INF5120 – Model-Based System Development

Lecture #9: Method engineering, SPEM and EPF – Service systems
22 March 2010
Brian Elvesæter, SINTEF ICT


Outline

- Definition of Methodology
- Method Engineering (ME) and Process Framework (PF)
- Eclipse Process Framework (EPF)
- Software Process Engineering Metamodel (SPEM)
- Modelling with SoaML
- References
Definition of Methodology

Model-Driven Engineering Framework

Model-Driven Engineering (MDE) Framework

OMG MDA
Metamodelling
UML Profiles & DSLs
Model Transformations
Method Engineering

Reusable MDE Assets
- Method chunks
- Tools and services
- Models and metamodels
- Model transformations
- DSLs and UML profiles
- Reference examples
Challenges for system developers

- Interoperability
- Increasing complexity
- Effectiveness (shorter time to market)
- Increasing quality requirements
- Understand the market needs
- Flexibility
- Technology independence
- Maintainability

Methodology definition (1/2)

- Method
  - Systematic process, technique or mode of inquiry that is used to aid in the creation of a satisfactory software product. [Blum94]
  - Use a method to produce models
- Technique
  - A specific construct supporting a method
- Process
  - A sequence of actions leading to some result
- Method include technique and process
  - CRC method includes CRC technique and CRC process
- Methodology
  - Body of methods
  - Meant to support all software development phases
Methodology definition (2/2)

Method 1
Techn. Process

Method 2
Techn. Process

... 

Method n
Techn. Process

Underlying concepts (paradigm)
E.g. service-oriented software development

Role of the software (methodological) process

The software process ties people and technology together to develop software products in a specific environment.
**Scope of a methodology**

<table>
<thead>
<tr>
<th>Roles</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>project sponsor</td>
<td>envisioning proposal sales setup requirements</td>
</tr>
<tr>
<td>project manager</td>
<td>design &amp; code test deploy train after</td>
</tr>
<tr>
<td>expert user</td>
<td></td>
</tr>
<tr>
<td>business expert</td>
<td></td>
</tr>
<tr>
<td>lead designer</td>
<td></td>
</tr>
<tr>
<td>UI expert</td>
<td></td>
</tr>
<tr>
<td>reuse point</td>
<td></td>
</tr>
<tr>
<td>designer/programmer</td>
<td></td>
</tr>
<tr>
<td>tester</td>
<td></td>
</tr>
<tr>
<td>writer</td>
<td></td>
</tr>
<tr>
<td>trainer</td>
<td></td>
</tr>
<tr>
<td>secretary</td>
<td></td>
</tr>
<tr>
<td>contractor</td>
<td></td>
</tr>
<tr>
<td>night watchman</td>
<td></td>
</tr>
<tr>
<td>janitor</td>
<td></td>
</tr>
</tbody>
</table>

**Project Lifecycle**

**Roles**
- designer/programmer
- writer
- trainer
- secretary
- contractor
- night watchman
- janitor

**Activities**
- envisioning proposal sales setup requirements design & code test deploy train after
- project development
- development
- testing

**Techniques**
- workshops
- use cases
- CRC cards

**Tools**
- Java Project
- 3 month increments
- UML
- JEE

**Qualities**
- Precision
- Accuracy
- Tolerance

**Standards**
- Regression tests
- Object model
- Project plan
- Use cases

**Methodology gives the Who, What, When of key interactions between people**

**Products**
- Planning
- Staging
- Testing

**Teams**
- Project manager
- Documenter
- Designer
- Tester
Method Engineering (ME) and Process Framework (PF)

From the engineering perspective, a method is made up of a set of product models and a set of corresponding process models.

A product model represents the concepts that are used in the method, relationships between these concepts as well as constraints that they have to satisfy.

A process model represents the way to accomplish the development of the corresponding product.
Method engineering process

I. Reengineering of methods into method chunks
- Method reengineering guidelines
- Modular Method Description

II. Assembly-based Situation-specific Method Construction
- Situation Method
- Storage of the method chunks in a method chunks repository

Method chunk

A method chunk is an autonomous and coherent part of a method supporting the realisation of some specific system development or management activity. Such a modular view of methods favours their adaptation, configuration and extension. Moreover, this view permits to reuse chunks of a given method in the construction of new ones.
The responsibility of the *method engineer* is to identify and orchestrate the activities needed in the MDD system development process.

Based on the organisation’s specific needs, the method engineer selects the different process elements, from different process frameworks, and defines an appropriate system development process for an organisation.

The method engineer must ensure completeness of the defined system development process, for example that roles are coherent with roles existing within the organisation and so on.

### MDD process framework overview

- The method engineer builds a system development process based on process elements from the MDD process framework and other process frameworks.
- The project manager adapts the system development process to the project-specific context.
- The system development team uses the process adapted to the specificities of the system development process to build the system.
- Lastly, the method engineer, the application designer and the project manager should provide feedback to the knowledge engineer for the modification, deletion or insertion of new process elements in the process frameworks. This final task allows maintaining an updated source of knowledge within the organisation.

---

**Diagram:**

- Method engineer
- Other process framework
- Process modelling tool
- System development process

**Text:**

- **MDD process framework overview**

---

**Diagram:**

- Project manager
- Method engineer
- System development team
- System development process

**Text:**

- The method engineer builds a system development process based on process elements from the MDD process framework and other process frameworks.
- The project manager adapts the system development process to the project-specific context.
- The system development team uses the process adapted to the specificities of the system development process to build the system.
SHAPE Methodology Framework

Methodology tool architecture

SHAPE Methodology Tool
  • custom methodology creation & management
  • methodology library management

SHAPE Methodology Description Model

Discipline Library
Method Library
Custom Methodology Creation

1. Define overall engineering process
   - Discipline level (process of disciplines)
   - Tool support:
     - Suggest possible order
     - Validation (wrt references processes)

2. Refine engineering process
   - Define tasks for each discipline
   - Tool support:
     - Suggest candidate tasks

3. Select actual Methods
   - Chose methods for each task
   - Tool support:
     - Suggest candidates suggestion & validation
     - Validation (wrt notation / result / engin. process)

Representation in EPF
Eclipse Process Framework (EPF)

http://www.eclipse.org/epf/

The EPF Project: Overview

- EPF is an Open Source project within the Eclipse Foundation

- The goals of EPF are to provide:
  - An extensible framework and tooling for authoring, configuring and publishing processes
  - Exemplary processes - first delivered is OpenUP

- EPF Project initiated in January 2006.

- EPF is NOT:
  - Only applicable for Eclipse Java development.
  - Intended to create the “perfect process”
The EPF Project: Two Audiences

- Process Authors and Coaches (Process Management Team)
  - Tooling for creating and publishing processes
  - Foundational process for starting point
  - Libraries of additional content that can be plugged-in

- Process Consumers (Project Team)
  - Published website of process content for simple browsing
  - Guidance in the form of checklists, concepts, guidelines
  - Browse the content adapted to your experience level

What development teams are facing today

- No **common language** or terminology between processes – redundancy and inconsistencies.
- Knowledge cannot easily be **customized** for different projects or new best practices
- No **central community** or **communication framework** to facilitate convergence of best practices across domains.
A better approach

Standardize representation and manage libraries of reusable method content

Develop and manage processes for performing projects

Content on agile development
Content on managing iterative development
Guidance on non-traditional JEE

JUnit user guide
Content on JEE
Configuration guidelines

Lessons learned from previous projects and revisions

Corporate guidelines on compliance

Process assets

Standard or template processes

Project plan templates

Cohesive Web site customized for my project needs

Project plan templates and optional process instrumentation, relevant for the context of my project

Project goals

- Provide an **extensible framework** and **exemplary tools and content** for software process engineering
  - Extensible framework
    - Metamodel based on OMG SPEM
    - Core extensible process tooling framework
  - Exemplary and extensible tools
    - Method and Process authoring
    - Library management and content extensibility
    - Configuring and publishing
  - Exemplary and extensible process content
    - Range of software development and management processes supporting
      - iterative, agile, and incremental development
      - applicable to a broad set of development platforms and applications
High-level architecture

Some tools and services

- **Method Authoring**
  - Best practices can be captured as a set of reusable method building blocks as defined in the meta-model: roles, work products, tasks, and guidance, such as templates, guidelines, examples, and check lists.
  - A rich-text editor allows you to document method elements, and graphical views present diagrams showing relevant relationships.
  - Reuse is facilitated by allowing you to create a method element as a derivative of another method element through various inheritance type of relationships.

- **Process Authoring**
  - Reusable process building blocks can be organized into processes along a lifecycle dimension by defining e.g. Work Breakdown Structures (WBSs), and when in the lifecycle to produce what work products in which state.
  - The tool allows you to construct reusable chunks of processes through so called capability patterns.
  - A capability pattern may for example define how to define, design, implement and test a scenario or a user story, and this pattern can now be reused in a variety of processes.

- **Library Management and Content Extensibility**
  - An XMI-based library enables persistency and flexible configuration management as well as content interchange for distributed client-server implementations.
  - Method and process content can be packaged into content plug-ins and content packages allowing simple distribution, management and extensibility of content.

- **Configuring and Publishing**
  - A process configuration can be created by selecting a set of content plug-ins and content packages.
  - Optionally, an exemplary process configuration can be used as a starting point, and content plug-ins and content packages added or removed from this exemplary configuration.
EPF Composer

- EPF Composer is a tool platform for process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for development organizations or individual projects.

- Aims to:
  - provide for development practitioners a knowledge base of intellectual capital that allows them to browse, manage and deploy content.
  - provide process engineering capabilities by supporting process engineers and project managers in selecting, tailoring, and rapidly assembling processes for their concrete development process.

EPF Composer Authoring Perspective

- Library View
- Task editor (form based)
- Configuration View
EPF Composer Authoring Perspective

Form based plain text or...

...Rich Text editors

Software Engineering Process Metamodel (SPEM)
SPEM

- Software Process Engineering Metamodel
- Metamodel and UML profile to describe software engineering processes
  - Identifies the typical concepts of a process (process, phase, role, model, etc.)
  - Defines them using UML extensions (stereotypes applied to various elements: class, use cases, operations, etc.)
  - Assigns a characteristic icon to each new item.
- Although the title implies Software Processes, any process can be represented using SPEM.
- EPF uses SPEM.

Method Library

- Method Library
  - All Method Elements are stored in a Method Library
- Method Plug-in
  - A Method Plug-in represents a physical container for Method Packages and Process Packages. It defines a largest granularity level for the modularization and organization of method content and processes.
- Method Configuration
  - a logical subset of a Method Library
- Delivery Process
  - a complete and integrated approach for performing a specific type of project.
Method Content & Process

- **Method Content (Who, What, Why, How)**
  - Highly re-useable information
  - Definition of Roles, Tasks, Work Products and associated relationships
  - Includes Guidance and Categories
  - No timing information

- **Process (When)**
  - End-End sequence of Phases, Iterations, Activities and Milestones that define the development lifecycle.
  - Defines When tasks are performed via Activity Diagrams and/or Work Breakdown Structures

Method Content: Role

- Roles define a set of related skills, competencies and responsibilities.
- Roles are **not** individuals
- Individuals on the development team may play multiple roles.
- Roles Perform Tasks
- Roles are Responsible for Work Products.
### Method Content: Work Product

- Work Products (in most cases) represent the tangible things used, modified or produced by a Task.
- Roles use Work Products to perform tasks and produce Work Products in the course of performing tasks.
- Work Products are the responsibility of a Role.

There are three types of work products:
- Artifact: typically a configuration managed item
- Deliverable: required customer/stakeholder deliverable
- Outcome: “intangible” result of a task such as an installed server or tool.

---

### Method Content: Task

- A Task defines an assignable unit of work (usually a few hours to a few days in length).
- Tasks are performed by Roles (one primary, and optionally additional supporting roles).
- Tasks have a clear purpose, and provide step-by-step descriptions of the work that needs to be done to achieve the goal.
- Tasks modify or produce Work Products.
- Tasks do not define when they are performed in the lifecycle.
### Method Content: Guidance

- **Guidance may be associating with Roles, Tasks, and Work Products.**
- Different types of Guidance depending upon purpose.
- Use Guidance for detailed methodology and supporting information. This will simplify tailoring.
  - For example, Tasks should tell you "what" needs to be done, Guidelines provide detailed "how to".

### Types of Guidance:
- Checklist
- Concept
- Example
- Guideline
- Estimate
- Considerations
- Practice
- Report
- Reusable Asset
- Roadmap
- Supporting Material
- Template
- Term Definition
- Tool Mentor
- Whitepaper

### Process: Capability Patterns (1/2)

- **Capability Patterns define the sequence of related Tasks, performed to achieve a greater purpose.**
- Task can be specialized for the given context (ex. suppress steps, work products)
Process: Capability Patterns (2/2)

- Capability Patterns may be nested and viewed graphically.
- An Activity is an instance of a Capability Pattern.

Activity

(Instance of a capability pattern)

Process: Delivery Process

- Defined using Work Breakdown Structures and/or Activity Diagrams.
- Defines end-end full-lifecycle process.
- May include Iterations, Phases, Milestones (types of Activities).
- This is just one example, any other lifecycle can be defined.
Modelling with SoaML

http://www.modeldriven.com/EnterpriseSoaML.pdf

Designing a methodology around SoaML

- The architectures described with SOA may be business architectures, mission architectures, community architectures or information technology systems architectures – all can be equally service oriented.
- The SOA approach to architecture helps with separating the concerns of what needs to get done from how it gets done, where it gets done or who or what does it.
- SoaML integrates modelling capabilities in support of using SOA at different levels and with different methodologies.
- In particular support for a “contract based” and “interface based” approach which, in general, follow the “ServiceContract” and “ServiceInterface” elements of the SoaML profile.
  - Simple Interfaces – the simple interface focuses attention on a one-way interaction provided by a participant on a port represented as a UML interface.
  - ServiceInterface based – a ServiceInterface based approach allows for bidirectional services, those where there are “callbacks” from the provider to the consumer as part of a conversation between the parties.
  - ServiceContract based – a service contract approach defines service specifications (the service contract) that specify how providers, consumers and (potentially) other roles work together to exchange value.
Business process modelling

- Semi-automated approach based on a model-to-model (M2M) transformation from BPMN models to SoaML models.
- Mapping rules:
  - Task → (Service) Interface
    - As a task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process.
  - Pool → Services Architecture
    - A pool in BPMN stands for a business entity or a participant of a process.
  - Role / Lane → Participant
    - A lane represents a participant or a department in BPMN and is situated in a pool.
  - Annotation (Message) → Message Type
    - The Annotation construct in BPMN is connecting two participants, from which one is providing and another is consuming the service in question.
  - Patterns (roles, tasks, messages) → Service contract
    - The ServiceContract construct is a complex one even in SoaML itself. There is also no single construct in SoaML representing this entity, but rather a certain pattern of objects.
    - Examples of patterns: Two simple tasks following one after another in a container, connected with a sequence flow and associated with one data object.

Work analysis refinement model (WARM)

- WARM is a refinement of the basic business process model which concentrates on "Work Analysis", i.e., which kinds of resources do which kind of work.
  - Specifically, it concentrates on those resources that will form part of the services being identified/developed, and the behaviour that they will be required to exhibit.
  - The kinds of step performed by resources in the model are further categorised as follows:
    - A Human Step is a step performed by a human with no involvement of the service being modelled.
    - An Extended Step is a step in which the intermediate states are of interest to the business, and may have to be remembered. An extended step is a candidate for choreography by a workflow.
    - A Tool Step is a step performed by a human user interacting with a service. The human user will use some form of interactive device (e.g. a GUI) to interact with the service.
    - An Immediate Step is a step that is required to complete as soon as possible, and whose intermediate states are of no concern to the business. It is performed autonomously, with no intervention from a human.
Capabilities identifying candidate services

- Goal-service modelling, which identifies capabilities needed to realize business requirements such as strategies and goals
- Domain decomposition, which uses activities in business processes and other descriptions of business functions to identify needed capabilities
- Existing asset analysis, which mines capabilities from existing applications

Services architecture

- A service architecture is a formal specification of the business requirements that are performed by interacting service participants, without addressing any IT architectural or implementation concerns.
- In this case, the service architecture contains the same information as the original business process and can be treated as a specification for how to realize that business process.
Service interfaces: ShippingService

- The ShippingService service interface refines the ShippingContract service contract.
- The ShippingService service interface involves two roles:
  - The shipper role is a provider role. It is responsible for fulfilling the shipping responsibilities that are given by its type, the shipping interface.
  - The orderer role is responsible for processing the shipping schedule. This is shown by its ScheduleProcessing type.

Service interfaces: InvoicingService

- The InvoicingService service interface refines the InvoicingServiceContract service contract.
- InvoicingService is a ServiceInterface that specifies the conversation, communication protocol, or interaction rules between an orderer and invoicer.
- The protocol details are captured in a UML ownedBehavior.
- The protocol indicates that an orderer must initiate a price calculation before attempting to get the complete price calculation. The orderer must then be prepared to respond to a request (callback, in this case) to process the final invoice.
Message-centric vs. RPC-style services

- There are several SOA interaction paradigms in common use including document centric messaging, remote procedure calls (RPC), and publish-subscribe.
- The decision depends on cohesion and coupling, state management, distributed transactions, performance, granularity, synchronization, ease of development and maintenance, and best practices.
- SoaML supports both document-centric messaging and RPC-style service data.
- Service data is data that is exchanged between service consumers and providers. The data types of parameters for service operations are typed by a DataType, PrimitiveType, or MessageType.
- The Purchasing service interface uses document messaging style service data. Its operation parameters are all typed by SoaML MessageTypes POMessage and InvoiceMessage.
- The Invoicing service interface, by contrast, uses the data types defined.

Service realization: Composite structures

- The Invoicer must provide a design for the implementation or method for each of the service operations.
- Each service operation provided by a service provider must be realized by either a behavior or an action:
  - A Behavior (Activity, Interaction, StateMachine, or OpaqueBehavior) that is the method of the operation
  - An AcceptEventAction (for asynchronous calls) or AcceptCallAction (for synchronous request or reply calls) in an Activity that belongs to the component.
References