Essence – Kernel and Language for Software Engineering Methods

Arne.J.Berre@sintef.no
Introduction to Essence

Ivar Jacobson, IJI

Diagram:
- Methods
- Practices
- The Kernel
- The Language

- Methods are composed of
- Practices are described using kernel elements
- The Kernel is defined in terms of
  - The Language
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

Customer

Opportunity < provide

Stakeholders

use and consume

helps to address

Solution

Requirements < demand

Software System < fulfils

set up to address

Support

Endeavor

Work < performs and plans

Team < produces

Way of Working < applies

< guides

updates and changes
Requirements

Description
What the software system must do to address the opportunity and satisfy the stakeholders.

It is important to discover what is needed from the software system, share this understanding among the stakeholders and the team members, and use it to drive the development and testing of the new system.

Associations
scopes and constrains : Work

- 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**
Activity Spaces: Examples

Scrum Essentials Practice

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Activity Space

Specify the Software

- Identify Use Cases
- Specify Use Cases

Activity

Activity Predecessor Relationship
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

- Methods
- Practices
- The Kernel
- The Language

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

- **Alpha**
  - < involves targets >
  - < has Alpha State >
  - < describes, organizes >
- **Alpha State**
  - < evidences >
  - < targets >
  - < results in Work Product >
  - < organized by Activity Space >
- **Work Product**
  - ^ produces, updates Activity
- **Activity Space**
  - < organizes >
- **Activity**
  - < requires Competency >
  - < organized by Activity >
- **Resource**
  - Can be added to anything
- **Pattern**
The Kernel provides the blueprint

- **Alpha**<br>  - **Alpha State**<br>    - **targets**<br>      - **Activity Space**<br>        - **involves**<br>          - **Competency**
Practices add the detail
Resources and Patterns enable extension

- Alpha
  - has
  - evidences
  - involves

Alpha State
- targets
  - results in
  - involves

Activity Space
- requires

Competency

Work Product
- produces

Activity
- progresses by
- updates
Some example uses of resources

- **Alpha State**: - describes - organizes
- **Work Product**: - evidences
- **Activity Space**: - targets - results in - organizes
- **Activity**: - requires
- **Competency**: - involves - requires - helps people to develop

**Resource**
- Can be added to anything

**Pattern**
- Helps create Resources (such as Templates & Examples)

**Pattern**
- Helps perform Resources (such as Scripts & Tool Mentors)

**Pattern**
- Helps people to develop Resources (such as Training Courses)
Some example uses of Patterns

**Patterns that synchronize Alpha State progression**
- e.g. Checkpoint or Milestone
  - Patterns that synchronize Alpha State progression
- e.g. Phase
  - Patterns that sequence and filter activities

**Patterns that sequence and filter activities**
- ends with
- results in set of

**Patterns that describe alternative ways to perform and activity**
- e.g. Different Approaches
- < helps perform e.g. Different Approaches
- < helps up profile teams and team members
- < requires e.g. Team Role
- < qualifies responsibilities

**Patterns that qualify team membership such as Team Roles and Team Structures**
Language: Structure of the Metamodel

- ActivitySpaceAndActivity
- Competency
- AlphaAndWorkProduct
- View
- Foundation
Language: Competency

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

- **Competency**
  - **CompetencyLevel**
    - name: String
    - briefDescription: String
    - level: Integer
    - possibleLevel

*
Language: View

```
Foundation::LanguageElement

ViewSelection
+name: String
+description: String

FeatureSelection
+featureName: String

* IncludedViewSelection

* ConstructSelection

1..* FeatureSelection
+featureName: String
```

```
* +constructSelection
1..* +construct

* +featureSelection

1..* ViewSelection
+name: String
+description: String
+includedViewSelection
```
Practices and Optional Kernel Extensions

Ian Spence, IJI

Practices – Conformance levels and examples

Optional Kernel Extensions – help to get started.

Is defined in terms of

The Language

The Kernel

Practices

Methods

Are composed of
Optional Kernel Extensions

- **Business Analysis Extension** – adds two Alphas, Need and Stakeholder Representative, to drive forward the Opportunity and the Stakeholders.

- **Development Extension** – adds two Alphas, Requirement Item and System Element to drive forward the Requirements and the Software System. As well as System Element it also adds Bug to monitor the health of the Software System. Bugs are an important thing to monitor, track and address in any software development endeavor, and one which will inhibit, rather than drive, progress being made to the Software System.

- **Task Management Extension** – adds three Alphas, Team Member, Task and Practice Adoption, to drive forward the Team, Work and Way-of-Working.
Development Extension

A specific condition or capability that the Software System must address has been identified.

The Requirement Item is ready to be implemented.

The Requirement Item is implemented in the Software System and demonstrated to work.

Successful implementation of the Requirement Item in the Software System has been confirmed.
Development Extension (2)

Bug

- Detected
- Located
- Fixed
- Closed

- An error, fault or flaw in the Software System is observed and logged.
- The cause of the Bug in the Software System has been found.
- The Bug has been removed from the Software System.
- The removal of the Bug from the Software System has been confirmed.

Inhibits

Software System

- Architecture Selected
- Demonstrable
- Usable
- Ready
- Operational
- Retired
A system element has been identified as part of the Software System and its responsibilities and its position in the Software System are clear.

The System Elements interfaces have been agreed.

The System Element has been implemented and tested, and is believed to be ready for integration into the Software System.

The System Element has been verified and is ready for live use as part of the Software System.
Practice Description Conformance Levels - Proposed

• 0 - Narrative
  – A referenceable resource written in free-format text

• 1 – Illustrative
  – Free-format content types using the conceptual model & tags.

• 2 - Modelled
  – Navigable and composable into a reference

• 3 – Actionable
  – Drives progress through Alphas and states
  – Essential and optional elements clear etc

• 4 – Fully Conformant
  – To be defined
Practice Examples

• Scrum
• User Story
• Multi-Phase Waterfall – V-Model
• Munich Re Collaboration Models
  – Exploratory, Standard, Maintenance, Support

• With more to come…..
  – Catalogue of short-form one-page descriptions in development
Extension for KUALI-BEH

Miguel Trujillo and Hanna Oktaba, UNAM

- KUALI-BEH main concepts
- KUALI-BEH Practices in Essence language
- KUALI-BEH Method concept in Essence
- KUALI-BEH Enactment in Essence
- KUALI-BEH Essence Kernel extensions and practices
KUALI-BEH

- KUALI-BEH main concepts
## Alignment with KUALI-BEH

### KUALI-BEH Practices in Essence language

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>[name]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>[objective]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[expected characteristics]</td>
<td>[expected characteristics]</td>
</tr>
</tbody>
</table>

### Verification Criteria

| [criterionA, criterionB, …] |

### Instance States

| [stateA, stateB, …] |

### Guide

<table>
<thead>
<tr>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[activity]</td>
</tr>
</tbody>
</table>

### Input

| Output |

### Tasks (optional)

<table>
<thead>
<tr>
<th>Resource (optional)</th>
<th>Knowledge and Skills</th>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>[list of proposed tools]</td>
<td>[abilities, competences, attainments, …]</td>
<td>[measureA, measureB, …]</td>
</tr>
</tbody>
</table>

| [toDoThis, …, toDoThat, …] |
Alignment with KUALI-BEH

- KUALI-BEH Method concept in Essence

- Method is the top level composition of practices for an Endeavour

- A method is an articulation of a coherent, consistent and complete set of practices, with a specific purpose that fulfills the stakeholder needs under specific conditions
Alignment with KUALI-BEH

- KUALI-BEH Enactment in Essence
Alignment with KUALI-BEH

• KUALI-BEH Essence Kernel extensions and practices

  – KUALI-BEH Task Management Extension - Work
    – New sub-ordinate ALPHAs

  – KUALI-BEH practice authoring - Way-of-Working
    – Practice Template
    – Method Template

  – KUALI-BEH method usage - Work and Team
    – Method Enactment Board
    – Practice Instance Board
• The Essence submission has come a long way in terms of compatibility with SPEM
  – Support for breakdown structures
  – The ability to share resource elements

• Agreement has been reached on the need to support additional features even though the details have not yet been worked out
  – Categorization
  – Grouping of elements for versioning

Todd Fredrickson (and Bruce MacIsaac), IBM
Working with both Essence and SPEM

• There are still major differences between SPEM and Essence
  – Terminology
  – Emphasis and Coverage
  – Approach

• Neither standard currently supports the automatic migration to the other
Key features that are not covered by SPEM

• The Essence language introduces some key concepts that don’t currently exist in SPEM, but represent the current direction of methods
  – Practices
  – Alphas
  – Activity Spaces

• Look at what would be required to incorporate new language features introduced as part of Essence into SPEM

• Essence adds the concept of the kernel which represents a standard set of content to be used as basis for development
ISO 24744 introduces a Dual-Layer modeling
- Powertypes to relate (language) concepts in the Method(ology) and Endeavor domains
- Clabjects (instances) to endow properties at enactment
- Powertypes and Clabjects are not compatible with MOF

The Essence Dynamic Semantics is compatible with MOF
- Abstract super classes (at level 1) from which you can define sub-types
- Ensure that occurrences at level 0 are endowed with the properties needed at enactment

ISO 24744 and Essence expresses the same, but in two different ways.
Essence is MOF-based, ISO 24744 is not!
Alignment with SEMDM (ISO 24744)

• Essence separates the Kernel from the Language.
  – This is similar to the dual-modeling approach of ISO 24744.
• Some of the ISO 24744 concepts map to elements in the Kernel (or optional Kernel extensions)
  – Task maps to Task (which is an Alpha in the optional Kernel extension)
• Some of the ISO 24744 concepts map to concepts in the Essence Language
  – WorkProductKind maps to WorkProduct (language construct in Essence)
  – WorkProduct maps to my_WorkProduct (abstract super class in Essence)
• Difference above between ISO 24744 and Essence, because of different use of naming conventions
Satisfaction of Requirements – Kernel

6.5.1.1 Domain model – 7 *alphas* and 15 *activity spaces* ✔
6.5.1.2 Key conceptual elements – *Alphas* ✔
6.5.1.3 Generic activities – *Activity spaces* ✔
6.5.1.4 Kernel elements – *Alpha* and *activity space descriptions* ✔
6.5.1.5 Scope and coverage – *See examples* ✔
6.5.1.6 Extension – *By construction* ✔
Satisfaction of Requirements – Language

6.5.2.1.1 MOF metamodel ✔
6.5.2.1.2 Static and operational semantics ✔
6.5.2.1.3 Graphical syntax ✔
6.5.2.1.4 Textual syntax ✔
6.5.2.1.5 SPEM 2.0 metamodel reuse – Not reused – but mapping for migration is in progress. ✔
6.5.2.2.1 Ease of use – separation of concerns ✔
6.5.2.2.2 Separation of views – Views target practitioners ✔
6.5.2.2.3 Specification of kernel elements ✔
6.5.2.2.4 Specification of practices ✔
6.5.2.2.5 Composition of practices – Algebra defined, some syntax TBD ✔
6.5.2.2.6 Enactment of methods ✔
Satisfaction of Requirements – Practices

6.5.3.1 Examples of practices – See Annex C ✔

6.5.3.2 Existing practices and methods – ✔
Recommendation

• It is agreed to incorporate KUALI-BEH concepts as a separate annex, with minimal needed changes to Essence.
• It is feasible to have both SPEM and FACESEM/Essence as OMG standards.
• There are key features and concepts in Essence that the user community would benefit from being able to use – sooner rather than later.
• An activity should be started to ensure the further evolution of SPEM – separate from the FACESEM/Essence submission and finalisation process.
• It is a goal to have a consistent family of standards in this area – with migration paths between.
Next steps

- Establishment of Evaluation team
- Revised submission date, November 12th, 2012
The Essence of Software Engineering

Applying the SEMAT Kernel

Rough Cuts

Ivar Jacobson
Ng Pan-Wei
Paul McMahon
Ian Spence
Svante Lidman

Book is available now – Safaribooksonline/Addison Wesley
To Do

- Practice area (library of practices)
- Tools – open source
- Education (4-5 areas of education …)
- Books, publications, ….
- Theory area
- Research topics
- Chapters of SEMAT