Software Development Methodologies

Lecturer: Raman Ramsin

Lecture 17

Methodology Engineering
Methodology Engineering

Motivated by the prevalent belief that no one methodology fits all situations.

First introduced by Kumar and Welke in 1992 as a discipline aimed at constructing methodologies to match given organizational settings or specific development projects.

Later came to be known as Method Engineering, a term proposed by Brinkkemper in 1996;

- Definition: “The engineering discipline to design, construct, and adapt methods, techniques and tools for the development of information systems”.

The most well-known subfield is Situational Method Engineering (SME): concerned with the construction/adaptation of a methodology specifically attuned to the project at hand.
Methodology Engineering: Alternative Approaches

- **Ad-hoc**: Concerned with constructing a new methodology from scratch.

- **Paradigm-based**: Concerned with instantiating, abstracting or adapting an existing meta-model in order to produce the target methodology.

- **Extension-based**: Concerned with enhancing an existing methodology with new concepts and properties.

- **Assembly-based**: Concerned with constructing the target methodology or enhancing an existing methodology through reusing parts of other methodologies.
Generic Model for Situational Method Engineering

[Ralyté et al. 2003]
Extension-Based SME

[Domain-driven strategy]  
Select a Meta-pattern  

Start  

Pattern-matching strategy  
Extend a Method  

Completeness strategy

[Pattern-based strategy]

[Start diagram]

[Ralyté et al. 2003]
Paradigm-Based SME

[Diagram showing a process model with steps such as Start, Adaptation, Instantiation, Utilization, Abstraction, Refinement strategy, Simple strategy, Context-driven, Strategy-driven, Pattern-driven, Completeness strategy, and a stop node.]

[Ralyté et al. 2003]
Assembly-Based SME

[Ralyté et al. 2003]
Assembly-Based SME

[Mirbel and Ralyté 2006]
Method Chunk

[Mirbel and Ralyté 2006]
Assembly-Based ME In OPEN/OPF

[Image of diagram showing Framework/metamodel and Repository of process components]

[Henderson-Sellers 2003]
Assembly-Based ME In OPEN/OPF

OPEN Process Framework

Method/Process Metamodel

Repository of Predefined Method/Process Components

Construction Guidelines

Methodology (Including Process)

Method/Process Instance

Step 1. Method/Process Engineer selects Method/Process Components and constructs Methodology

Step 2. Project Manager creates Method/Process Instance by allocating specific resources

[Henderson-Sellers 2003]

Department of Computer Engineering

Sharif University of Technology
OPF Repository

- Contains a range of predefined instances for each class and subclass in the OPF metamodel; e.g.:
  - 30 predefined instances of Activity
  - 160 instances of Task
  - 200 instances of Techniques
  - 76 instances of Role
## OPF: Task-Activity Matrix

<table>
<thead>
<tr>
<th>Task</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>Construct the object model</td>
<td></td>
</tr>
<tr>
<td>Develop and implement resource allocation plan</td>
<td></td>
</tr>
<tr>
<td>- develop iteration plan</td>
<td></td>
</tr>
<tr>
<td>- develop timebox plan</td>
<td></td>
</tr>
<tr>
<td>- set up metrics collection program</td>
<td></td>
</tr>
<tr>
<td>- specify quality goals</td>
<td></td>
</tr>
<tr>
<td>Evaluate quality</td>
<td></td>
</tr>
<tr>
<td>Identify CIRTs (Class, Instance, Role, or Type)</td>
<td></td>
</tr>
<tr>
<td>Map roles onto classes</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
</tr>
<tr>
<td>Write manuals and other documentation</td>
<td></td>
</tr>
</tbody>
</table>

**Key**

1. Project planning
2. Modeling and implementation: OO analysis, design, programming
3. Verification and validation
4. User review
5. Consolidation
6. Evaluation

[Henderson-Sellers 2003]
## OPF: Technique-Task Matrix

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Abstract class identification</td>
<td>x</td>
</tr>
<tr>
<td>Abstraction utilization</td>
<td>x</td>
</tr>
<tr>
<td>Class internal design</td>
<td>x</td>
</tr>
<tr>
<td>Class naming</td>
<td>x</td>
</tr>
<tr>
<td>Collaborations analysis</td>
<td>x</td>
</tr>
<tr>
<td>Complexity measurement</td>
<td></td>
</tr>
<tr>
<td>Contract specification</td>
<td>x</td>
</tr>
<tr>
<td>Coupling measurement</td>
<td></td>
</tr>
<tr>
<td>CRC card modeling</td>
<td>x</td>
</tr>
<tr>
<td>Generalization and inheritance identification</td>
<td>x</td>
</tr>
<tr>
<td>Implementation of services</td>
<td>x</td>
</tr>
<tr>
<td>Implementation of structure</td>
<td></td>
</tr>
<tr>
<td>Inspections</td>
<td></td>
</tr>
<tr>
<td>Interaction modeling</td>
<td></td>
</tr>
<tr>
<td>Package and subsystem testing</td>
<td></td>
</tr>
<tr>
<td>Prototyping</td>
<td>x</td>
</tr>
<tr>
<td>Relationship modeling</td>
<td>x</td>
</tr>
<tr>
<td>Responsibility identification</td>
<td>x</td>
</tr>
<tr>
<td>Role modeling</td>
<td></td>
</tr>
<tr>
<td>Service identification</td>
<td>x</td>
</tr>
<tr>
<td>State modeling</td>
<td>x</td>
</tr>
<tr>
<td>Textual analysis</td>
<td></td>
</tr>
<tr>
<td>Timeboxing</td>
<td></td>
</tr>
<tr>
<td>Unit testing</td>
<td></td>
</tr>
<tr>
<td>Walkthroughs</td>
<td></td>
</tr>
</tbody>
</table>

**Key**
- 1. Code
- 2. Construct the object model
- 3. Develop and implement resource allocation plan
- 4. Evaluate quality
- 5. Identify CIRTs
- 6. Map roles onto classes
- 7. Test

[Henderson-Sellers 2003]
Hybrid Methodology Design

- **Alternative Approaches:**
  - *Instantiation approach:* instantiating an already available process metamodel
  - *Artefact-oriented approach:* devising a seamless complementary chain of artefacts and building the process around it
  - *Composition approach:* using one of the already available libraries of process patterns
  - *Integration approach:* integrating features, ideas and techniques from existing methodologies

- **Hybrid design approach:** using different alternatives from among the above-mentioned for different parts of the process and/or at different levels of abstraction
Hybrid Design Process

1. Requirements
2. Prioritize Requirements
3. Identification of Methodologies
4. Prioritized Requirements
5. Methodology Analysis Results
6. Process-Centred Description of Methodologies
7. Identify Next Abstraction Level
8. Refine and Revise Methodology
9. Define and Apply Hybrid Design Method
10. Integrate Elements into Methodology
11. Methodology Elements
12. Final Methodology
13. If Stabilized and Complete, Finalize Methodology

[Source: Ramsin 2006]
Hybrid Design: Emphasis on Approaches during Design

- Instantiation Approach
- Artefact-Oriented Approach
- Composition Approach
- Integration Approach

[Design Start] Iterations [Design Finish]

[Ramsin 2006]
Hybrid Design: Sample Iterations

Iteration 1

Prepared by:

[Image of a diagram showing the process of hybrid design with sample iterations]

Department of Computer Engineering

Sharif University of Technology

[Ramsin 2006]
Hybrid Design: Sample Iterations (Contd.)

Iteration 2

- Preliminary Analysis
- Detailed Analysis
- Architectural Design
- Detailed Design
- Implementation and Test
- Transition

Iteration 3

- Preliminary Analysis
- Domain Modeling and Requirements Elicitation
- System Specification
- Architectural Design
- Detailed Design
- Implementation and Test
- Transition

Iteration 4

- Preliminary Analysis
- Domain Modeling and Requirements Elicitation
- System Specification
- Architectural Design
- Development Engine
- Plan by Feature
- Design by Feature
- Build by Feature
- Transition
References


