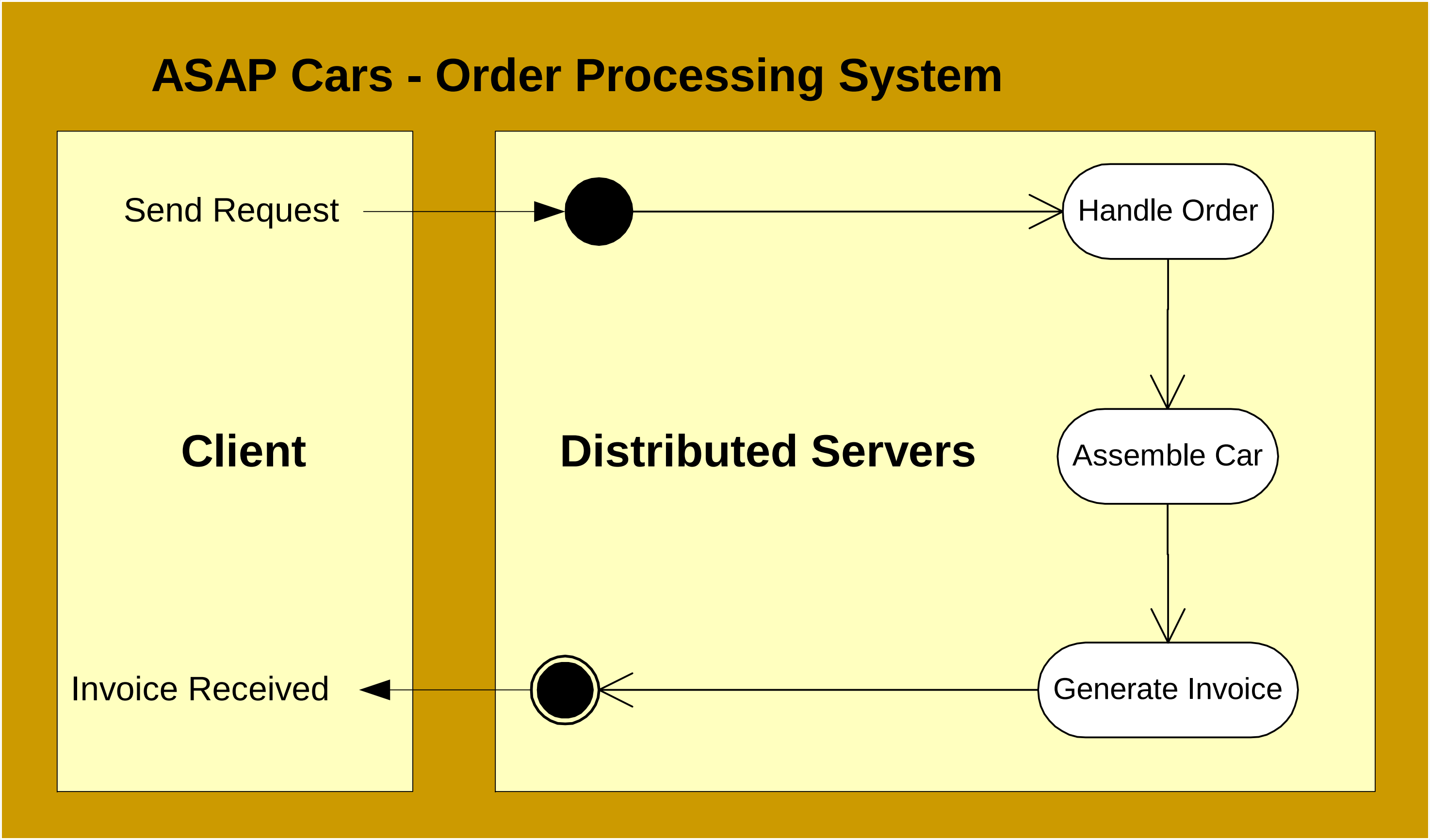
- Reactive interactions



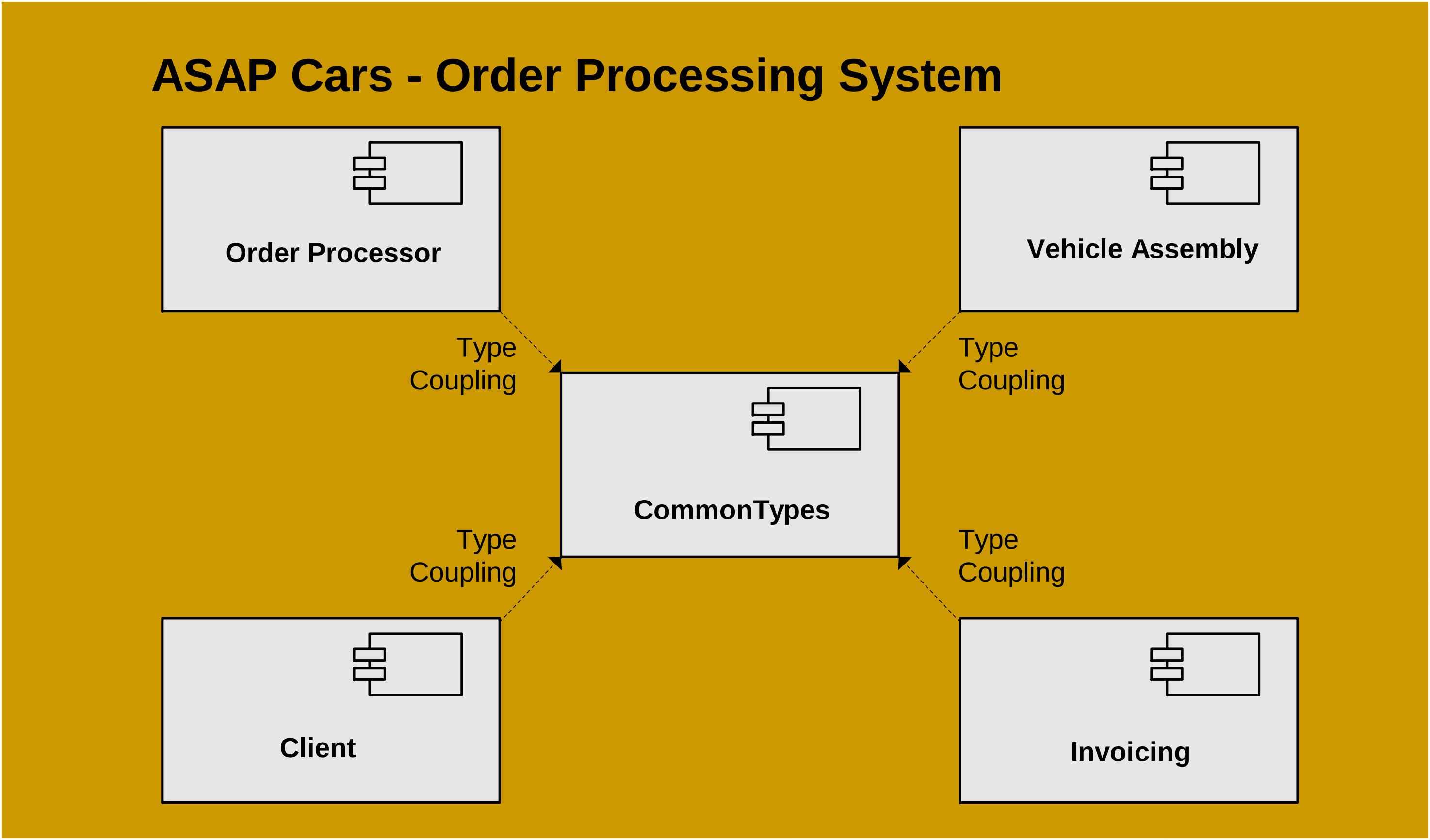
Complexity Versus Size

- Heavily coupled systems: complexity grows exponentially with size
- Decoupled systems: complexity grows linearly with size

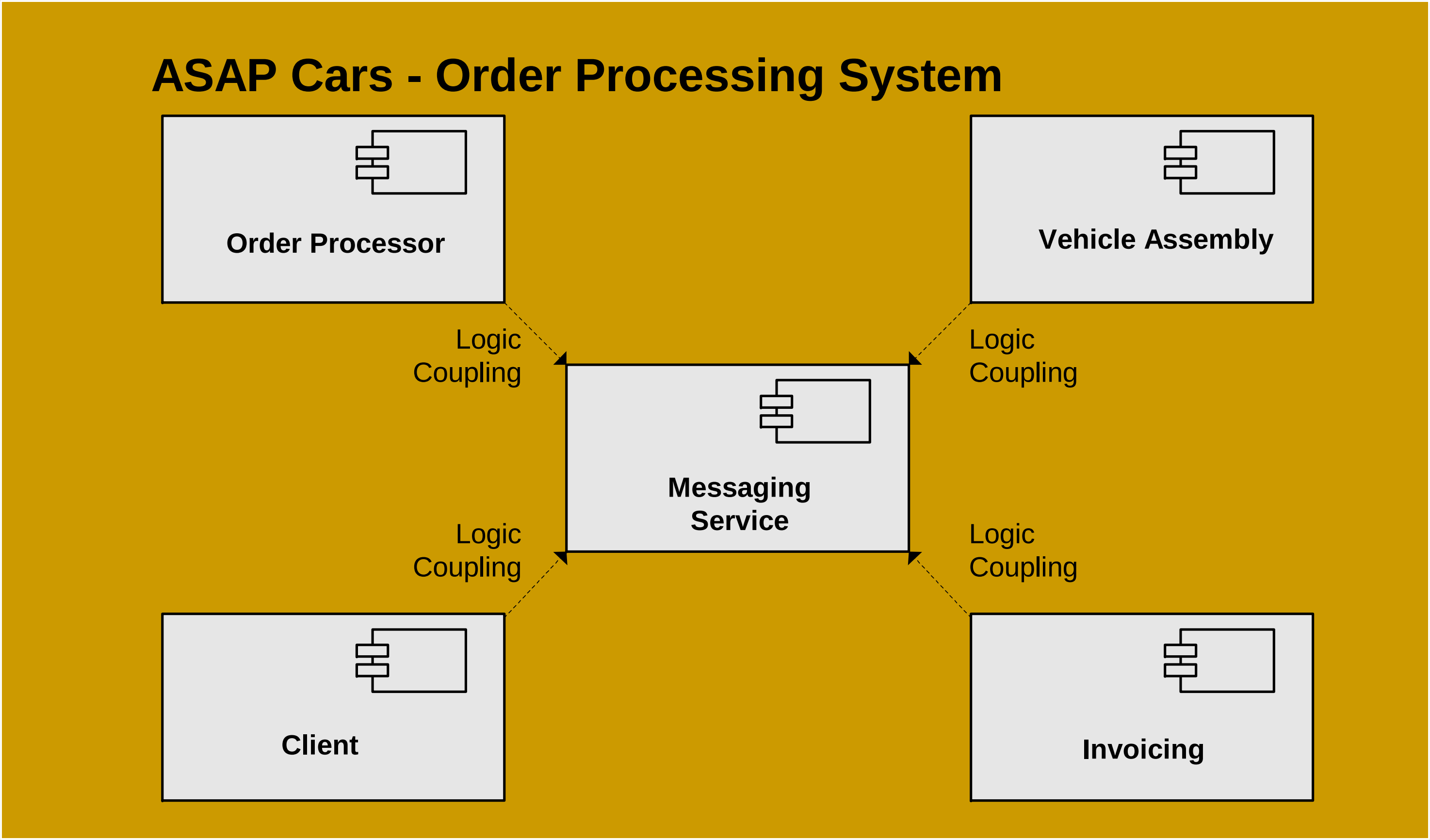
Case Study 1: A Distributed Workflow System



System Coupling Diagram

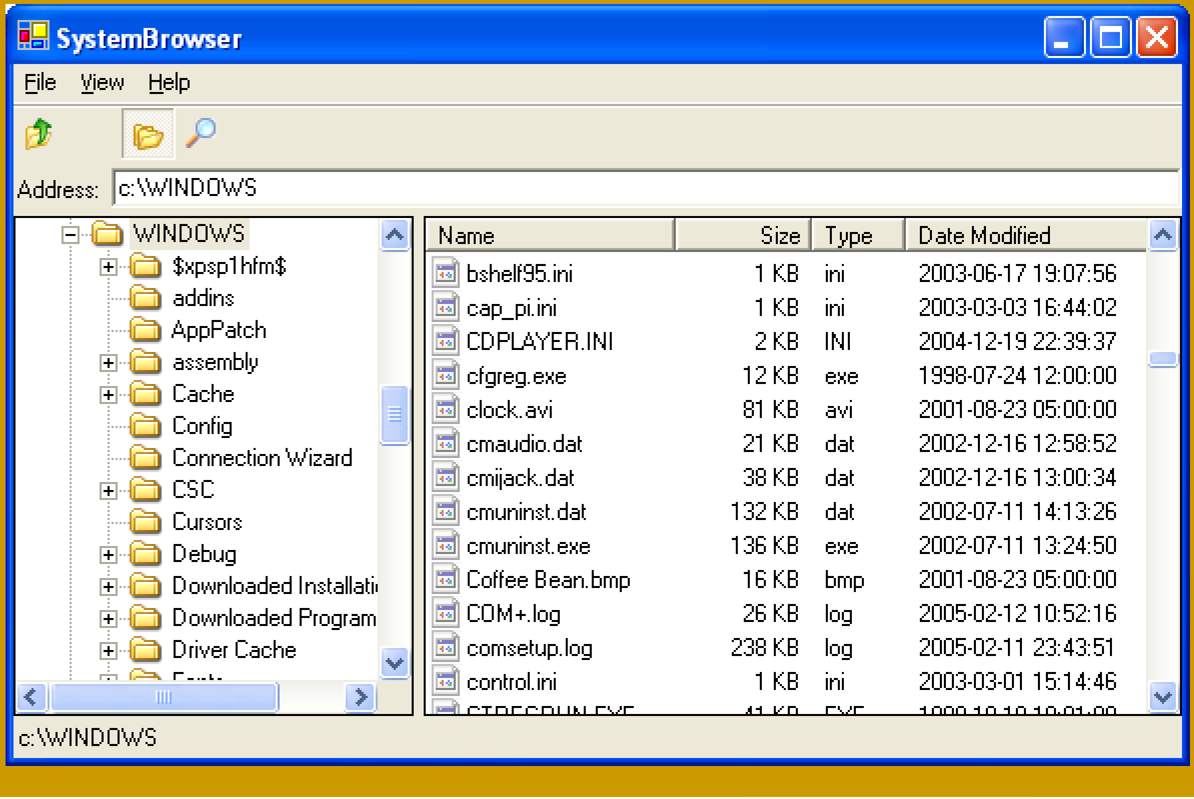


System Communication



Case Study 2: A System Browser

User Interface - File Browser



Address: c:\WINDOWS

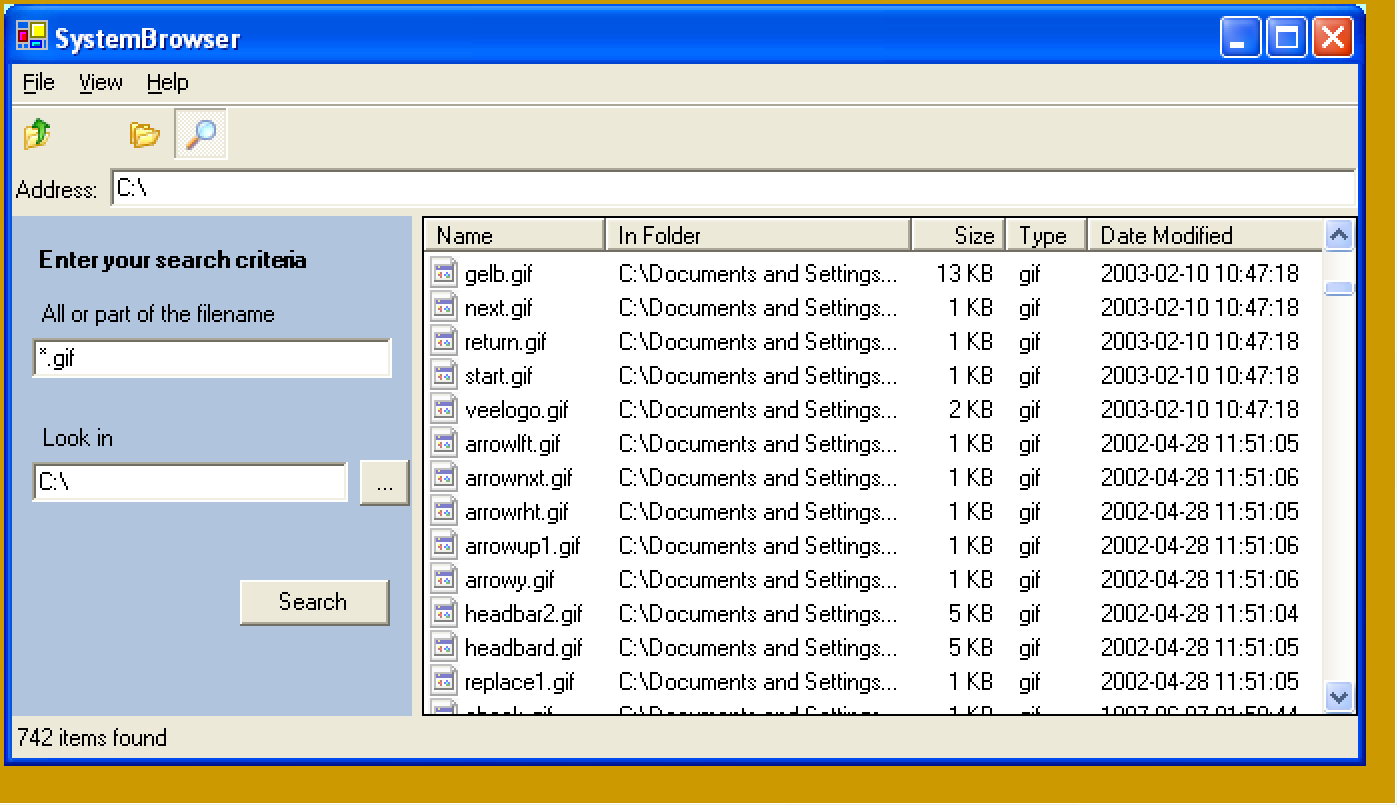
- WINDOWS
  - $xpsp1hfm$
  - addins
  - AppPatch
  - assembly
  - Cache
  - Config
  - Connection Wizard
  - CSC
  - Cursors
  - Debug
  - Downloaded Installati
  - Downloaded Program
  - Driver Cache

Name	Size	Type	Date Modified
bshelf95.ini	1 KB	ini	2003-06-17 19:07:56
cap_pi.ini	1 KB	ini	2003-03-03 16:44:02
CDPLAYER.INI	2 KB	INI	2004-12-19 22:39:37
cfgreg.exe	12 KB	exe	1998-07-24 12:00:00
clock.avi	81 KB	avi	2001-08-23 05:00:00
cmaudio.dat	21 KB	dat	2002-12-16 12:58:52
cmijack.dat	38 KB	dat	2002-12-16 13:00:34
cmuninst.dat	132 KB	dat	2002-07-11 14:13:26
cmuninst.exe	136 KB	exe	2002-07-11 13:24:50
Coffee Bean.bmp	16 KB	bmp	2001-08-23 05:00:00
COM+.log	26 KB	log	2005-02-12 10:52:16
comsetup.log	238 KB	log	2005-02-11 23:43:51
control.ini	1 KB	ini	2003-03-01 15:14:46

c:\WINDOWS

File Searcher

User Interface - File Searcher



Address: C:\

Enter your search criteria

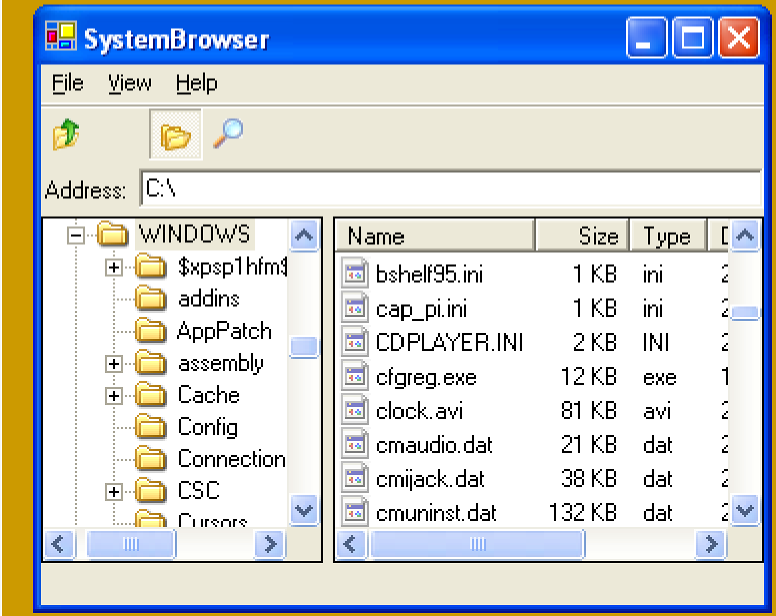
All or part of the filename: *.gif

Look in: C:\

Name	In Folder	Size	Type	Date Modified
gelb.gif	C:\Documents and Settings...	13 KB	gif	2003-02-10 10:47:18
next.gif	C:\Documents and Settings...	1 KB	gif	2003-02-10 10:47:18
return.gif	C:\Documents and Settings...	1 KB	gif	2003-02-10 10:47:18
start.gif	C:\Documents and Settings...	1 KB	gif	2003-02-10 10:47:18
veelogo.gif	C:\Documents and Settings...	2 KB	gif	2003-02-10 10:47:18
arrowlft.gif	C:\Documents and Settings...	1 KB	gif	2002-04-28 11:51:05
arrownxt.gif	C:\Documents and Settings...	1 KB	gif	2002-04-28 11:51:06
arrowrht.gif	C:\Documents and Settings...	1 KB	gif	2002-04-28 11:51:05
arrowup1.gif	C:\Documents and Settings...	1 KB	gif	2002-04-28 11:51:06
arrowy.gif	C:\Documents and Settings...	1 KB	gif	2002-04-28 11:51:06
headbar2.gif	C:\Documents and Settings...	5 KB	gif	2002-04-28 11:51:04
headbard.gif	C:\Documents and Settings...	5 KB	gif	2002-04-28 11:51:05
replace1.gif	C:\Documents and Settings...	1 KB	gif	2002-04-28 11:51:05

742 items found

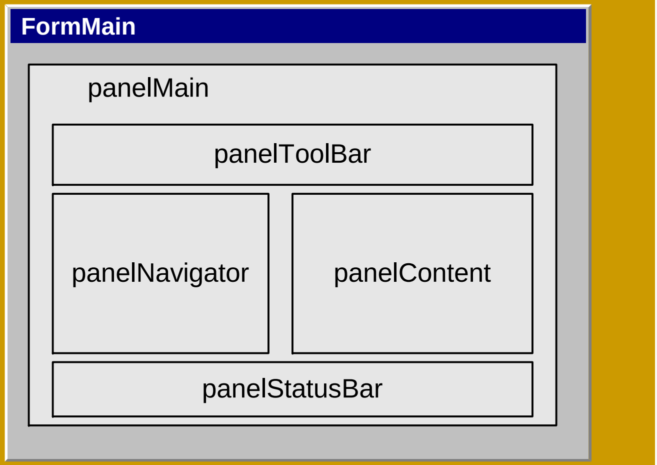
User Interface Structure



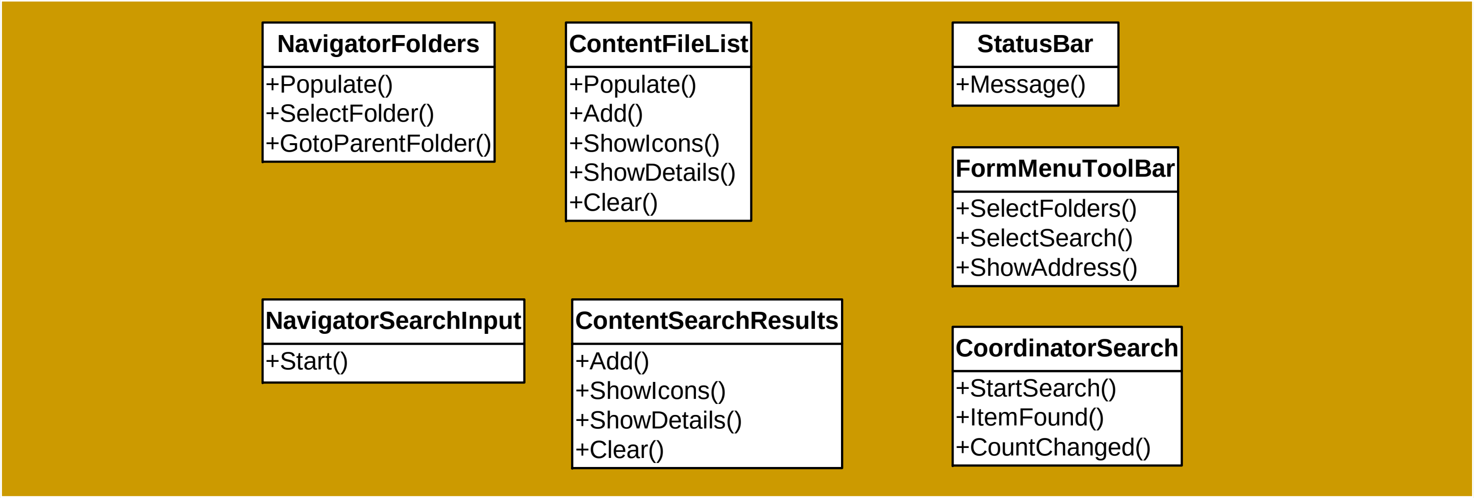
Address: C:\

- WINDOWS
  - $xpsp1hfm$
  - addins
  - AppPatch
  - assembly
  - Cache
  - Config
  - Connection
  - CSC
  - Cursors

Name	Size	Type	D
bshelf95.ini	1 KB	ini	2
cap_pi.ini	1 KB	ini	2
CDPLAYER.INI	2 KB	INI	2
cfgreg.exe	12 KB	exe	1
clock.avi	81 KB	avi	2
cmaudio.dat	21 KB	dat	2
cmijack.dat	38 KB	dat	2
cmuninst.dat	132 KB	dat	2



Class Diagram – Main Parts



NavigatorFolders
+Populate()
+SelectFolder()
+GotoParentFolder()

ContentFileList
+Populate()
+Add()
+ShowIcons()
+ShowDetails()
+Clear()

StatusBar
+Message()

FormMenuToolBar
+SelectFolders()
+SelectSearch()
+ShowAddress()

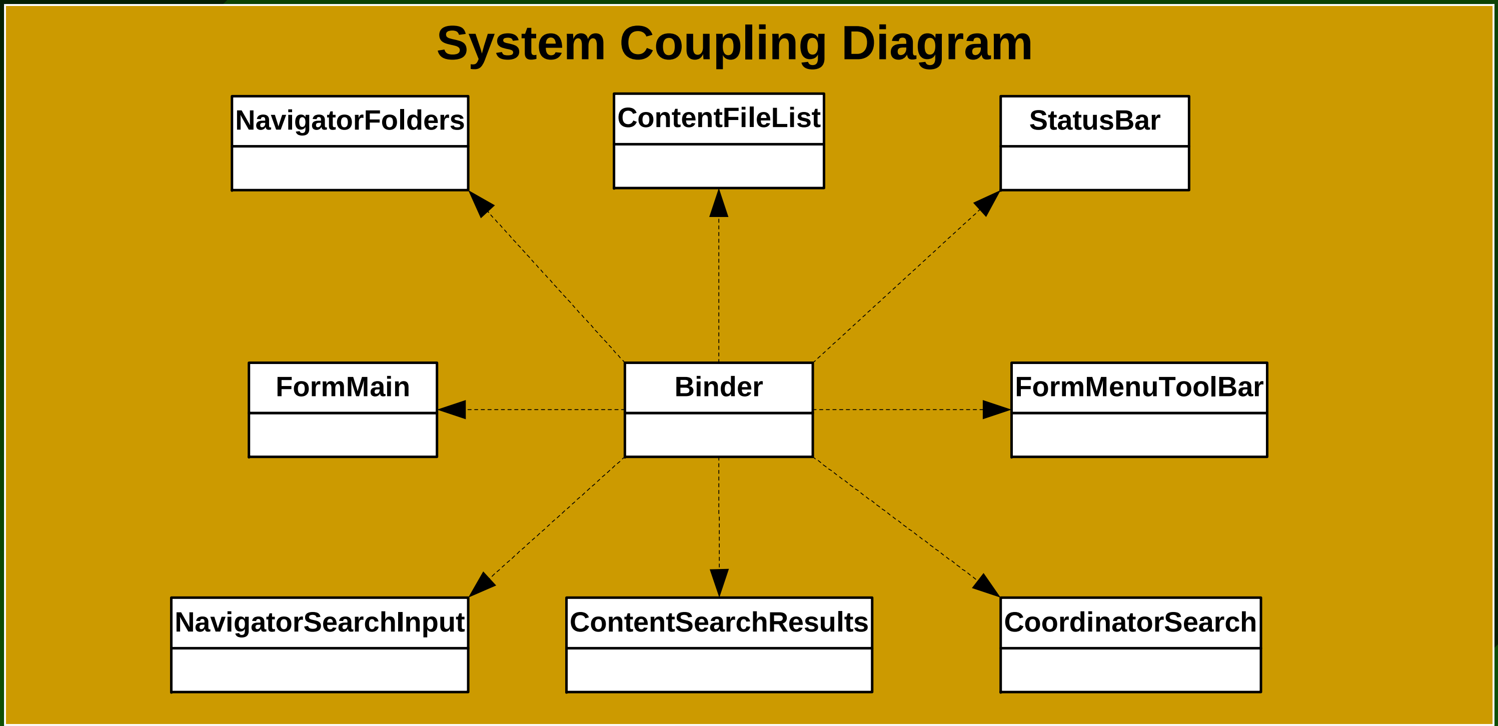
NavigatorSearchInput
+Start()

ContentSearchResults
+Add()
+ShowIcons()
+ShowDetails()
+Clear()

CoordinatorSearch
+StartSearch()
+ItemFound()
+CountChanged()

- There are no relationships between the main classes, meaning there is no static coupling between them
- Objects interact using event notifications

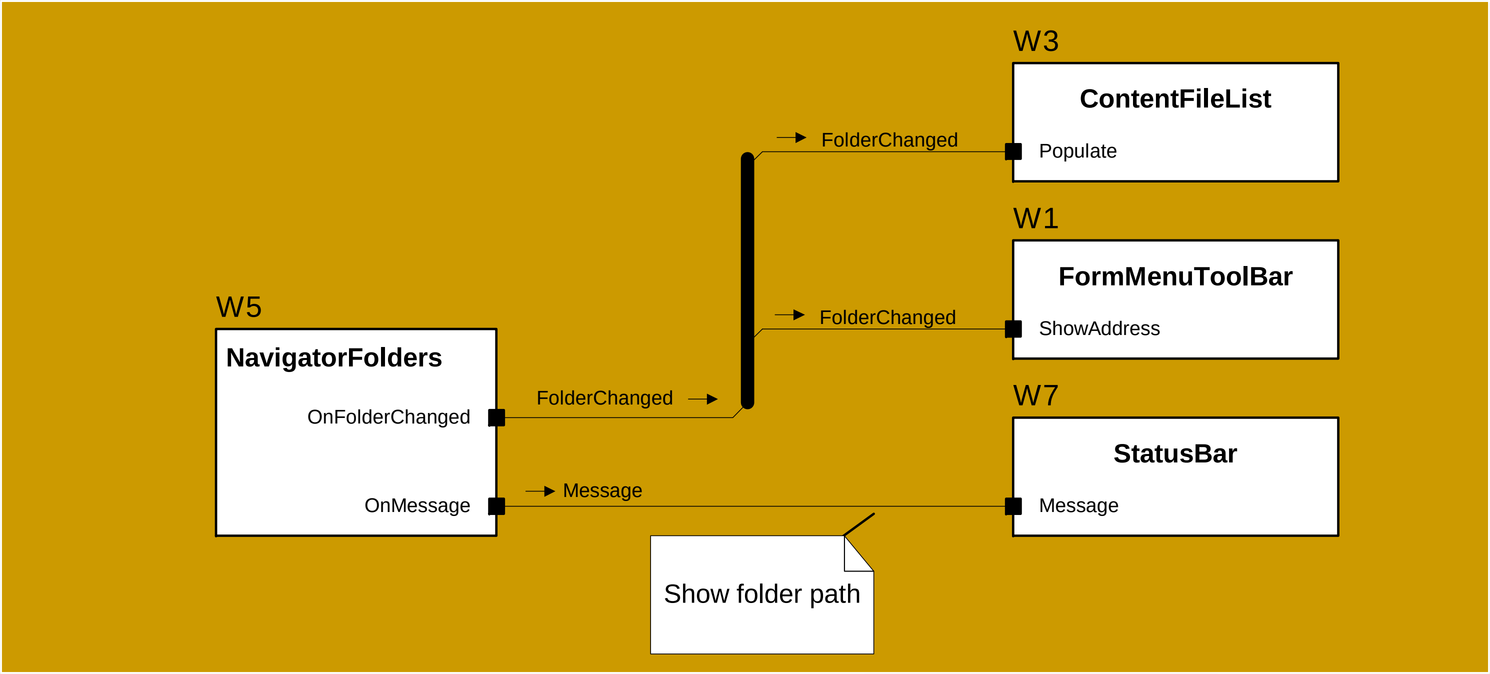
The Binder



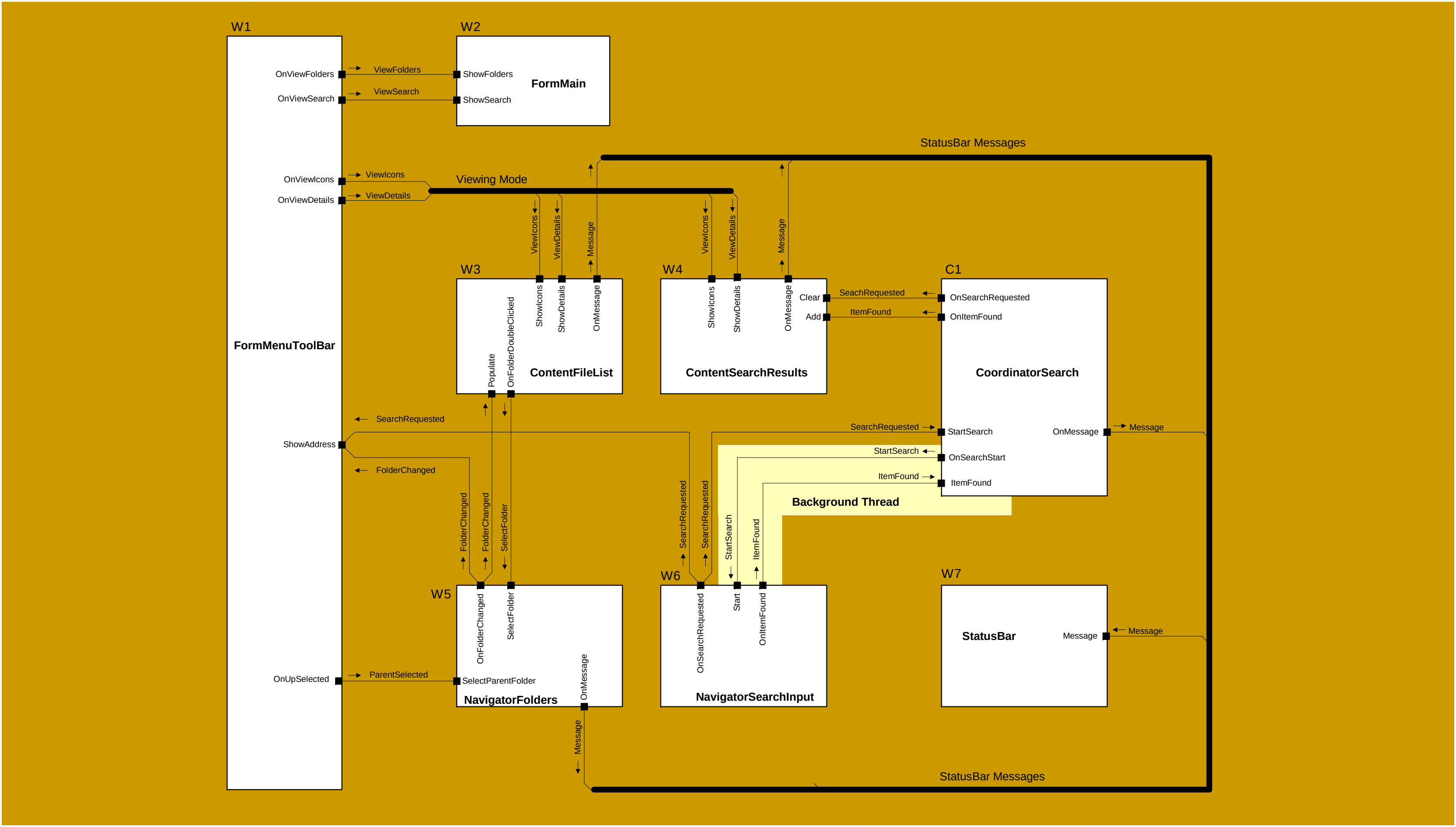
- The Binder is coupled to all the classes in the system

Signal Wiring Diagrams

- Use Case: User selects a folder in the Folders navigator



The Wiring Diagram as a blueprint of connectivity



Advantages of EBSs

- Most parts of a system are statically decoupled from the others
- Decoupled parts are easier to design, because they don’t call other parts
- Decoupled parts are easier to develop and maintain, because they can be tested in isolation from the rest of the system
- Decoupled systems are easier to extend and evolve, since the main parts are not aware of the others