Adaptive Software Engineering  
G22.2440-001  

Session 9 - Main Theme  
Software Engineering Tools Primer  

Dr. Jean-Claude Franchitti  

New York University  
Computer Science Department  
Courant Institute of Mathematical Sciences  

Agenda  

• Software Engineering Tools Primer  
  • Build Tools (e.g., Ant)  
  • Continuous Build Process Frameworks (e.g., CruiseControl)  
  • Unit Testing Frameworks (e.g., jUnit)  
  • Refactoring Browsers (e.g., IntelliJ’s IDEA)  
  • Selecting Appropriate Tools  
• Summary  
  • Individual Assignment #5  
  • Course Project (Part 3)  
  • Readings
Summary of Previous Session

- Enterprise Architectural Patterns
- Sample Middleware Reference Architectures
- Architectural Capabilities
- Object-Oriented Design Guidelines
- Summary
  - Individual Assignment #5
  - Project (Part 3)

Part I

Software Engineering Tools Primer
eXtreme/Agile Programming Tools

- JUnit (CppUnit, NUnit, VBUnit) - JUnit is a regression testing framework written by Erich Gamma and Kent Beck. for Java and ported to C++, C# and Visual Basic.
- JUnit Report - an Ant task that merges the individual XML files generated by the JUnit task and applies a stylesheet on the resulting merged document to provide a browsable report of the testcases results.
- JUnitPerf - a collection of JUnit test decorators used to measure the performance and scalability of functionality contained within existing JUnit tests
- Clover - Clover is a code coverage tool for Java. It discovers sections of code that are not being executed. This may be used to determine where tests are not adequately exercising the code.

(continued)

Thoughtworks:
- Continuous Integration: A fully automated build and test process that allows you to build several times a day
- Test Driven Development: TDD is a process by which the tests are written before the code itself. Through a rapid cycle of adding new tests, making them pass, and then refactoring to clean code, the software design evolves through the tests.

Thoughtworks Tools:
- CruiseControl: CruiseControl is a framework for a continuous build process. It includes, but is not limited to, plugins for email notification, Ant, and various source control tools. A web interface is provided to view the details of the current and previous builds.
eXtreme/Agile Programming Tools (continued)

- Thoughtworks Tools (continued):
  - Dashboard: Dashboard is a plugin for IntelliJ IDEA and Eclipse that gives instant feedback to the developer regarding the current status of a continuous integration build on a remote machine. It is currently used to give a stage by stage, visual representation of the state of a cruise control build.
  - Marathon: Marathon is a gui-test tool that allows you to play and record scripts against a java swing ui. It's written in java, and uses python as its scripting language - the emphasis being on an extremely simple, readable syntax that customers/testers/analysts feel comfortable with.

eXtreme/Agile Programming Tools (continued)

- Apache Software Foundation (Jakarta)
  - Ant - is a platform independent build system (somewhat analogous to make).
  - JMeter - a 100% pure Java desktop application designed to load test functional behavior and measure performance.
  - Cactus - Cactus is a simple test framework for unit testing server-side java code.

- SourceForge.net
  - Unit Tests Generator - part of "Web Test Tools" - provides tools for automatically generating unit tests. Integrate these using Ant to get automatic generation of new tests when a class or method is added.
  - Dbunit - This database testing framework is a JUnit extension which sets up your database in a known state before executing your tests.
eXtreme/Agile Programming Tools
(continued)

- SourceForge.net (continued)
  - XMLUnit - Seems to be mostly about testing for differences.
  - JUnitEE - provides a TestRunner which outputs HTML and a servlet which can be used as an entry point to your test cases.
  - JUNITPP - an extension to the JUNIT framework that allows a test data repository and load/stress test from the command line.
  - Hansel is an extension to JUnit implementing code coverage tests.
  - Grinder is a Java load-testing framework that is freely available under a BSD-style open-source license.
  - JTestCase helps you in separating test data from test code. In another word, JTestCase enables unit tests to be "data-centric".

- JFunc is an extension to the JUnit testing framework to make it easier for use with functional (integration) tests.
- TagUnit is a framework whereby JSP custom tags can be tested inside the container, and in isolation to the application specific pages on which they will ultimately be used.
- MaxQ is a free web functional testing tool. It includes an HTTP proxy that records your test script, and a command line utility that can be used to playback tests.
- StrutsTestCase is an extension of the JUnit TestCase class that allows testing of individual Action objects with or without a running servlet engine. This framework provides both mock servlet objects as well as Cactus support to simulate the environment.
Refactoring
http://www.refactoring.com/

- Technique to restructure code in a disciplined way
  - Small code changes (a.k.a., refactorings) are applied to support new requirements and/or keep design as simple as possible
- Enables programmers to safely and easily evolve their code to fulfill new requirements or improve its quality
- Refactoring is a fundamental coding practice of XP and is orthogonal to Agile Modeling, which does not address programming-related issues
- See Java refactoring guidelines at
  - http://www.cafeaulait.org/slides/javapolis/refactoring/
- Refactoring tools:
  - Eclipse supports renaming refactorings that allow you to rename a compilation unit, type, method, field, or parameter
  - Other refactorings allow you to move code, extract methods, and self-encapsulate fields

Design Patterns and Refactoring

- Refactoring improves code design without adding new behavior
- A design pattern is the description of a design problem and of its solution, which comes with certain benefits and liabilities
  - See http://cs.wwc.edu/~aabyan/PATTERNS/
- Do design patterns drive refactoring or are design patterns discovered in the refactoring result?
  - See Refactoring to Patterns
Java IDEs

- Eclipse
- NetBeans
- Sun ONE Studio
- JBuilder
- Visual Age for Java
- VisualCafè
- Codewarrior for Java
- WebGain Studio
- ModelJ
- J2ME Wireless Toolkit
- IntelliJ
- etc.

(Java) IDEs – Selection Criteria

- User Interface Ease of Use
- IDE Customization Features
- Code Editing Features
- On-the-Fly Code Analysis
- Code Style, Generation, Inspection
- Refactoring Support
- J2EE Support (JSP, XML, EJB, etc.)
- Navigation
- Search and Replace
- JavaDoc Support
- Local Version Control Support
- Version Control System (VCS) Integrations
- Compiler
- Running/Debugging Support
- Debugger
- Other Integrated Tools
- Open API
- Written in Java / Available on All Platforms
Software Tools for XP

- Cactus
  - Test framework for unit testing server-side java code (Servlets, EJBs, Tag Libs, Filters, …)
  - Cactus uses JUnit and extends it
  - JUnit is a regression testing framework written by Erich Gamma and Kent Beck. It is used by the developer who implements unit tests in Java
  - Provides a packaged and simple mechanism based on Ant to automate server-side testing

Software Tools for XP
(continued)
Part II

XML Support for MDA Technology
See: Session 3 Main Theme (Part IV) on Application Architecture and Modeling

Model Driven Architectures (MDA)

- MDA Technology Relies on:
  - UML, MOF, and CWM
- Applications Based on MDA are Platform Independent
- Implementations/Realizations can be Targeted to Any Application Server Platform
  - XML-Based Web-Enabled/Web Services-Enabled Platforms Based on CORBA, J2EE, Microsoft
**MDA**

**UML Model (PIM)**

- **Auto**
  - Color: String
  - Door: Integer
  - Engine: Integer

**XMI Document (PSM)**

```
<Auto>
  <Color> Red </Color>
  <Door> 4 </Door>
  <Engine> 2 </Engine>
</Auto>
```

**XMI DTD, Schema (PSM)**

```
<!Element Auto (Color*, Door*, Engine*)>
```

**IDL, Java… (PSM)**

```
interface Auto {
  public String color;
  public int Door;
  public int Engine;
}
```

---

**MDA**

- OMG's MDA (Model-Driven Architecture) specification describes:
  - a PIM - platform-independent models (i.e. business design)
  - PSMs - the mapping of a PIM to one or more platform-specific model

- MDA => Model Once, Generate Everywhere
- Review MDA presentations:
  - [http://www.io-software.com](http://www.io-software.com)
UML’s Business Engineering Methodology

- Business Model/Architecture
  - Use Case View/Model
- Application Model/Architecture
  - Logical and Process View/Models
    - Content, Data, and Process Model (e.g., OIM’s knowledge management, and database/datawarehousing models)
- Application Infrastructure Model/Architecture
  - Implementation View
    - Component Model (e.g., OIM’s component and object model)
- Technology Model/Architecture
  - Deployment View/Model

UML and Modeling Methodologies

- UML: object modeling
- XML: content modeling
- ORM: data modeling
Towards XML Model Based Computing
(review)

- Step 1: Document Object Model
- Step 2: XML Data Binding
- Step 3: Standard XML Information Models
- Step 4: XML Application Services Frameworks
  - Processing, Rendering, Querying, Secure Messaging
- Step 5: XML-Based “Web Object Model”
  - Web Services Architecture
- Step 6: XML Model Driven Architectures (to come)

Current XML-Based Software Development

- Business Engineering Methodology
  - Language + Process + Tools
  - e.g., Rational Unified Process (RUP)
- XML Application Development Infrastructure
  - Metadata Management (e.g., XMI)
  - XML APIs (e.g., JAXP, JAXB)
  - XML Tools (e.g., XML Editors, XML Parsers)
- XML Applications:
  - Application(s) of XML
  - XML-based applications/services
    - MOM & POP
    - Other Services
  - Application Infrastructure Frameworks
XML Metadata Management

- Issue: UML may not provide enough modeling views and enough expressive power in each view to represent a complete application
- Possible Solutions:
  - Extend UML
    - See CWM’s Analysis and Design Model
  - Use Different Modeling Languages:
    - See handout on “XML Information Modeling” (uses different models such as UML, XML, and ORM)
  - Use a Meta-Model: MOF and XMI

XML Support for UML Modeling

- Meta Object Facility (MOF)
  - CORBA Common Facility for the Management of Meta Information such as UML Models, Database Schemas, Programming Language Types, etc.
- XML Metadata Interchange (XMI)
  - Enables interchange of metadata between modeling tools
  - Develop models using Rational Rose or Java, and the XMI toolkit (now part of IBM’s EMF).
  - Use Objects by Design xmi-to-html.xsl style sheet and Cocoon framework to present the resulting model
MDA Development Approach

Sample MOF MetaModel
Sample MOF MetaModel
(continued)

MOF to XML Mapping
XML Information Modeling

- Steps
  - Documenting the Information Structure
  - Representing the Information Structure in XML Form
  - Defining XML DTDs and/or Schemas

- Modeling Techniques
  - UML: object modeling
  - XML: content modeling
  - ORM: data modeling
  - UML, MOF and XMI
Using UML to Model MLs

(continued)

Glossaries

Entry

place holder (from XML)

SignedEntry

authorName : string

creationDate : date

Glossary

topic : string

creationDate : date

0..n
+entry 0..n

0..n
+sub-glossary 0..n

0..n

DialectCode

frenchCanadian : string

dialectCode : DialectCode

<<xsd:complexType>>

DialectInfo

frenchCanadian : string

dialectCode : DialectCode

<<xsd:choice>>

Entry

english (0..1) : string

french (0..1) : string

<<xsd:complexType>>

Glossary

topic : string

creationDate : date

0..n
+entry 0..n

0..n
+sub-glossary 0..n

<<xsd:simpleType>>

PlaceHolder

(from XHTML)

0..2
+description 0..2

<<xsd:schema>>

Datatypes

string

<<xsd:simpleType>>

DialectCode

(from ISO 639-1)
Practical XML Metadata Management

- Practical Use of XMI:
  - Develop a model using Rational Rose or Java, and the XMI toolkit
  - Use Objects by Design xmi-to-html.xsl style sheet and Cocoon framework to present the resulting model
- Current State of Tools:
  - Forward-Engineering of UML into XML Schemas is possible
    - Can export .mdl files from Rational Rose, and convert them into XMI using SoftModeler (www.softera.com)
    - SoftModeler can convert UML into XML W3C Schemas
  - Reverse-Engineering of XML Schemas into UML is more difficult
    - Not supported by current tools

Part III

Architectural Considerations
Patterns of Enterprise Application Architectures
AspectJ – Locking Sample

• Creating a new aspect called lock with introduce and advise cross-cuts

01 aspect Lock {
03 Data sharedDataInstance;
04 Lock( Data d ) {
05 sharedDataInstance = d;
06 }
08 introduce Lock Data.lock;
09
10 advise Data() {
11    static after {
12        thisObject.lock = new Lock( thisObject );
13    }
14 }
15 }

AspectJ – Locking Sample (continued)

• Advising classes that work with the data (note that all the locking code is included in an aspect!)

15    boolean locked = false;
16
17 advise Worker.perform*(..), AnotherWorker.perform*(..) {
18    before {
19        if ( thisObject.sharedDataInstace.lock.locked ) // enqueue, wait
20            thisObject.sharedDataInstace.lock.locked = true;
21    }
22    after {
23        thisObject.sharedDataInstace.lock.locked = false;
24    }
25 }
26 }
Catalog of Patterns
(www.martinfowler.com/eaaCatalog)

- Domain Logic Patterns (e.g., Service Layer)
- Data Source Architectural Patterns
  - e.g., Data Mapper
- Object-Relational Behavioral Patterns
  - e.g., Unit of Work
- Object-Relational Structural Patterns
  - e.g., Inheritance Mappers
- Object-Relational Metadata Mapping Patterns
  - e.g., Repository
- Web Presentation Patterns
  - e.g., MVC, Application Controller
- Distribution Patterns (e.g., Remote Façade, DTO)
- Offline Concurrency Patterns (e.g., Implicit lock)
- Session State Patterns (e.g., Server Session State)
- Base Patterns (e.g., record set, value object)

MVC Pattern

splits user interface interaction into three different roles
Application Controller Pattern

Data Transfer Object Pattern

object that carries data between processes to reduce the number of calls
Remote Facade Pattern
provide coarse-grained object façade to optimize network usage

Open Source Frameworks

- Apache Software Foundation (Jakarta)
  - Tomcat - A servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies.
  - Struts - a popular open source framework for building Web applications that integrate with standard technologies like Java Servlets JavaBeans, and JavaServer Pages (JSP).
  - Cocoon - an XML publishing framework that raises the usage of XML and XSLT technologies for server applications to a new level. Designed for performance and scalability around pipelined SAX processing, Cocoon offers a flexible environment based on a separation of concerns between content, logic, and style. Transform data into Web applications with Cocoon.
Open Source Frameworks
(continued)

- SourceForge.net
  - HttpUnit is a free, open source Java API for accessing web sites without a browser.
  - JBoss - A J2EE application server for Enterprise Java Beans (EJB).

Part V

Conclusion
Course Assignments

• Individual Assignments
  • Problems and reports based on case studies or exercises

• Project-Related Assignments
  • All assignments (other than the individual assessments) will correspond to milestones in the team project.
  • As the course progresses, students will be applying various methodologies to a project of their choice. The project and related software system should relate to a real-world scenario chosen by each team. The project will consists inter-related deliverables which are due on a (bi-) weekly basis.
  • There will be only one submission per team per deliverable and all teams must demonstrate their projects to the course instructor.
  • A sample project description and additional details will be available under handouts on the course Web site.

Course Project

• Project Logistics
  • Teams will pick their own projects, within certain constraints: for instance, all projects should involve multiple distributed subsystems (e.g., web-based electronic services projects including client, application server, and database tiers). Students will need to come up to speed on whatever programming languages and/or software technologies they choose for their projects - which will not necessarily be covered in class.
  • Students will be required to form themselves into "pairs" of exactly two (2) members each; if there is an odd number of students in the class, then one (1) team of three (3) members will be permitted. There may not be any "pairs" of only one member! The instructor and TA(s) will then assist the pairs in forming "teams", ideally each consisting of two (2) "pairs", possibly three (3) pairs if necessary due to enrollment, but students are encouraged to form their own 2-pair teams in advance. If some students drop the course, any remaining pair or team members may be arbitrarily reassigned to other pairs/teams at the discretion of the instructor (but are strongly encouraged to reform pairs/teams on their own). Students will develop and test their project code together with the other member of their programming pair.
Sample Project Methodology
Very eXtreme Programming (VXP)

• After teams formed, 1/2 week to Project Concept
• 1/2 week to Revised Project Concept
• 2 to 3 iterations
• For each iteration:
  – 1/2 week to plan
  – 1 week to iteration report and demo

Sample Project Methodology
Very eXtreme Programming (VXP)
(continued)

• Requirements: Your project focuses on two application services
• Planning: User stories and work breakdown
• Doing: Pair programming, write test cases before coding, automate testing
• Demoing: 5 minute presentation plus 15 minute demo
• Reporting: What got done, what didn’t, what tests show
• 1st iteration: Any
• 2nd iteration: Use some component model framework
• 3rd iteration: Refactoring, do it right this time
Revised Project Concept (Tips)

1. Cover page (max 1 page)
2. Basic concept (max 3 pages): Briefly describe the system your team proposes to build. Write this description in the form of either user stories or use cases (your choice). Illustrations do not count towards page limits.
3. Controversies (max 1 page)

First Iteration Plan (Tips)

• Requirements (max 2 pages):
• Select user stories or use cases to implement in your first iteration, to produce a demo by the last week of class
• Assign priorities and points to each unit - A point should correspond to the amount of work you expect one pair to be able to accomplish within one week
• You may optionally include additional medium priority points to do “if you have time”
• It is acceptable to include fewer, more or different use cases or user stories than actually appeared in your Revised Project Concept
First Iteration Plan (Tips)

• Work Breakdown (max 3 pages):
• Refine as engineering tasks and assign to pairs
• Describe specifically what will need to be coded in order to complete each task
• Also describe what unit and integration tests will be implemented and performed
• You may need additional engineering tasks that do not match one-to-one with your user stories/use cases
• Map out a schedule for the next weeks
• Be realistic – demo has to been shown before the end of the semester

2nd Iteration Plan (Tips): Requirements

• Max 3 pages
• Redesign/reengineer your system to use a component framework (e.g., COM+, EJB, CCM, .NET or Web Services)
• Select the user stories to include in the new system
  – Could be identical to those completed for your 1st Iteration
  – Could be brand new (but explain how they fit)
• Aim to maintain project velocity from 1st iteration
• Consider what will require new coding vs. major rework vs. minor rework vs. can be reused “as is”
2nd Iteration Plan (Tips): Breakdown

- Max 4 pages
- Define engineering tasks, again try to maintain project velocity
- Describe new unit and integration testing
- Describe regression testing
  - Can you reuse tests from 1st iteration?
  - If not, how will you know you didn’t break something that previously worked?
- 2nd iteration report and demo to be presented before the end of the semester

2nd Iteration Report (Tips): Requirements

- Max 2 pages
- For each engineering task from your 2nd Iteration Plan, indicate whether it succeeded, partially succeeded (and to what extent), failed (and how so?), or was not attempted
- Estimate how many user story points were actually completed (these might be fractional)
- Discuss specifically your success, or lack thereof, in porting to or reengineering for your chosen component model framework(s)
2\textsuperscript{nd} Iteration Report (Tips): Testing

- Max 3 pages
- Describe the general strategy you followed for unit testing, integration testing and regression testing
- Were you able to reuse unit and/or integration tests, with little or no change, from your 1\textsuperscript{st} Iteration as regression tests?
- What was most difficult to test?
- Did using a component model framework help or hinder your testing?

Project Presentation and Demo

- All Iterations Due
- Presentation slides (optional)
Readings

• Readings
  • Slides and Handouts posted on the course web site
  • Slides and Handouts posted on the course web site
  • Documentation provided with software engineering tools

• Project Frameworks Setup (ongoing)
  • As per references provided on the course Web site
  • See Session 6 Handouts: “Information Modeling with XML”, “Using UML to Model Applications of XML”

• Individual Assignment
  • See Session 8 Handout: “Assignment #5”

• Team Assignment
  • See Session 9 Handout: “Team Project” (Part 3)

Next Session

Business Model Engineering – Advanced Topics

• TBA