Essence (with KUALI-BEH) – Kernel and Language for Software Engineering Methods

Arne J. Berre, SINTEF
Ivar Jacobson, IJI
Michael Striewe, University of Duisburg-Essen
Brian Elvesæter, SINTEF
Ian Spence, IJI
Shihong Huang, FAU
Paul E. McMahon, PEM Systems
Miguel Ehécatl Morales Trujillo, Hanna Oktaba, Magdalena Dávila Muñoz, UNAM (KUALI-BEH)
Todd Fredrickson, IBM
Hiroshi Miyazaki, Fujitsu
Tom Rutt, Fujitsu
Ed Seidewitz, Model Driven Solutions
Outline

• Plan for the presentation – Arne J. Berre

• Introduction to Essence – Ivar Jacobson
• Revised version of Essence Language – Michael Striewe
• Revised version of Essence Kernel – Ian Spence
• Agreed integration with KUALI-BEH – Miguel Trujillo
• Harmonisation and relationship with SPEM – Todd Fredrickson
• Satisfaction of RFP requirements
• Recommendations
Introduction to Essence

Ivar Jacobson, IJI
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

Revision: Only one Alpha Revised
**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

---

**UPDATE:**

Previously – Sufficiently Described

**UPDATE:**

Previously – Satisfactorily Addressed
UPDATE:
Previously – Involve Stakeholders.
Revision:
Only one Activity Space Revised
Activity Spaces: Examples

Scrum Essentials Practice

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

Activity Space

Specify the Software

Activity

Identify Use Cases

Specify Use Cases

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.
Small, simple and flexible.

The Language

Michael Striewe, UDE
The Language: Small, Simple and Flexible

- Alpha
  - < has
  - < describes
  - < organizes
  - < evidences

- Alpha State
  - < targets
  - < evidences

- Work Product
  - ^ produces
  - ^ updates

- Activity Space
  - ^ organizes

- Activity
  - ^ requires

- Competency
  - < involves
  - Can be added to anything
The Kernel provides the blue print

- Alpha
- Alpha State
- Activity Space
- Competency

Alpha has targets involving Activity Space involves Competency.
Practices add the detail

- Alpha State
  - Alpha
  - Competency
  - Activity Space
  - Activity
  - Work Product
  - evidences
  - targets

- involves
- produces
- updates
- requires
Resources and Patterns enable extension

Alpha has Alpha State

Alpha State evidences Work Product

Activity Space progresses by Activity

Activity involves Competency

Resource and Pattern can be added to anything
Some example uses of resources

- **Competency**
- **Alpha State**
- **Activity Space**
- **Work Product**
- **Alpha**
- **Resource**
- **Pattern**
- **Can be added to anything**

Resources:
- **Resources** (such as Templates & Examples)
- **Resources** (such as Scripts & Tool Mentors)
- **Resources** (such as Training Courses)
Some example uses of Patterns

- **Alpha State**: Alpha involves targets.
  - Alpha State has activity space.
  - Alpha describes work product.
  - Alpha produces or updates work product.
  - Alpha helps perform activity.

- **Activity Space**: Activity space organizes activity.
  - Activity space involves competency.
  - Activity space requires resources.

- **Competency**: Competency involves pattern.
  - Competency has resource.

- **Resource**: Resource can be added to anything.
  - Resource requires pattern.
  - Resource helps up profile teams and team members.

- **Pattern**: Patterns that help sequence and filter activities.
  - Patterns that help progress Alpha State.
  - Patterns that help up profile teams and team members.
  - Patterns that describe alternative ways to perform and activity.

- **Work Product**: Work product organizes activity.
  - Work product produces or updates work product.
  - Work product helps perform activity.

- **Alpha**: Alpha involves competency.
  - Alpha describes work product.
  - Alpha organizes activity.

- **e.g. Checkpoint or Milestone**: Checkpoint or milestone synchronizes progress of Alpha State.
  - Checkpoint or milestone has competency.

- **e.g. Phase**: Phase sequences activity.
  - Phase involves competency.

- **e.g. Different Approaches**: Different approaches help perform activity.

- **e.g. Team Role**: Team role helps up profile teams and team members.

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

- ActivitySpaceAndActivity
- Competency
- AlphaAndWorkProduct
- View
- Foundation

Diagram showing relationships between the concepts.
Language: Foundation
Practices and Optional Kernel Extensions

Ian Spence, IJI

- Methods
- Practices
- The Kernel
- The Language

Is defined in terms of

Are composed of

Practices – Conformance levels and examples

Optional Kernel Extensions – help to get started.
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.
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)

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.
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 Element’s 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
Alignment with 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
Alignment with 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>

#### Objective

<table>
<thead>
<tr>
<th>[objective]</th>
</tr>
</thead>
</table>

#### Source

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

#### Verification Criteria

<table>
<thead>
<tr>
<th>[criterionA, criterionB,...]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### Instance States

<table>
<thead>
<tr>
<th>[stateA, stateB,...]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### Guide

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

#### Activity

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Tasks (optional)

<table>
<thead>
<tr>
<th>[ToDoThis, ...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ToDoThat, ...]</td>
</tr>
</tbody>
</table>

#### Resource (optional)

<table>
<thead>
<tr>
<th>[list of proposed tools]</th>
</tr>
</thead>
</table>

#### Knowledge and Skills

<table>
<thead>
<tr>
<th>[abilities, competences, attainments, ...]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### Measures

<table>
<thead>
<tr>
<th>[measureA, measureB, ...]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
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
Alignment with SPEM – and EPF/RMC

• 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!
• 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
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
Book is available now – Safaribooksonline/Addison Wesley