INF5120 – Model-Based System Development

Lecture #7: Method engineering and service modelling
3 March 2008
Brian Elvesæter, SINTEF ICT

Based on material developed in the ATHENA (IST-507849), COMBINE (IST-1999-20839), INTEROP (IST-508011), and MODELWARE (IST-511731) research projects.

ATHENA Model-Driven Interoperability (MDI) Framework

MDA & Interoperability
Metamodelling
UML Profiles & DSLs
Model Transformations
Method Engineering

Reusable MDI Assets
- Method chunks
- Tools and services
- Models and metamodels
- Model transformations
- DSLs and UML profiles
- Reference examples
Outline

- Motivation – Why systems development methodology?
- Method engineering
- MDD Process Framework (MDD PF) and Software Process Engineering Metamodel (SPEM)
- Eclipse Process Framework (EPF)
- Service modelling in the COMET-S methodology
- References

Motivation – Why system development methodology?
Challenges for system developers

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

Challenges in system development

- Reduce the risks of
  - Develop inappropriate systems
  - Exceed budget
  - Exceed time (too late to market)
- Want to
  - Develop maintainable systems
  - Reuse
  - Develop consistent systems
  - Modularise
  - Interoperate with existing/other systems
From problem domain to solution domain

Problem domain

Solution domain – Web Services

Solution domain?

Model

System scope

<table>
<thead>
<tr>
<th>ANALYSIS</th>
<th>ARCH.</th>
<th>DESIGN</th>
<th>IMPLEM.</th>
<th>MAINT. &amp; INTEGRATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20% 80%

- It is important to define the scope of a development.
- Otherwise a good answer may be given, but to the wrong question.
  - Scope: to the delivery point?
  - Or to the “end of life” for the system?
Solution

- Model-based software development methodology
  - Precise & standardized notation
  - Well-defined process
  - Well-defined deliverables
  - Well-defined responsibilities
  - Improved communication
  - Documentation
  - Knowledge maintenance
- Project management
- Tool support
- QA
- ...

Model-based system development

- A model is an abstraction
- A model is never complete
  - Focus on certain aspects
- A model is created for a purpose
  - Describe a real world phenomena
  - Describe a new phenomena (not existing, not yet existing)
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)

Methodological process

- 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 process

- Product
- Environment
- Technology
- People
- Environment

The software process ties people and technology together to develop software products in a specific environment.
9 principals for modern software development

1. Architecture-centric
   - Service-oriented architecture
2. Iterative and incremental process
3. Service- and component-orientation
4. Manage a highly dynamic environment
   - Process: Iterative and incremental
   - Software system: Easy to change
5. Model-based
6. “Round-trip” engineering
7. Divide and conquer
8. Quality check
9. Configurable process

Different process for different type of project

- Process depends on type of system
  - Brand new system (very rare)
  - Reengineering (old system exist)
  - Modification (fixing a major problem)
  - Adding a new module (functionality)
Some popular methodologies

- UP (RUP)
  - (Unified Process, Rational unified process)
- KOBRA
- Catalysis
- Select Perspective
- OOram
  - Object Oriented Role Analysis and Modelling
- Lightweight methodologies
  - XP
  - Adaptive Software Development (ASD)
  - SCRUM
  - Crystal Clear

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

- Existing Method
- New domain
- Experience

- Method reengineering guidelines

- Modular Method Description

- Method chunks

- Repository

- Method chunks selection and assembly guidelines

II. Assembly-based Situation-specific Method Construction

- Situational 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.

ATHENA MPCE modelling and execution platform for method engineering

MDD Process Framework (MDD PF) and Software Process Engineering Metamodel (SPEM)
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.

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.

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.

The system development team uses the process adapted to the specificities of the system development process to build the system.
SPEM (1/3)

- 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.

SPEM (2/3)

- Process
- Process Role
- Phase
- Activity
- <<Process>> Process
- <<ProcessRole>> Process Role
- <<Phase>> Phase
- <<Activity>> Activity
Phase: A phase is a specialization of WorkDefinition such that its precondition defines the phase entry criteria and its goal (often called a “milestone”) defines the phase exit criteria. Phases can be sequential or can run in parallel.

WorkDefinition: It’s a kind of operation that describes the work performed in the process. Its main subclass is Activity, but phase, iteration and lifecycle are also subclasses of WorkDefinition.

Relationships: Finally we need to relate the Phase with the WorkDefinitions. For doing so, we will use the UML relationship include.

Example:
Activities and roles (1/2)

- **Activity:** It’s the main subclass of WorkDefinition. An activity describes any part of work executed or assisted by a ProcessRole like tasks, operations and actions. An activity may consist of atomic elements called steps.
- **Process Role:** The process role is the performer of activities. Also defines responsibilities over specific WorkProducts, and defines the roles that perform and assist in specific activities.
- **Process Performer:** The process performer is the performer of higher-level aggregate WorkDefinitions that cannot be associated with individual ProcessRoles. ProcessPerformer represents abstractly the “whole process” or one of its components, and is used to own WorkDefinitions that do not have a more specific owner.
- **Relationships:** Finally we need to relate the Activities with the Process Roles and Process Performers for doing so we will use the UML relationships perform and assist.

Activities and roles (2/2)

- **Example**

```
Customer

Requirements analyst

Business analyst

Documentation generation

Capture requirements

Build the System Boundary model

Build Use Case Scenario model

Build PIM requirements model

Document requirements

perform

perform

perform

perform

perform

perform

<assist>

<assist>

<assist>

<assist>
```

SINTEF ICT
Inputs and outputs (1/2)

- **WorkProduct**: A work product or artefact is anything produced, consumed, or modified by a process. It may be a piece of information, a document, a model, source code, a service provided to an end user and so on. A WorkProduct describes one class of work product produced in a process.
- **Document**: It is a specialization of a WorkProduct that contains information.
- **UML Model**: It is a specialization of a WorkProduct that allows to model static, dynamic and behavioural view of process.
- **UML Profile**: A specialisation of UML to assist in the modelling. This same element is used as illustration of: Tool Mentor; Technique, Template, Guidance, Guideline, Checklist.

Inputs and outputs (2/2)

- **Example:**

![Diagram](image)
Relationships among work products

Example:

Eclipse Process Framework (EPF)

http://www.eclipse.org/epf/
Eclipse Process Framework

- Eclipse Process Framework (EPF) presents a process management tool platform and conceptual framework for authoring, tailoring and deploying development processes.

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

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
      - iterative, agile, and incremental development
      - applicable to a broad set of development platforms and applications
High-level architecture

![High-level architecture diagram]

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 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 Concepts to Create Process Frameworks

Process Framework
- Responsible for creating and modifying work products
- Input or output of performing roles
- Assigned to a role in a creation of modification of a work product
- Express process knowledge for a key area of interest
- Complete process template for a specific type of project
- Used to define processes, can relate to other activities to create work flows

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 – capabilities

- Redesigned tools for authoring, configuring, viewing, and publishing development processes.
- Just-in-time generation of publication previews
- Management of method content using simple form-based user interfaces.
- Intuitive rich text editors for content description
- Processes with:
  - breakdown structure editors
  - workflow diagrams
- Support for many alternative lifecycle models.
- Improved reuse and extensibility capabilities.
- Reusable dynamically-linked process patterns

EMF Composer – Method content

- allows structuring of method contents through schemas
- method content represented as constructs of roles
- input from:
  - best practices
  - Books or publications
  - Standards or regulations
  - Homegrown methods
EMF Composer – Processes

- Represented as workflows or breakdown structures
- Based on different:
  - Development approaches
  - Development cultures
  - Development process representation

Service modelling in the COMET-S methodology
Extending COMET for SOA (2)

Inserting MDD "SOA method chunk" into software development method

4+2 tier reference architecture

Key:
- Component
- Local Adapter
- Local Storage
- Resource Adapter
- Database
- Inter-component communication
Architecture model (1/2)

- The architecture model describes the overall architecture of the system and its partitioning into components.
  - The collaborations of components are described in terms of component interactions, component interfaces and protocols.
- The architecture model describes two aspects of the system; the static (structure) and dynamic (behaviour).
  - The structural model describes the components, their dependencies, and internal component design.
  - The dynamic model describes the component interfaces, interactions and protocols.
- The architecture model is a platform independent system specification.
  - To keep platform independence there is a need to be able to also specify the data types in a platform independent way.

Architecture model (2/2)

- Service architecture structure and internal describes the static aspect of the system.
  - It includes the overall architecture and the partitioning into components as well as the internal design of the components.
- Service interface and interaction specification describes the dynamic aspect of the system.
  - It specifies the interfaces and related protocols as well as defining the interactions between sets of components necessary to provide the required services.
  - The abstract object model provided through an interface are also described.
- PIM data types contains the platform independent data-types used in the architecture model.
UPMS metamodel (partial)

Example scenario
Service architecture structure and internal

Service interface and interaction

Refinements
Service architecture structure and internal

Service interface and interaction
Service architecture structure and internal

References
References (1)

http://www.athena-ip.org/


References (2)

- Rational: The Rational Development process
- Paul Allen, Stuart Frost, Component-Based Development for Enterprise Systems, Applying the SELECT Perspective. SIGS Book and Multimedia 1998