Adaptive Software Engineering
G22.3033-007

Session 9 - Main Theme
Business Model Engineering (BME)

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- Review
- Business Process Modeling
- Capturing the Organization and Location Aspects
- Developing a Process Model
- Business Process Management
- Model Driven Architectures
- Workflow Support Technologies
- Business Model Engineering Frameworks
- BME Advanced Topics
- Summary
  - (Individual) Assignment #6
  - Project (Part 3) ongoing
Summary of Previous Session

- Towards Agile Enterprise Architecture Models
- Building an Object Model Using UML
- Architectural Analysis
- Design Patterns
- Architectural Patterns
- Architecture Design Methodology
  - Achieving Optimum-Quality Results
  - Selecting Kits and Frameworks
  - Using Open Source vs. Commercial Infrastructures
- Summary
  - Individual Assignment #5
  - Project (Part 3)

Part I

Review
Extreme Programming and Agile Modeling

http://www.agilemodeling.com/essays/agileModelingXP.htm

Overall XP Process (review)
Agile Model-Driven Development Lifecycle
(review)

Initial Requirements Modeling (days) → Initial Architectural Modeling (days)
Cycle 0: Initial Modeling

Detailed Modeling (minutes) → Implementation (Ideally Test Driven) (hours)
Cycle 1: Development
Cycle 2: Development
Cycle n: Development

Reviews (optional) All Cycles (hours)

eXtreme Programming (XP) Map
http://www.extremeprogramming.org

Extreme Programming Project

User Stories

Architectural Spike

Release Planning

Iteration

Acceptance Tests

Small Releases

Copyright 2006 J. Darwen Wells
**XP Project Lifecycle**

**XP Iteration Planning**

![Diagram of XP Project Lifecycle](image1)

![Diagram of XP Iteration Planning](image2)
Agile Modeling & XP
http://www.agilemodeling.com/, http://www.agilemodeling.com/resources.htm

- Practices-based software process whose scope is to describe how to model and document in an effective and “agile” manner
- One goal is to address the issue of how to apply modeling techniques on software projects taking an agile approach such as:
  - eXtreme Programming (XP)
  - Dynamic Systems Development Method (DSDM)
  - SCRUM
  - etc.
- Using modeling throughout the XP lifecycle

“Agile” Methodologies

- See Agile Project Development Methodology at Work:
Structured Applications Design Tips

- Reuse: should focus on Domain Models/System Family Architectures
- Applications should separate the various information elements (i.e., content, logic, style, and architecture/handling schemes)
- Various content formats: presentation, message, storage, etc.
- Application architecture supports:
  - Web Enabling (WE), XML Enabling (XE), Data Enabling (DE), Enterprise System Assurance Enabling (ESAE)
- Various application support services to support:
  - Interactions with users via content (content + logic) - WE
  - Encoding of user requests as secure (portable) messages (content generation) - XE/ESAE
  - Processing of user requests via logic (content + logic) - XE
  - Rendering of content via logic using style (content + style + logic) - WE/XE
  - Querying information via logic (content + logic) - XE/DE
  - Interactions with back office via content (content + logic) - XE/ESAE
Generic Architecture Blueprint
+ Architecture Design Methodology + Mgmt

XML Applications (logic)
Applications of XML (structured content/object data, style information)
XML Application Services (logic)
XML Application Infrastructure (logic)
Technology Infrastructure (hardware platforms)

Logical View
Implementation View
Use Case View
Process View
Deployment View

Management

Applications Of XML
Structure
Content
Style

XML Applications
Processing
Rendering
Querying
Additional Services

Application and Technology Infrastructure

Sample Conceptual Architecture Diagram
(e.g., virtual classroom environment)

Presentation Enabling
(posting, querying, locating, viewing)
SMIL

Q&A Enabling
(questions capture, integration, viewing)
XLF

XML Authoring Tool

Web Community Avatar-Based Chat Platform

Web-Based Infrastructure
(lightweight client machines, server platforms)

Legend

XML Application (logic)
Applications of XML (content/style)
XML Application Services (logic)
3rd Party Tool
XML Application Infrastructure (logic)
Technology Infrastructure (OS and hardware)
Investigating Logging Infrastructure
(e.g., virtual classroom environment)

Refined Application Architecture Blueprint
(e.g., virtual classroom environment)
Part II

Business Process Modeling (BPM)
Section Outline

- Business Process Modeling background
- Business Process Modeling Phases

Discussion

- Has anyone built something that missed the mark?
  - It failed to meet the business need
  - End-users did not use it
- How do you know that what you are doing is the right thing to do?
- Who do you build your interfaces for?
- How can you ensure that you are “doing the right thing”?
What is BPM?

• A standardized approach to analyze, streamline and integrate business processes independent of organizational boundaries, to provide maximum operating efficiencies via clearly defined business and IS projects

• A business process focused methodology
  – Not Business Process Reengineering
  – Led by the business
    – Facilitated by IS

Why BPM?

• Identify and document business process improvement opportunities
  – To gain operating efficiencies through analysis of current and future business processes

• Align business and IS
  – To promote integration of business processes and technologies by identifying strategic initiatives

• Identify and reuse processes
  – To identify core business processes
  – To share and reuse business processes across organizational boundaries
BPM Objectives

- **Analyze** business processes independent from organization
- **Quantify** the benefits and savings for each proposed initiative
- Identify opportunities for process elimination and/or reassignment
- Identify opportunities for business process improvement through automation and optimization

Risks without BPM

- Systems could be implemented as a “forced fit”
- Potential loss of business and technology alignment
- Increased manual processes and “workarounds”
- Building unnecessary user interfaces
- Not identifying the best possible solutions
The Value of BPM

- Business area and IS develop a common understanding of business requirements
  - Simple modeling notation
  - Consistent, repeatable approach
- **Short**, focused BPM efforts
  - Swift identification of process improvement initiatives
- Promotes seamless integration and reuse of processes, data and technologies
- Captures current business processes that can be used as a knowledge-base for training

---

BPM Methodology Phases
Project Planning

- Develop project plan
  - High-level scope definition
- Build project team
- Conduct project kick-off meeting
  - Core team
  - Stakeholders
- Initiate project management activities
- Training of Core team and subject matter experts

Business Needs Analysis

- Define BPM detailed scope
  - Define project purpose, goals and objectives
  - Identify problems to be addressed
  - Determine key measures
Current Process Understanding

♦ Develop business process models
♦ Identify business process information needs
♦ Capture current process metrics and characteristics
♦ Identify areas of opportunity for improvement

New Process Design

• Develop Global Business Function Model
  – Based on annotated CPU models from Optimization
  – Incorporate process improvement opportunities
Transformation

- Document recommended business initiatives
- Produce list of recommended projects
- Identify sequence for recommended IS projects

Part III

Capturing the Organization and Location Aspects
Sample Business Process Map (Trading)

Understanding Business Model Engineering

- Projects creation and update
- Projects approval
- KPI creation
- Project information retrieval
- System administration

- Data conversion
- Supplied volume data

- Hardware
  - Standard system software
  - Reporting software
  - Ad-hoc spreadsheet functions
  - Security and performance

- Office hosting development

- Location
- Application
- Business Process
- Organization
- Data

- Site navigation design
  - Site content design
  - Reusable components
  - Security workflow
  - Help

- Implementation Team
- Training Team
Part IV

Developing a Process Model

UML and Business Process Modeling

See Also:
http://xml.coverpages.org/ni2003-08-29-a.html
What is a Business Process?

A business process defines how an organization achieves its purpose.

The process represents the flow of work and information throughout the business.

This involves:
- People
- Machines
- Materials
- Products
- Roles
- Documents

Models of Business Processes

UML became the standard modeling language for business processes. Of the 9 UML diagram types, 3 are particularly important:

use cases,
activity diagrams and scenarios (sequence or collaboration diagrams).

Use cases show informally how the business process interacts with outside actors (customers, stakeholders, suppliers etc.)

The process dynamics is modeled by activity diagrams.

Often only specific scenarios are of interest. E.g. service monitoring, system reconfiguration, performance degradation etc.
A Use Case

Activity Diagrams

Activity diagrams constitute the core of a business model.

Activity Diagram: state transition diagram with concurrent states (Statechart)

Activity: state with internal action

BPM: activity = Mealy state
A UML Business Process Model: Different Views

Class Diagram

Activity Diagram

Statechart

Use Case
Quantitative Evaluation of BPMs

UML is a semi-formal language: there is little formal semantics attached to the individual diagrams.

In order to evaluate a visual UML-model we have, therefore, to enrich it with additional information and then transform it to a model with precise semantics.

This mapping supplies additional semantics to the source language.

UML- Model  ⇒  Analytical Model

Activity Diagram Applications

• Intended for applications that need control flow or object/data flow models …
• ... rather than event-driven models like state machines.
• For example: business process modeling and workflow.
• The difference in the three models is how a step in a process is initiated, especially with respect to how the step gets its inputs.
Control Flow

- Each step is taken when the previous one finishes
- …regardless of whether inputs are available, accurate, or complete.
- Emphasis is on order in which steps are taken.

Object/Data Flow

- Each step is taken when all the required input objects/data are available …
- … and only when all the inputs are available.
- Emphasis is on objects flowing between steps.
State Machine

- Each step is taken when events are detected by the machine …
- … using inputs given by the event.
- Emphasis is on reacting to environment.

Kinds of Steps in Activity Diagrams

- Action (State)
- Subactivity (State)

Just like their state machine counterparts (simple state and submachine state) except that …
- … transitions coming out of them are taken when the step is finished, rather than being triggered by a external event, …
- … and they support dynamic concurrency.
**Action (State)**

- An action is used for anything that does not directly start another activity graph, like invoking an operation on an object, or running a user-specified action.

- However, an action can invoke an operation that has another activity graph as a method (possible polymorphism).

**Subactivity (State)**

- A subactivity (state) starts another activity graph without using an operation.

- Used for functional decomposition, non-polymorphic applications, like many workflow systems.

- The invoked activity graph can be used by many subactivity states.
THE model (application) is completely OO when all action states invoke operations, and all activity graphs are methods for operations.
Dynamic concurrency

- Invokes an action or subactivity any number of times in parallel, as determined by an expression evaluated at runtime.
- Upper right-hand corner shows a multiplicity restricting the number of parallel invocations.
- Outgoing transition triggered when all invocations are done.
- Currently no standard notation for concurrency expression or how arguments are accessed by actions. Attach a note as workaround for expression.

Object Flow (State)

- A special sort of step (state) that represents the availability of a particular kind of object, perhaps in a particular state.
- No action or subactivity is invoked and control passes immediately to the next step (state).
- Places constraints on input and output parameters of steps before and after it.
Object Flow (State)

- Take Order must have an output parameter giving an order, or one of its subtypes.
- Fill Order must have an input parameter taking an order, or one of its supertypes.
- Dashed lines used with object flow have the same semantics as any other state transition.

Coordinating Steps

- Inherited from state machines
- Initial state
- Final state
- Fork and join
Coordinating Steps

- *Decision point* and *merge* ( ◇ ) are inherited from state machines.
- For modeling conventional flow chart decisions.

![Diagram 1](image1.png)

Coordinating Steps

- *Synch state* ( ○ ) is inherited from state machines but used mostly in activity graphs.
- Provides communication capability between parallel processes.

![Diagram 2](image2.png)
Convenience Features (Synch State)

- Forks and joins do not require composite states.
- Synch states may be omitted for the common case (unlimited bound and one incoming and outgoing transition).

Activity diagram notation

Convenience Features (Synch State)

- Object flow states can be synch states
Convenience Features

- Fork transitions can have guards.

- Instead of doing this:

Convenience Features

- *Partitions* are a grouping mechanism.
- *Swimlanes* are the notation for partitions.
- They do not provide domain-specific semantics.
- Tools can generate swimlane presentation from domain-specific information without partitions.
Convenience Features

- Signal send icon
  
- … translates to a transition with a send action.

- Signal receipt icon
  
- … translates to a wait state (an state with no action and a signal trigger event).

Case Study

<table>
<thead>
<tr>
<th>Submission Team</th>
<th>Task Force</th>
<th>Revision Task Force</th>
</tr>
</thead>
<tbody>
<tr>
<td>action state</td>
<td>control flow</td>
<td>initial state</td>
</tr>
</tbody>
</table>

Begin

Develop technology specification

Submit specification draft

Issue RFP

Fork of control

join of control

RFP [issued]

Specifications (initial proposal)

Collaborate with competitive submitters

Evaluate initial submissions

Finalize specification

Specifications (final proposal)

Adapted from Kobryn, "UML 2001" Communications of the ACM October 1999
When to Use Activity Diagrams

- Use activity diagrams when the behavior you are modeling ...
  - does not depend much on external events.
  - mostly has steps that run to completion, rather than being interrupted by events.
  - requires object/data flow between steps.
  - is being constructed at a stage when you are more concerned with which activities happen, rather than which objects are responsible for them (except partitions possibly).
Activity Diagram Modeling Tips

- Control flow and object flow are not separate. Both are modeled with state transitions.

- Dashed object flow lines are also control flow.

- You can mix state machine and control/object flow constructs on the same diagram (though you probably do not want to).
Activity Diagram Modeling Tips

- Activity diagrams inherit from state machines the requirement for well-structured nesting of composite states.

- This means you should either model as if composite states were there by matching all forks/decisions with a correspond join/merges …

- … or check that the diagram can be translated to one that is well-nested.

- This insures that diagram is executable under state machine semantics.
Activity Diagram Modeling Tips

Well-nested:

![Well-nested Activity Diagram]

Not well-nested:

![Not well-nested Activity Diagram]

Apply structured coding principles. (Be careful with goto’s!)
Activity Diagram Modeling Tips

Can be translated to well-nested diagram on earlier slide:

![Activity Diagram](image)

Business Process Modeling Notation (BPMN) and BPML / BPEL4WS

See also:

(BPMN standards)
http://www.bpmi.org
http://www.processwave.net/Articles/SoftwareProcess/BusinessModelingArticles.htm
http://cde.berkeley.edu/resources/bpmn/

(BPMN tools)
http://www.popkin.com/
http://www.popkin.com/customers/customer_service_center/demos/demos_with_overview_process_map.htm
Procedural Knowledge Road Map

<table>
<thead>
<tr>
<th>Type of knowledge/Levels</th>
<th>Conceptual model</th>
<th>Deployment methodology</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Instance level</td>
<td>Process instance concept</td>
<td>BPMN+</td>
<td></td>
</tr>
<tr>
<td>Process definition level</td>
<td>Process definition concept BPMC meta-model+</td>
<td>XDL+</td>
<td></td>
</tr>
</tbody>
</table>

- BPMN - Business Process Modeling Notation, BPMI, 11.02
- XPDL - XML Process Definition Language, WfMC, 07.02
- Wf-XML - Workflow Interoperability Binding, WfMC, 11.01

ICONS’ conceptual models

- Process definition concepts - WfMC’s workflow meta-model extended with
  - time modeling
  - flexible Workflow Participant Assignment [Momotko2002]
- Process instance concepts
  - advanced process instance and activity instance behavioral models (timing and criticality behavior)
  - advanced time management
BPMN and XPDL extensions

• BPMN extension (BPMN+)
  – idea - standard well-known notation for both process definition adjusted to the needs for process instance
  – ICONS’ approach - a BPMN extension to visualise process execution

• XPDL extension (XPDL+)
  – performer relationships (WPA)
  – pre & post conditions

Process definition - an example
Possible process execution

Part V

Business Process Management

See: http://www.ebpml.org/technoforum_2003_b_eng.ppt
Part VI

Model Driven Architectures

Also See http://www.omg.org/mda

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
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 OIM’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
    - See handouts on “UML, MOF, and XMI” and “OMG’s XML Metadata Interchange Format (XMI)”

Open Information Model

- Analysis and Design Model
  - Unified Modeling Language (UML) - uml.dtd
  - UML Extensions - umlx.dtd
  - Common Data Types - dtm.dtd
  - Generic Elements - gen.dtd
- Components and Object Model
  - Component Description Model - cde.dtd
- Database and Warehousing Model
  - Database Schema Elements - dbm.dtd
  - Data Transformation Elements - tfm.dtd
  - OLAP Schema Elements - olp.dtd
  - Record Oriented Legacy Databases - rec.dtd
- Knowledge Management Model
  - Semantic Definition Elements - sim.dtd
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
  - Use Objects by Design xmi-to-html.xsl style sheet and Cocoon framework to present the resulting model

MDA Development Approach
Sample MOF MetaModel

(continued)
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

MDA (continued)

UML Model (PIM)

XMI Document (PSM)

XMI DTD, Schema (PSM)

IDL, Java… (PSM)
Vision

Business Component Marketplace

- The business component marketplace is projected to be a 10b market in 5 years
- Consider the value of XML components that wrap popular legacy
- New application functionality built from components
- Components for integration and transformation
- XML and web services makes an excellent basis for such components
- Technology components, such as for repositories and DBMS
OMG Model Driven Architecture

- High level – platform independent models
- Technology Models
- Mapping
  - Custom
  - Standard
- Standard Models produce technology specific standards artifacts

Automated Model Driven Architecture

Profile (E.G. EDOC)

Infrastructure Mapping (E.G. XML)

Tools Produce & Integrate

Mapping is tuned to the infrastructure

Enterprise Components

Framework & Infrastructure (E.G. XML)
High level tooling & infrastructure

• MUST BE SIMPLE!
  – We must be able to create better applications faster
  – We must separate the technology and business concerns, enable the user
• Tooling + Infrastructure
  – Executable models are source code
  – Tooling must be technology aware
  – Infrastructure must support tooling, not manual techniques
• Model based component architectures
High level tooling & infrastructure

Model Driven Development Tool
(http://www.sygel.com/)
Part VII

Workflow Support Technologies

See http://www.wfmc.org/

Workflow Reference Model
Part VIII

Business Model Engineering Frameworks

Solution: Re-engineering the “Vision”
Business Model Engineering Context

- **Focus on Architectural IDEs**
- **No Support for Business Model Reengineering**
- **Limited Applications to IT Outsourcing**

The first level of automation. ~30 Years

<table>
<thead>
<tr>
<th>Environment</th>
<th>PIM</th>
<th>Generator, Projection</th>
<th>PSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming IDEs (e.g. JBuilder, Visual Age, NetBeans)</td>
<td>Programming Language</td>
<td>Compiler Engine</td>
<td>Diverse HW/OS Platforms</td>
</tr>
</tbody>
</table>

- **Environment**
  - Higher level of expression
  - Easier to understand
  - Portable
  - Standardized

- **PIM**
  - Generator,
  - Projection

- **PSM**
  - Diverse HW/OS Platforms

- **Environment**
  - Dependable
  - Flexible
  - Configurable
  - Optimizing
  - Complete: Linker, Debugger, Etc.
MDA= New automation levels ~ Last 8 Years

**Architectural IDEs**

- **Environment**
  - **PIM** (Model (UML, BOM...) & Modeling Style (J2EE, eEPC, COBOL, .NET...))
  - **Generator Engine**
  - **PSM** (P-Stack: A Level of Automation)

**Generator, Projection**

- **Models to Code**
- **Models to Models**

- **Model (UML, BOM...) & Modeling Style (J2EE, eEPC, COBOL, .NET...)**

**Core Modules**

- **Open MDA/UML/XML Repository**
- **MDA Cartridges**
- **Pattern Refinement Assistant**
- **UML Refinement Assistant**
- **MDA-Engine with Meta IDE**
- **Build, Deploy & Test Support**

**Optional Integrated Tools**

- **IDS ARIS**
- **Rational Rose**
- **Programming IDE**

**Std. MDA Projections**

- **J2EE/EJB, .NET**
- **BEA WebLogic**
- **IBM WAS NT, z/OS**
- **Borland, JBoss**
- **Oracle, IONA**

**Architect Edition adds support for custom infrastructure**

Towards a Holistic IT-Architecture Platform
Third Party Business Modeling IDEs

The Meta-Tools Solution

- Focus on Architectural Layers
- Meta-Tools for Model Driven Architectures
Initial Focus on Architectural Layers

- Business Architecture
- Application Suite
- Application Infrastructure
- Technology Infrastructure

Meta-Tools For Model Driven Architectures

As information is collected, work effort, estimates and solution becomes concrete
Business Model and Model Driven Architecture Engineering Toolkit

Case 1: No Business Model
No existing application

Case 2: Business Model
No existing application

Case 3: Business Model
Existing application

Reusable Models
Database
Analyzer
Generator
Application
Application Infrastructure
Technical Infrastructure Spec

Business Model Engineering Portfolio and Collateral

• Process Tools
  – Development Model
  – Delivery Process
    • Process Map Engineering
    • Information Gathering Instruments
    • Cost / Time Metrics

• Modeling Markup Language
• Reference Models
• Evaluation Metrics
Modeling Markup Language Sample

Part IX

Conclusion
Course Assignments

- Individual Assignments
  - 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.
Readings

- Readings
  - Slides and Handouts posted on the course web site
  - Documentation provided with software engineering tools
  - XP Explained (all sections)
  - Agile Software Development Ecosystems (parts I and II)
- Project Frameworks Setup (ongoing)
  - As per references provided on the course Web site
- Individual Assignment
  - See Handouts: “Assignment #6”
  - See Handouts: “Project (Part 3)” (ongoing)

Next Session:
Building Software

- Language and Platform Issues
- Component Infrastructures
- Pair Programming
- Refactoring
- Test Driven Development (TDD)
- Distributed Development and Agile Methods
  - Scalability
- Summary
  - (Individual) Assignment #6 (ongoing)
  - Project (Part 4)