Analysis of Software Artifacts

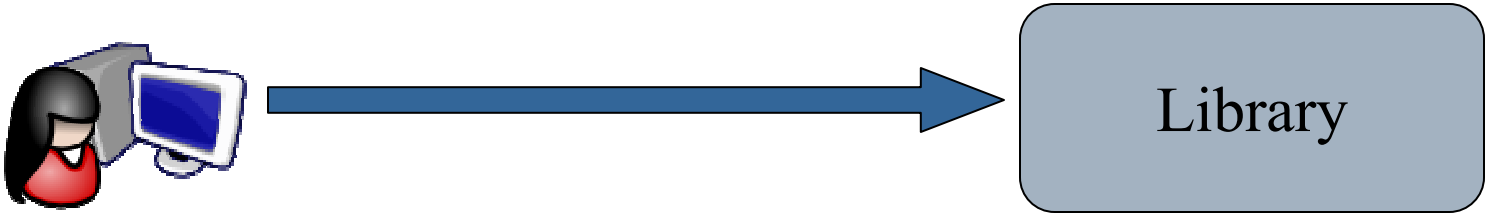
Patterns and Frameworks

Jonathan Aldrich

Some material adapted from
Ciera Jaspan and
William Scherlis

Libraries

- Libraries – familiar to programmers
  - Our code makes outside calls to do some task
- We control
  - Program control flow
  - Architecture
- Library controls
  - Data representations
- The library does a routine, and passes back data



Math

Collections

I/O

Swing

Frameworks

- Frameworks – familiar to professional developers
- Frameworks controls
  - Program flow
  - Architecture
  - Data representations
- Calls back to customized client code
  - The Hollywood principle: “Don’t call us; we’ll call you.”

Frameworks are beneficial

- Examples
  - EJB, .NET, Applet, Eclipse
- Domain knowledge
  - Evolved from VisualAge (circa 1985)
  - Embodies best practice
- Large-scale code reuse
  - Eclipse framework: ~2,000,000 LoC
  - Eclipse client: 12 LoC
  - … of course you need to know which 12 lines to write

Framework essentials

- Purpose:
  - A framework is ***the skeleton of an application***
    - Can be customized by an application developer.*
- Structure:
  - A framework is ***a reusable design*** of a (sub)system
    - Represented by a set of abstract classes and the way their instances interact.*
  - Our code is a ***plug-in*** to the framework
    - A customized piece of code developed to use a framework
  - Methods
    - Callback methods – services *required* by framework to client
      - Client gives customized functionality (overrides a framework method or abstract method)
      - Lifecycle method – gets called according to state of plug-in
    - Service methods – services *provided* by framework to client
      - Framework gives library-like functionality

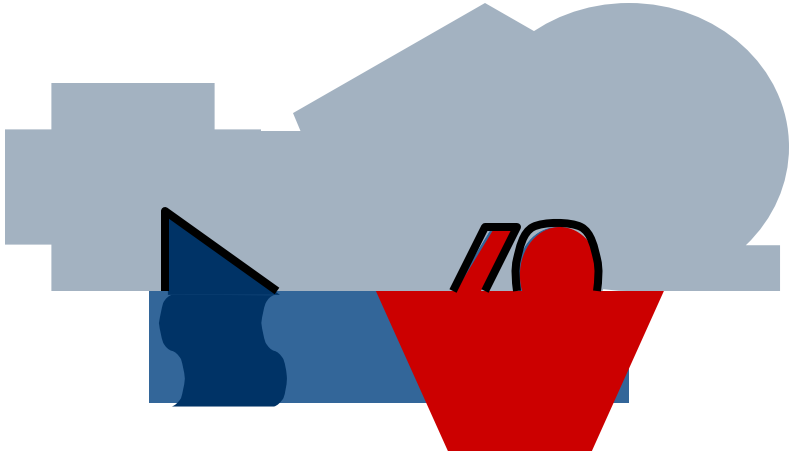
* Ralph E. Johnson. **Frameworks = (components + patterns).** CACM, 40(10): 39–42, 1997.

Partial Design Patterns

- Another way to think about frameworks
- The framework provides one part of the design pattern
- We provide the other part

Partial Design Patterns

- Another way to think about frameworks
- The framework provides one part of the design pattern
- We provide the other part



Example: An Eclipse Plugin

- A popular Java IDE
- More generally, a framework for tools that facilitate "building, deploying and managing software across the lifecycle."
- Plug-in framework based on OSGI standard
- Starting point: Manifest file
  - Plugin name
  - Activator class
  - Meta-data

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: MyEditor Plug-in
Bundle-SymbolicName: MyEditor;
      singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator: myeditor.Activator
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime,
 org.eclipse.jface.text,
 org.eclipse.ui.editors
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment:
      JavaSE-1.6
```

Example: An Eclipse Plugin

- plugin.xml
  - Main configuration file
  - XML format
  - Lists extension points
- Editor extension
  - extension point: org.eclipse.ui.editors
  - file extension
  - icon used in corner of editor
  - class name
  - unique id
    - refer to this editor
    - other plugins can extend with new menu items, etc.!

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>

  <extension
        point="org.eclipse.ui.editors">
    <editor
          name="Sample XML Editor"
          extensions="xml"
          icon="icons/sample.gif“
          contributorClass="org.eclipse.ui.texteditor.
          BasicTextEditorActionContributor"
          class="myeditor.editors.XMLEditor"
          id="myeditor.editors.XMLEditor">
    </editor>
  </extension>

</plugin>
```

Example: An Eclipse Plugin

- At last, code!
- XMLEditor.java
  - Inherits TextEditor behavior
    - open, close, save, display, select, cut/copy/paste, search/replace, …
    - REALLY NICE not to have to implement this
    - But could have used ITextEditor interface if we wanted to
  - Extends with syntax highlighting
    - XMLDocumentProvider partitions into tags and comments
    - XMLConfiguration shows how to color partitions

```
package myeditor.editors;

import org.eclipse.ui.editors.text.TextEditor;

public class XMLEditor extends TextEditor {
    private ColorManager colorManager;

    public XMLEditor() {
        super();
        colorManager = new ColorManager();
        setSourceViewerConfiguration(
                new XMLConfiguration(colorManager));
        setDocumentProvider(
                new XMLDocumentProvider());
    }

    public void dispose() {
        colorManager.dispose();
        super.dispose();
    }
}
```

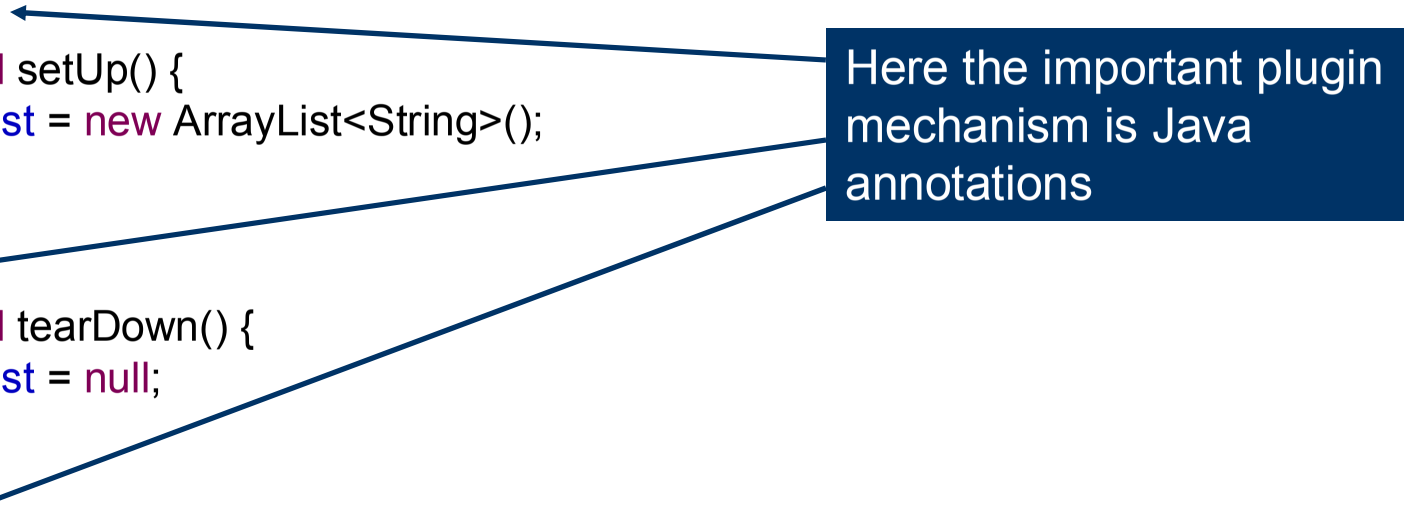
Example: a JUnit Plugin

```
public class SampleTest  {
    private List<String> emptyList;

    @Before
    public void setUp() {
        emptyList = new ArrayList<String>();
    }

    @After
    public void tearDown() {
        emptyList = null;
    }

    @Test
    public void testEmptyList() {
        assertEquals("Empty list should have 0 elements",
                0, emptyList.size());
    }
}
```



Example: Drop-down list

- Simple **ASP.NET** Page with a drop down list
  - Derive from Page
  - Add the control
  - Handle any user actions on control
- 10 simple Page callbacks
  - Many more complex ones
- Where do we add the controls?
- Where do we set up the click event?
- Working the simple case not enough
- Where does the framework expect it to happen?

- `PreInit`
- `Init`
- `InitComplete`
- `PreLoad`
- `Load`
- `Control events…`
- `LoadComplete`
- `PreRender`
- `SaveStateComplete`
- `Render`
- `Close`

Quality attributes and architecture

- Quality attributes = Non-functional requirements
  - Performance
  - Security
  - Scalability
  - Evolvability (change-support-ability)
  - *-ility
- QA's trade off with each other
- Engineering for quality attributes
  - Old way: hack quality attributes in after development
  - New way: Embed quality attributes into the architecture
    - More cost effective, less refactoring
    - Handled at high level, not scattered in program
    - Works if you know your QA tradeoffs up front

Benefits

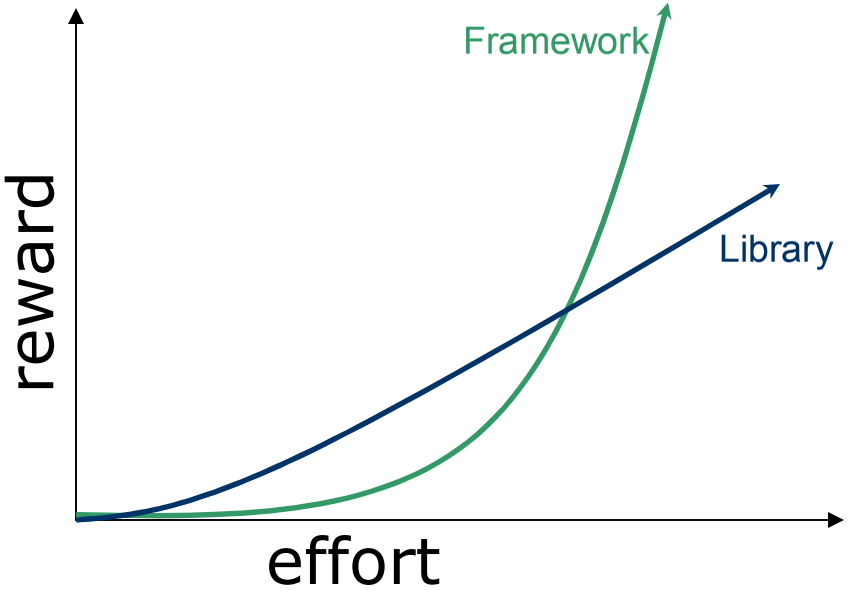
- Architecture reuse
- Abstraction of quality attributes
- Code reuse
- Maintainability
- Existing knowledge of employees
- External community support

- Engineering risk management!
  Risk = Cost of consequence * Likelihood of consequence

Risks

- Quality attributes built into framework architecture
  - Do the frameworks quality attributes match yours?
  - Mismatches are difficult to work around
    - Eclipse: can't have the same file open in 2 editors
    - JUnit: number of tests in suite is statically fixed

- Learning curve
  - Learn framework essentials
  - Learn how each task is supported
  - Great power once mastered



Getting up a framework's learning curve

- Tips on using frameworks
  - Tutorials, Wizards, and Examples
    - SourceForge, Google Code Search
  - Communities – email lists and forums
    - Eclipse.org
  - Group knowledge dispersal
    - Wiki of resources, Problem/solution log
- Common client trick: Follow the leader
  - Appropriate code from examples – find an "imputed pattern"
    - Search source code
    - Infer compatible intent
    - Identify scope (not too much, not too little)
  - Copy it
    - Tear out the app-specific logic, keep the bureaucracy
    - Insert your own logic into the reused bureaucracy
  - But there's a problem
    - Classic copy-and-paste problem – looks just like my own code
    - Design intent is lost – "my intention is to use the framework this way"
- Framework designer's conundrum: complexity vs. capability