Software Project Management and UML

Ali Bigdelou

Computer Aided Medical Procedures (CAMP),
Technische Universität München, Germany
Outline

• Intro to Software Project Management
• Project Requirements Specification
• Intro to UML
• UML Use Case Diagram
• UML Class Diagram
• UML Sequence Diagram
• UML Activity Diagram
• UML Resources
What is software project management?

Software project management is the art and science of planning and leading software projects.

It is a sub-discipline of project management in which software projects are planned, monitored and controlled.
Laws of Project Management

• Projects progress quickly until they are 90% complete. Then they remain at 90% complete forever.

• When things are going well, something will go wrong. When things just can’t get worse, they will. When things appear to be going better, you have overlooked something.

• If project content is allowed to change freely, the rate of change will exceed the rate of progress.

• Project teams detest progress reporting because it manifests their lack of progress.
Project Management Skills

- Leadership
- Communications
- Problem Solving
- Negotiating
- Influencing the Organization
- Mentoring
- Process and technical expertise

Four Project Dimensions

- People
- Process
- Product
- Technology
Project Phases

- All projects are divided into phases
- All phases together are known as the Project Life Cycle
- Each phase is marked by completion of Deliverables
- Identify the primary software project phases
Deliverables by Phase

Possible Deliverables by Phase
- Concept Document
- Statement of Work (SOW)
- Project Charter
- RFP & Proposal
- Requirements Document (Software Requirements Specification)
- Work Breakdown Structure (WBS)
- Functional Specification (Top Level Design Specification)
- Entity Relationship Diagram
- Data Flow Diagram
- Detailed Design Specification
- Object Diagrams
- Detailed Data Model
- Coding Standards
- Working Code
- Unit Tests
- Acceptance Test Procedures
- Tested Application
- Integration Plan
- Detailed SQA Test Plan
- SQA Test Cases
- User Documentation
- Training Plan
- Maintenance Specification
- Deployed Application

Software Concept

Requirements

Analysis

Design

Coding and Debugging

Systems Testing

Deployment & Maintenance
Requirement Specifications

• Capture key project objectives
• In collaboration with Clinical Partner and Technical Advisor
• The RS is a communications tool among the development team, the customer, and the advisors
• It should be a short document (around 5 pages).
• A presentation should be followed after confirming the document with customers (clinical partners) and technical advisors.
Content of the Requirements Specification

• Problem/opportunity
• Goal
• Objectives
• Users and Key features of the system
  – Documented as well using the Use-Case diagram
• Key System Components
  – Hardware Requirements
  – Third Party Library Dependencies
  – Component diagrams (to show how the different component are combined in the system)
• Usage Scenarios
  – Documented as well in Activity Diagram
• Basic Design of the System
  – Documented in Class diagram
UML
What is UML?

• UML stands for “Unified Modeling Language”
• UML is a notational system which is principally graphical and aims at modeling system using object oriented concepts.
• UML is termed as a “Visual Modeling Language’.
• Generally UML is used for modeling software systems.
• It is an industry-standard graphical language for specifying, visualizing, constructing, and documenting the artifacts of an object-oriented system under development.
• The UML uses mostly graphical notations to express the OO analysis and design of software projects.
• Simplifies the complex process of software design
Quick Tour

• The UML is a graphical language for
  – specifying
  – visualizing
  – constructing
  – documenting
  the artifacts of software systems
• Added to the list of OMG adopted technologies in November 1997 as UML 1.1
• Most recent minor revision is UML 1.3, adopted in November 1999
• Next minor revision will be UML 1.4, planned to be adopted in Q2 2001
• Next major revision will be UML 2.0, planned to be completed in 2002
Why UML for Modeling?

- A model is a simplified representation of a complex reality.
- Complex systems and software cannot be understood without properly modeling them.
- Provide structure for problem solving
- Today, software are getting complex and consequently we need to understand them through modeling.
- A graphical notation to communicate more clearly than natural language (imprecise) and code (too detailed).
- Help acquire an overall view of a system.
- UML is not dependent on any one language or technology.
- UML moves us from fragmentation to standardization.

In simple words, we need simpler representations for complex models and modeling is a mean for dealing with complexity.
Key UML Components

• **UML consists of**
  – **Views**: shows different faces of the system and links with the process
  – **Diagrams**: are basically the graphs that explain the contents of view.
  – **Model Elements**: are contained within the diagrams.

• **Main UML Diagrams**
  – **Use case diagram**
  – **Class diagram**
  – **State diagram**
  – **Object diagram**
  – **Sequence diagram**
  – **Collaboration diagram**
  – **Component diagram**
  – **Deployment diagram**
  – **Activity diagram**
Usecase Diagram
Use Case Diagram

- Used for describing a set of user **scenarios**
- Mainly used for capturing user requirements
- Work like a **contract** between end user and software developers
Model Elements in Use-Case Diagrams

- **ACTOR**

- **USE-CASE**

- **INTERACTION:**
  denotes set of messages exchanged among objects

- **NOTES/COMMENTS**
Use-Case Diagram (University Management System)

Student

Submit Assignments

Instructor

Mark Assignments

Upload Results

View Results

Package: University Management System

SYSTEM BOUNDARY
Class Diagram
Class diagram

- Used for describing structure and behavior in the use cases
- Provide a conceptual model of the system in terms of entities and their relationships
- Used for requirement capture, end-user interaction
- Detailed class diagrams are used for developers
- Elements of class diagram
  - Classes
  - Composition
  - Generalization
  - Dependency
  - Association
Classes

```
Window

+default-size: Rectangle
#maximum-size: Rectangle

+display ()
+hide ()
+create ()

-xptr: XWindow*
-attachXWindow(xwin:Xwindow*)

{abstract,
author=Joe,
status=tested}

+size: Area = (100,100)
#visibility: Boolean = true

Window

size: Area
visibility: Boolean

display ()
hide ()
```
Composition

Window

scrollbar [2]: Slider
title: Header
body: Panel

Window

Slider

Header

Panel
Composition

```plaintext
Window
  scrollbar:Slider 2
  title:Header 1
  body:Panel 1
```
Generalization

Separate Target Style

Shared Target Style
Dependencies

ClassA

ClassB

ClassC

ClassD

ClassE

«friend»

«call»

«refine»

ClassC combines two logical classes

operationZ()
Derived Attributes and Associations

```plaintext
{age = currentDate - birthdate}
```

```plaintext
{ Person.employer=Person.department.employer }
```
Class Diagram Example

- **OrderBean** (abstract)
  - +getOrderStatus
  - +setOrderStatus
  - +getLineItems
  - +setLineItems
  - +getCreditApproved
  - +setCreditApproved
  - ...

- **CreditCard** (abstract)
  - PMCreditCard

- **EntityBean**

- **PMOrder**

- **PMLineItem**

- **PMCreditCard**

- **Customer**
  - buyer
  - * order

- **OrderBean**
  - 1 order
  - * item

- **LineItem** (abstract)
  - * item
  - 1 commodity

- **Product**
Sequence Diagram
Sequence diagram

- Sequence diagrams demonstrate the behavior of objects in a use case by describing the objects and the messages they pass.
- The horizontal dimension shows the objects participating in the interaction.
- The vertical arrangement of messages indicates their order.
Activity Diagram
Activity Diagram

• Intended for applications that need control flow or object/data flow models …
• For example: business process modeling and workflow.
• Kinds of Steps in Activity Diagrams
  - Action (State)
  - Subactivity (State)
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.
Example

POEmployee.sortMail

Deliver Mail

POEmployee

sortMail()

Deliver Mail

Check Out Truck

Put Mail In Boxes
• Application is completely OO when all action states invoke operations
• All activity graphs are methods for operations.
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

- Initial state
- Final state
- Fork and join
Coordinating Steps

- Decision point and merge. 
- For modeling conventional flow chart decisions.

```
Calculate Cost

[cost < $50]

Get Authorization

[cost >= $50]

Charge Account
```
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).
### Convenience Features

- Partitions are a grouping mechanism.
- They do not provide domain-specific semantics.

<table>
<thead>
<tr>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluate Impact</td>
</tr>
<tr>
<td>[ priority = 1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Bug</td>
</tr>
<tr>
<td>Fix Bug</td>
</tr>
<tr>
<td>Test Fix</td>
</tr>
<tr>
<td>Release Fix</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fix Bug</td>
</tr>
<tr>
<td>Test Fix</td>
</tr>
</tbody>
</table>

---

[Diagram showing the process flow from management to support to engineering, with key steps such as register bug, fix bug, test fix, and release fix.]
Activity Diagram Modeling Tips

From UML User Guide:

Customer
- Request Return
  - Ship Item
    - Item [returned]
  - Get Return Number

Telesales

Accounting
- Credit Account
  - Item [available]

Warehouse
- Receive Item
  - Restock Item
UML Modeling Tools

- Rational Rose ([www.rational.com](http://www.rational.com)) by IBM
- ArgoUML (free software) ([http://argouml.tigris.org/](http://argouml.tigris.org/))
- OpenSource; written in java
Reference

1. **UML Distilled:** A Brief Guide to the Standard Object Modeling Language
   - Martin Fowler, Kendall Scott

