INF5120 – Model-Based System Development

Method Engineering, Software Process Frameworks, SPEM/EPF, ISO 24744, FACESEM/ESSENCE

11 March 2013
Outline

- Method Engineering (ME)
  - Why Software Development Methodology?
  - Software Process/Method Frameworks
- SPEM – Software & Systems Process Engineering Metamodel
  - Eclipse Process Framework (EPF) Composer
- ISO 24744 – Software Engineering Metamodel for Development Methodologies
- Essence – Kernel and Language for Software Engineering Methods
  - EssWork Practice Workbench
Why Software Development Methodology?
Challenges for system developers

- Interoperability
- Increasing complexity
- Effectiveness (shorter time to market)
- Increasing quality requirements
- Understand the customer/market needs
- Flexibility
- Technology independence
- Maintainability
- …
From problem domain to solution domain

Problem domain

Solution domain?

Solution domain – Web Services

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

- **Quality**
  - Precision
  - Accuracy
  - Tolerance
  - Regression tests
  - Object model
  - Project plan
  - Use cases

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

- **Standards**

- **Techniques**
  - Planning
  - Staging
  - Testing
  - Workshops
  - Use cases
  - CRC cards

- **Activities**
  - Use cases

- **Roles**
  - Project manager
  - Documenter
  - Designer
  - Tester

- **Teams**
  - Personality

**Methodology**
- Methodology gives the **Who**, **What**, **When** of key interactions between people.
Role of the software process

The software process ties people and technology together to develop software products in a specific environment.
Lighter methodologies are more effective, but have limits.
Today’s process aids landscape

Just how different are all these process aids when it comes to helping people get their job done?
Software Process/Method Frameworks
Method engineering process

I. Reengineering of methods into method chunks

Method reengineering guidelines

II. Assembly-based Situation-specific Method Construction

Existing Method
New domain Experience …

Method chunks
selection and assembly guidelines

Situational Method

Method chunks Repository

Modular Method Description

Storage of the method chunks in a method chunks repository

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.
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.
Method architecture

- **Method**
  - A composition of practices

- **Practice**
  - A repeatable approach to doing something with a specific purpose in mind.
  - Provides a systematic and verifiable way of addressing a particular aspect of the work at hand.
  - It has a clear goal expressed in terms of the results its application will achieve.
  - It provides guidance to practitioners.

- **The Kernel**
  - The kernel includes the essential elements of software engineering.
  - The Kernel provides a concrete framework that can be used on every project, allowing projects to identify the key matters of interest and apply practices to direct the project towards success.

- **The Language**
  - The methods, practices and the essential elements are described in the Language.
  - EPF uses the Software & Systems Engineering Process Metamodel (SPEM) language specified by OMG.
MODUS Methodology Framework

- **MODUS Framework**
  - [http://modus.modelbased.net](http://modus.modelbased.net)
  - Library of reusable and composable Practices

- **INF5120 Methodology**
  - [http://inf5120.modelbased.net](http://inf5120.modelbased.net)
  - Methodology Wiki with Practices providing guidance

---

**EPF Composer**

- Method Library
- Practices

- Method Engineer
  - defines
  - uses

- Custom Methodology
  - ... 

- builds and publishes

- **Methodology Wiki**
  - guides development

- Developers
  - build and publish project-specific methodology websites

- Testers
SPEM – Software & Systems Engineering Process Metamodel
SPEM

- Software & Systems 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 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
SPEM 2.0 metamodel
SPERM 2.0 UML profile – Method content
SPEM 2.0 UML profile – Process
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 associated 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
Example – Method content
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.
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.
Eclipse Process Framework (EPF) Composer

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

1. Standardize representation and manage libraries of reusable method content
   - Content on agile development
   - Content on managing iterative development
   - Guidance on serialized Java beans

2. Develop and manage processes for performing projects
   - Lessons learnt from previous project and iteration
   - Corporate guidelines on compliance

3. Cohesive Web site customized for my project needs

4. 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
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.
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.

Diagram:
- OpenUP Library
  - DSDM Plug-in for OpenUP
    - extends
  - OpenUP Plug-in
    - depends on
  - Base Concepts Plug-in
EPF Composer Authoring Perspective

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

Form based plain text or... 

...Rich Text editors
ISO 24744 – Software Engineering Metamodel for Development Methodologies
Comparison at a glance

**SPEM**
- Process aspects only
- Method domain only
- Obscure and informal extension mechanisms
- Unable to express non-trivial scenarios
- Highly dependent on UML

**ISO 24744**
- Process, product and quality aspects
- Method and endeavour domains
- Easy and intuitive extension mechanisms
- Copes with complex scenarios
- Self-contained
Comparison (graphically)
History of ISO 24744

- ISO/IEC 24744 is an international standard
- Based on experience:
  - Fujitsu, Microsoft, SQI, JPL, plus other 20
  - Input from 25+ countries
  - Field consulting from UTS, Neco
  - Top-notch academic research
  - Errors of previous attempts
- Led from academia
- 2003-2007
- Submission to OMG, February 20th, 2012
## Moving on to metamodelling

### Task Kind
- **Name**: Code Writing
- **Purpose**: To write code...
- **MinCapLev**: 1

### Task
- **Start**: 12-Sep-07
- **End**: 15-Sep-07
- **Duration**: 3
- **Language**: C#
Putting things in context

<table>
<thead>
<tr>
<th>Task Kind</th>
<th>Name</th>
<th>Purpose</th>
<th>MinCapLev</th>
<th>Task</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Code Writing&quot;</td>
<td>&quot;To write code...&quot;</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Code Writing

Language

12-Sep-07
15-Sep-07
3
"C#"
Metamodel areas

Methodology elements

Templates

Resources

Endeavour elements
Metamodel overview

MethodologyElement

Template
  +name

StageKind

WorkProductKind
  +description

WorkUnitKind
  +purpose
  +minimumCapabilityLevel

ModelUnitKind
  +definition

ProducerKind
  +name

Producer
  +name

Stage
  +creationTime
  +lastChangeTime
  +status

WorkUnit
  +startTime
  +endTime
  +duration

ModelUnit

Outcome
  +description
  +minimumCapabilityLevel

Resource

Language

Constraint
  +expression

Guideline
  +description

EndeavourElement
Summary

SEMDM / ISO 24744:

- Captures what developers do (endeavour) by enhancing expressiveness
- Integrate endeavour and method domains (powertype patterns)
- Integrate process and product
- Address quality and capability issues (MinCapabilityLevel attribute)
- Support for affordable extensibility (regular OO mechanisms)
SEMDM Diagrams – Graphical Notation

- Lifecycle diagrams, which represent the overall structure of a method (or part of it).
- Enactment diagrams, which represent a specific endeavour (or part of it) and its relationship to the corresponding method.
- Dependency diagrams, which represent the abstract support/dependency relationships among the major components (i.e. producer kinds, work unit kinds and work product kinds) of a methodology.
- Process diagrams, which describe the details of the process kinds used in a method.
- Action diagrams, which show the detailed usage interactions between task kinds and work product kinds.
Essence – Kernel and Language for Software Engineering Methods

http://www.semat.org
The Kernel is described using a small subset of the Language.

A stripped-down, lightweight set of definitions that captures the essence of effective, scalable software engineering in a practice independent way.
Alphas: The Essential Things to Work With
Alphas: Example

- Conceived
- Bounded
- Coherent
- Acceptable
- Addressed
- Fulfilled

The need for a new system has been agreed.

The purpose and theme of the new system are clear.

The requirements provide a coherent description of the essential characteristics of the new system.

The requirements describe a system that is acceptable to the stakeholders.

Enough of the requirements have been addressed to satisfy the need for a new system in a way that is acceptable to the stakeholders.

The requirements that have been addressed fully satisfy the need for a new system.
Activity Spaces: The Essential Things to Do

- Explore Possibilities
- Understand Stakeholder Needs
- Ensure Stakeholder Satisfaction
- Use the System

- Understand the Requirements
- Shape the System
- Implement the System
- Test the System
- Deploy the System
- Operate the System

- Prepare to do the Work
- Coordinate Activity
- Support the Team
- Track Progress
- Stop the Work

Understand Stakeholder Needs
Activity Spaces: Examples

Scrum Essentials Practice

Specify the Software

Activity Space

Identify Use Cases

Specify Use Cases

Activity

Activity Predecessor Relationship

Sprint Planning Meeting

Daily Scrum

Sprint Review

Sprint

Retrospective
Focus areas

- Embodies the essence of software engineering in a kernel.
- Works with methods in an agile way that are as close to practitioners’ practice as possible.
- Applies the principle of “separate of concerns”, focusing on the things that matter the most.
- Focuses on helping the least experienced developers over helping more experienced developers.
- Reflects an understanding that the majority of the development community is interested in...
  - the use of methods, not their definition.
  - practice, not process or method engineering.
  - intuitive and concrete graphical syntax, not formal semantics.
The Language

Small, simple and flexible.
The Language: Small, Simple and Flexible

Alpha
< describes
organizes
<

Alpha State
< has
< evidences

Work Product
produces / updates

Activity Space
targets
<
organizes

Activity
produces / updates

Competency
< involves
< requires

Pattern
Can be added to anything

Resource
The Kernel provides the blue print

- Alpha
- Alpha State
- Activity Space
- Competency

< involves targets > has
Practices add the detail

- **Alpha** has **Alpha State**
- **Alpha State** targets **Activity Space**
- **Activity Space** involves **Competency**
- **Activity Space** involves **Work Product**
- **Work Product** produces **Activity**
- **Activity** involves **Competency**

ICT
Resources and Patterns enable extension

- Competency
- Alpha
- Alpha State
- Activity Space
- Work Product
- Activity
- Pattern
- Resource

Can be added to anything
Some example uses of resources

- Competency involves targets.
- Alpha has Activity Space, which produces Work Product.
- Work Product helps create Resources (such as Templates & Examples).
- Activity involves targets and requires resources.
- Activity Space involves Competency.
- Resources help people to develop.
- Resources (such as Scripts & Tool Mentors) help perform.
- Resources (such as Training Courses) help create.
Some example uses of Patterns

- **Alpha State**: Describes and organizes activity work product that evidences competency.
- **Activity Space**: Sequences and filters activities.
- **Activity**: Requires, involves, and produces updates.
- **Alpha**: Has activity space and work product.
- **Resource**: Can be added to anything.
- **Pattern**: Helps perform and up profile teams and team members.
- **Competency**: Qualifies team membership such as team roles and team structures.

Patterns that:
- **Synchronize Alpha State progression**
- **Sequence and filter activities**
- **Describe alternative ways to perform activities**
- **Describe different approaches**

Examples:
- Checkpoint or Milestone
- Phase
- Team Role
Language: Structure of the Metamodel

- ActivitySpaceAndActivity
- Competency
- AlphaAndWorkProduct
- View
- Foundation
Language: Foundation
Language: Alpha and Work Product

- **Foundation::LanguageElement**
  - **name**: String
  - **icon**: GraphicalElement[0..1]
  - **briefDescription**: String
  - **description**: String

- **Foundation::BasicElement**
  - **name**: String
  - **icon**: GraphicalElement[0..1]
  - **briefDescription**: String
  - **description**: String

- **AlphaContainment**
  - **name**: String
  - **lowerBound**: Integer
  - **upperBound**: Integer

- **AlphaAssociation**
  - **name**: String
  - **end1LowerBound**: Integer
  - **end1UpperBound**: Integer
  - **end2LowerBound**: Integer
  - **end2UpperBound**: Integer

- **WorkProductManifest**
  - **lowerBound**: Integer
  - **upperBound**: Integer

- **State**
  - **name**: String
  - **description**: String
  - **isSufficientLevel**: Boolean

- **LevelOfDetail**
  - **name**: String
  - **description**: String

- **Checkpoint**
  - **name**: String
  - **description**: String

- **Alpha**
  - **name**: String
  - **description**: String

- **WorkProduct**
  - **name**: String
  - **description**: String

- **AlphaContainment**:
  - **lowerBound**: Integer
  - **upperBound**: Integer

- **AlphaAssociation**:
  - **subordinateAlpha**: Integer
  - **superAlpha**: Integer

- **WorkProductManifest**:
  - **lowerBound**: Integer
  - **upperBound**: Integer

- **State**:
  - **name**: String
  - **description**: String

- **Checkpoint**:
  - **name**: String
  - **description**: String

- **LevelOfDetail**:
  - **name**: String
  - **description**: String
Language: Activity Space and Activity
EssWork Practice Workbench

http://www.ivarjacobson.com/EssWork_Practice_Workbench/
Sources

- **OMG Essence specification**

- **Scrum Guide**

- **Practice authoring tool**
  - EssWork Practice Workbench
The Practice Explorer shows Practice Workbench projects.

The Essence Kernel project contains the elements defined in the OMG Essence specification.

Alphas that represent the essential things to work with.

Activity Spaces that represent the essential things to do.
ETextile, Guideline and Card views

When selecting an element in the Practice Explorer you can switch between different views.

The ETextile Source view provides the main editor for authoring the practice using plain text and annotations.

The Guideline Preview renders how the guideline will be presented in HTML.

The Overview Card Preview renders the card presentation.
Scrum Essentials

The Scrum practice is created as a separate practice project in the Practice Workbench.

The Scrum practice extends the Essence Kernel by providing more detailed guidance.

Drag and drop the relevant Alphas to extend from the Essence Kernel into the Scrum practice project.

Drag and drop the relevant Activity Spaces to extend from the Essence Kernel into the Scrum practice project.
Scrum Roles

Scrum roles are represented as Patterns.

Product Owner (Guideline Preview)

Product Owner (Card Preview)
Scrum Sprint

- Sprint is represented as a sub-alpha of Work
- The Sprint has States with Checkpoints
- The Sprint has associated the Work Product Sprint Backlog that contains the set of Product Backlog items selected for the Sprint, and the plan for delivering the product Increment
- Sprint in Under Control State
- Under Control (State Card Preview)

Sprint (Card Preview)
Scrum Activities

The Scrum events (except the Sprint which is represented as an Alpha) are represented as Activities.

The Sprint Planning Meeting activity provides guidance on how to achieve the Planned state of the Sprint.