Lecture 2: USDP Overview
Review

• The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.
### Review

<table>
<thead>
<tr>
<th>Diagram</th>
<th>Book Chapters</th>
<th>Purpose</th>
<th>Linage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>11</td>
<td>Procedural and parallel behavior</td>
<td>In UML 1</td>
</tr>
<tr>
<td>Class</td>
<td>3, 5</td>
<td>Class, features, and relationships</td>
<td>In UML 1</td>
</tr>
<tr>
<td>Communication</td>
<td>12</td>
<td>Interaction between objects; emphasis on links</td>
<td>UML 1 collaboration diagram</td>
</tr>
<tr>
<td>Component</td>
<td>14</td>
<td>Structure and connections of components</td>
<td>In UML 1</td>
</tr>
<tr>
<td>Composite structure</td>
<td>13</td>
<td>Runtime decomposition of a class</td>
<td>New to UML 2</td>
</tr>
<tr>
<td>Deployment</td>
<td>8</td>
<td>Deployment of artifacts to nodes</td>
<td>In UML 1</td>
</tr>
<tr>
<td>Interaction overview</td>
<td>16</td>
<td>Mix of sequence and activity diagram</td>
<td>New to UML 2</td>
</tr>
<tr>
<td>Object</td>
<td>6</td>
<td>Example configurations of instances</td>
<td>Unofficially in UML 1</td>
</tr>
<tr>
<td>Package</td>
<td>7</td>
<td>Compile-time hierarchic structure</td>
<td>Unofficially in UML 1</td>
</tr>
<tr>
<td>Sequence</td>
<td>4</td>
<td>Interaction between objects; emphasis on sequence</td>
<td>In UML 1</td>
</tr>
<tr>
<td>State machine</td>
<td>10</td>
<td>How events change an object over its life</td>
<td>In UML 1</td>
</tr>
<tr>
<td>Timing</td>
<td>17</td>
<td>Interaction between objects; emphasis on timing</td>
<td>New to UML 2</td>
</tr>
<tr>
<td>Use case</td>
<td>9</td>
<td>How users interact with a system</td>
<td>In UML 1</td>
</tr>
</tbody>
</table>
Software Development Methodology (SDM)

- A framework for applying software engineering practices with the specific aim of providing the necessary means for developing software-intensive systems

Consisting of two main parts:
- A set of modeling conventions comprising a *Modeling Language* (syntax and semantics)
- A *Process*, which
  - provides guidance as to the order of the activities,
  - specifies what artifacts should be developed using the *Modeling Language*,
  - directs the tasks of individual developers and the team as a whole, and
  - offers criteria for monitoring and measuring a project’s products and activities
Object Oriented Software Development Methodology (OOSDM)

- Specifically aimed at viewing, modelling and implementing the system as a collection of interacting objects
- First appeared in late 1980s
- Categorized as:
  - Seminal (First and Second Generations)
  - Integrated (Third Generation)
  - Agile
- UML was the result of the war among seminal methodologies.
Analysis: Selected Methodologies

Seminal Methodologies
3. RDD (1990)
5. OMT (1991)
6. OSA (1992)
7. OOSE (1992)
8. BON (1992, 1995)
10. Syntropy (1994)

Integrated Methodologies
3. OPEN (1996)
5. EUP (2000, 2005)
6. FOOM (2001)

Agile Methodologies
5. dX (1998)

Process Patterns

Process Metamodels
1. OPF – as part of the OPEN methodology (2001)
2. SPEM (2002)

MDA
Agile History

• First appeared in 1995

• The once-common perception that agile methodologies are nothing but controlled code-&-fix approaches, with little or no sign of clear cut process.

• Essentially based on practices of program design, coding and testing that are believed to enhance software development flexibility and productivity.

• Most agile methodologies incorporate explicit processes, although striving to keep them as lightweight as possible.
Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
Unified Software Development Process (USDP)

- Also known as Unified Process (UP)
- First introduced in 1999
- A refined, simplified, and non-proprietary version of the Rational Unified Process (RUP)
- UML-Based
- Use-Case-Driven
- Architecture-centric
- Iterative and Incremental
Unified Software Development Process (USDP)
Unified Software Development Process

Software lifecycle is decomposed over time in four sequential phases

• **Inception (Vision Milestone)**
  • Define the vision of the product, scope of the project and the business case

• **Elaboration (Architecture Milestone)**
  • Refine the definition of the product
  • Define and baseline an architecture
  • Develop a more precise plan for its development and deployment

• **Construction (Initial Operational Capability Milestone)**
  • Build the product to the point where it can be delivered to its end-users for the first time

• **Transition (Product Release Milestone)**
  • Transition the product to the user community; this includes manufacturing, delivering, training, planning for supporting and maintaining the product

---

**Sharif University of Technology**
Unified Software Development Process

- Each phase can be further broken down into iterations
- An iteration is a complete development loop resulting in a release of an executable increment to the system
- Each iteration consist of nine disciplines perform during the iteration
- For each discipline, defines a set of:
  - Artefacts (work products)
  - Activities (units of work on the artefacts)
  - Roles (responsibilities taken on by development team members)
Disciples

- **Business Modelling**: Business use case model and Business object model
- **Requirements Management**: Use Case Model
- **Analysis and Design**: Design Model
- **Implementation**
- **Test**: system acceptance testing
- **Deployment**: installation and user documentation
- **Project Management**: Project planning and scheduling
- **Configuration and Change Management**: versioning and change management
- **Environment**: needs of organization and project
Phases and Iterations

Inception

Elaboration

Construction

Transition

Prelim Iteration

Arch Iteration

Dev Iteration

Dev Iteration

Trans Iteration

Release

Release

Release

Release

Release

Release

Release

Release

Release
Iterations and Workflows

Phases

- Inception
- Elaboration
- Construction
- Transition

Workflows

- Requirements
- Analysis
- Design
- Implementation
- Test

Iterations

An iteration in the elaboration phase
Features of the iterative approach

- Continuous integration
  - Not done in one lump near the delivery date
- Frequent, executable releases
  - Some internal; some delivered
- Attack risks through demonstrable progress
  - Progress measured in products, not documentation or engineering estimates
Risk Profile of an Iterative Development Process

Inception
Elaboration
Construction
Transition

Risk

Time

Preliminary Iteration
Architect. Iteration
Architect. Iteration
Devel. Iteration
Devel. Iteration
Devel. Iteration
Transition Iteration
Transition Iteration
Post-deployment
Use Cases Drive the Iteration Process

Inception | Elaboration | Construction | Transition

Iteration 1  Iteration 2  Iteration 3

“Mini-Waterfall” Process

- Iteration Planning
- Rqmts Capture
- Analysis & Design
- Implementation
- Test
- Prepare Release

Sharif University of Technology
Inception Phase

• The purpose of Inception is to "get the project off the ground":
  • establishing feasibility - this may involve some technical prototyping to validate technology decisions or proof of concept prototyping to validate business requirements;
  • creating a business case to demonstrate that the project will deliver quantifiable business benefit;
  • capturing essential requirements to help scope the system;
  • identifying critical risks.
Inception – Concerns

• The inception phase is a preparatory stage that attempts to answer the following questions:
  • What is the purpose and objectives of the project? Is it worth the effort?
  • Is the project feasible (e.g. technologically, financially, with current personnel)?
  • Should we buy the system, or build it?
  • Will it be developed now, or built from an existing system?
  • What are the estimated costs and risks?
  • Should we proceed with the project?

• This phase also deals with project planning and project management
  • This includes Gantt charts and plans, budgets, etc.
Inception – Postconditions and Deliverables

<table>
<thead>
<tr>
<th>Conditions of satisfaction</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>The stakeholders have agreed on the project objectives</td>
<td>A vision document that states the project’s main requirements, features, and constraints</td>
</tr>
<tr>
<td>System scope has been defined and agreed on with the stakeholders</td>
<td>An initial use case model (only about 10% to 20% complete)</td>
</tr>
<tr>
<td>Key requirements have been captured and agreed on with the stakeholders</td>
<td>A project glossary</td>
</tr>
<tr>
<td>Cost and schedule estimates have been agreed on with the stakeholders</td>
<td>An initial project plan</td>
</tr>
<tr>
<td>A business case has been raised by the project manager</td>
<td>A business case</td>
</tr>
<tr>
<td>The project manager has performed a risk assessment</td>
<td>A risk assessment document or database</td>
</tr>
<tr>
<td>Feasibility has been confirmed through technical studies and/or prototyping</td>
<td>One or more throwaway prototypes</td>
</tr>
<tr>
<td>An architecture has been outlined</td>
<td>An initial architecture document</td>
</tr>
</tbody>
</table>
Inception Timeline

• An important idea with Inception is that we do not yet know if a project will take place!
  • Often 1 or 2 iterations are required for Inception

• Therefore, since a project may be rejected, it makes sense that the Inception phase should be very short
  • Therefore, if the project gets scrapped, little time (and money) would have been wasted
  • It is not uncommon for Inception to last a few days to a few weeks, maximum
Elaboration Phase

• The purpose of Inception is to understand the problem, whereas Elaboration explores the solution:

  • create an executable architectural baseline;
  • refine the risk assessment;
  • define quality attributes (defect discovery rates, acceptable defect densities, and so on);
  • capture use cases to 80% of the functional requirements;
  • create a detailed plan for the construction phase;
  • formulate a bid that includes resources, time, equipment, staff, and cost.
Elaboration and the Workflows

• In the Elaboration phase, the focus in each of the core workflows is as follows:

• requirements - refine system scope and requirements;
• analysis - establish what to build;
• design - create a stable architecture;
• implementation - build the architectural baseline;
• test - test the architectural baseline.
Elaboration - Concerns

• After Elaboration, project risks are essentially eliminated
  • The Architecture and UI have been approved by customers and managers
  • Technically difficult software components have been implemented, or proof-of-concept code has been created to prove it was possible
  • Cost estimates are finalized, so budgets can be approved
  • Preliminary user manuals have been created and analyzed

• Analysis, architecture and design well underway after Elaboration
Elaboration – Postconditions and Deliverables

<table>
<thead>
<tr>
<th>Conditions of satisfaction</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A resilient, robust executable architectural baseline has been created</td>
<td>The executable architectural baseline</td>
</tr>
<tr>
<td>The executable architectural baseline demonstrates that important risks have been identified and resolved</td>
<td>UML static model, UML dynamic model, UML use case model</td>
</tr>
<tr>
<td>The vision of the product has stabilized</td>
<td>Vision document</td>
</tr>
<tr>
<td>The risk assessment has been revised</td>
<td>Updated risk assessment</td>
</tr>
<tr>
<td>The business case has been revised and agreed with the stakeholders</td>
<td>Updated business case</td>
</tr>
<tr>
<td>A project plan has been created in sufficient detail to enable a realistic bid to be formulated for time, money, and resources in the next phases</td>
<td>Updated project plan</td>
</tr>
<tr>
<td>The stakeholders agree to the project plan</td>
<td></td>
</tr>
<tr>
<td>The business case has been verified against the project plan</td>
<td>Business case</td>
</tr>
<tr>
<td>Agreement is reached with the stakeholders to continue the project</td>
<td>Sign-off document</td>
</tr>
</tbody>
</table>
Construction Phase

- The purpose of Construction is to iteratively enhance and evolve the previously created artefacts into the target system:
  - complete all requirements, analysis, and design
  - evolve the architectural baseline generated in Elaboration into the final system.
Construction and the Workflows

• We can summarize the kind of work undertaken in each workflow during Construction as follows:
  • requirements - uncover any requirements that had been missed;
  • analysis - finish the analysis model;
  • design - finish the design model;
  • implementation - build the Initial Operational Capability;
  • test - test the Initial Operational Capability.
Construction – Postconditions and Deliverables

<table>
<thead>
<tr>
<th>Conditions of satisfaction</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>The software product is sufficiently stable and of sufficient quality to be deployed in the user community</td>
<td>The software product</td>
</tr>
<tr>
<td></td>
<td>The UML model</td>
</tr>
<tr>
<td></td>
<td>Test suite</td>
</tr>
<tr>
<td>The stakeholders have agreed and are ready for the transition of the software to their environment</td>
<td>User manuals</td>
</tr>
<tr>
<td></td>
<td>Description of this release</td>
</tr>
<tr>
<td>The actual expenditures vs. the planned expenditures are acceptable</td>
<td>Project plan</td>
</tr>
</tbody>
</table>
Transition Phase

- The purpose of Transition is the ultimate deployment of the software produced at the end of Construction:
  - conduct beta test and acceptance test, and correct defects;
  - prepare the user sites for the new software;
  - tailor the software to operate at the user sites;
  - modify the software if unforeseen deployment problems arise;
  - create user manuals and other documentation;
  - provide user consultancy;
  - conduct a post-project review.
Transition and the Workflows

We can summarize the kind of work undertaken in each workflow during Transition as follows:

- Requirements - not applicable.
- Analysis - not applicable.
- Design - modify the design if problems emerge in testing.
- Implementation - tailor the software for the user site and correct problems uncovered in testing.
- Test - beta testing and acceptance testing at the user site.
Transition – Postconditions and Deliverables

<table>
<thead>
<tr>
<th>Conditions of satisfaction</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta testing is completed, necessary changes have been made, and the users agree that the system has been successfully deployed</td>
<td>The software product</td>
</tr>
<tr>
<td>The user community is actively using the product</td>
<td></td>
</tr>
<tr>
<td>Product support strategies have been agreed on with the users and implemented</td>
<td>User support plan</td>
</tr>
<tr>
<td></td>
<td>Updated user manuals</td>
</tr>
</tbody>
</table>
References


Sharif University of Technology 33