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
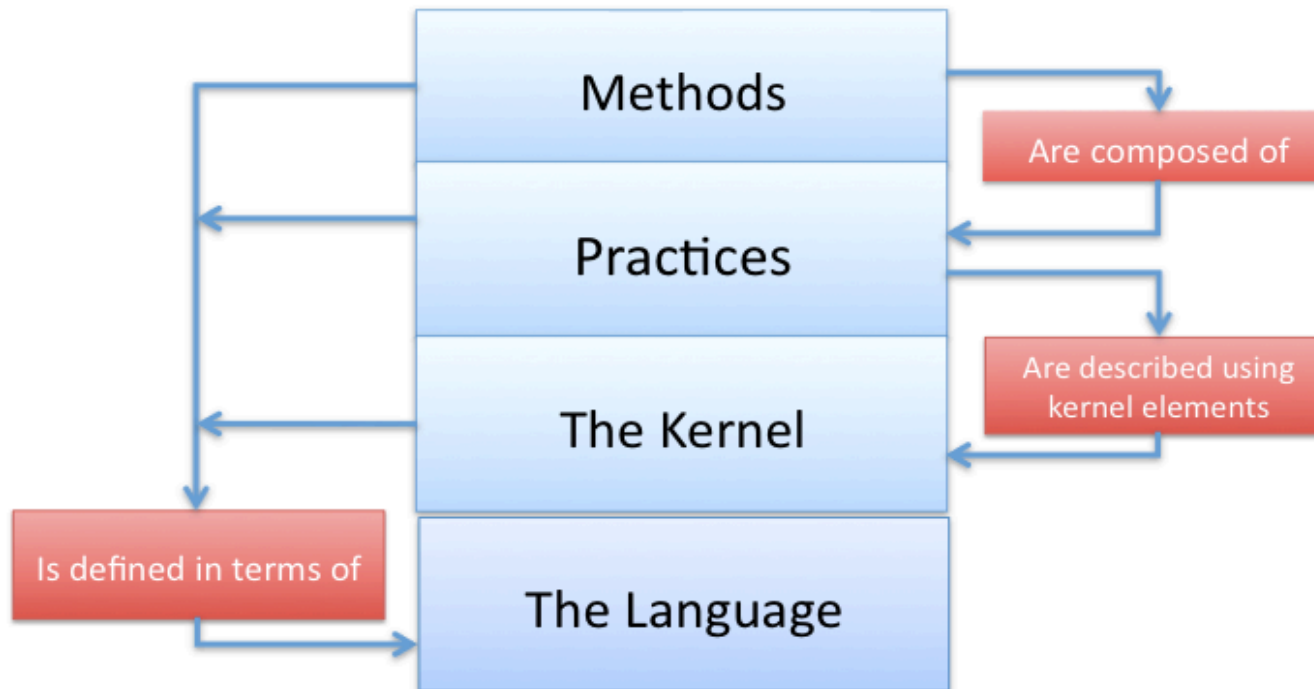
Hiroshi Miyazaki, Fujitsu

Tom Rutt, Fujitsu

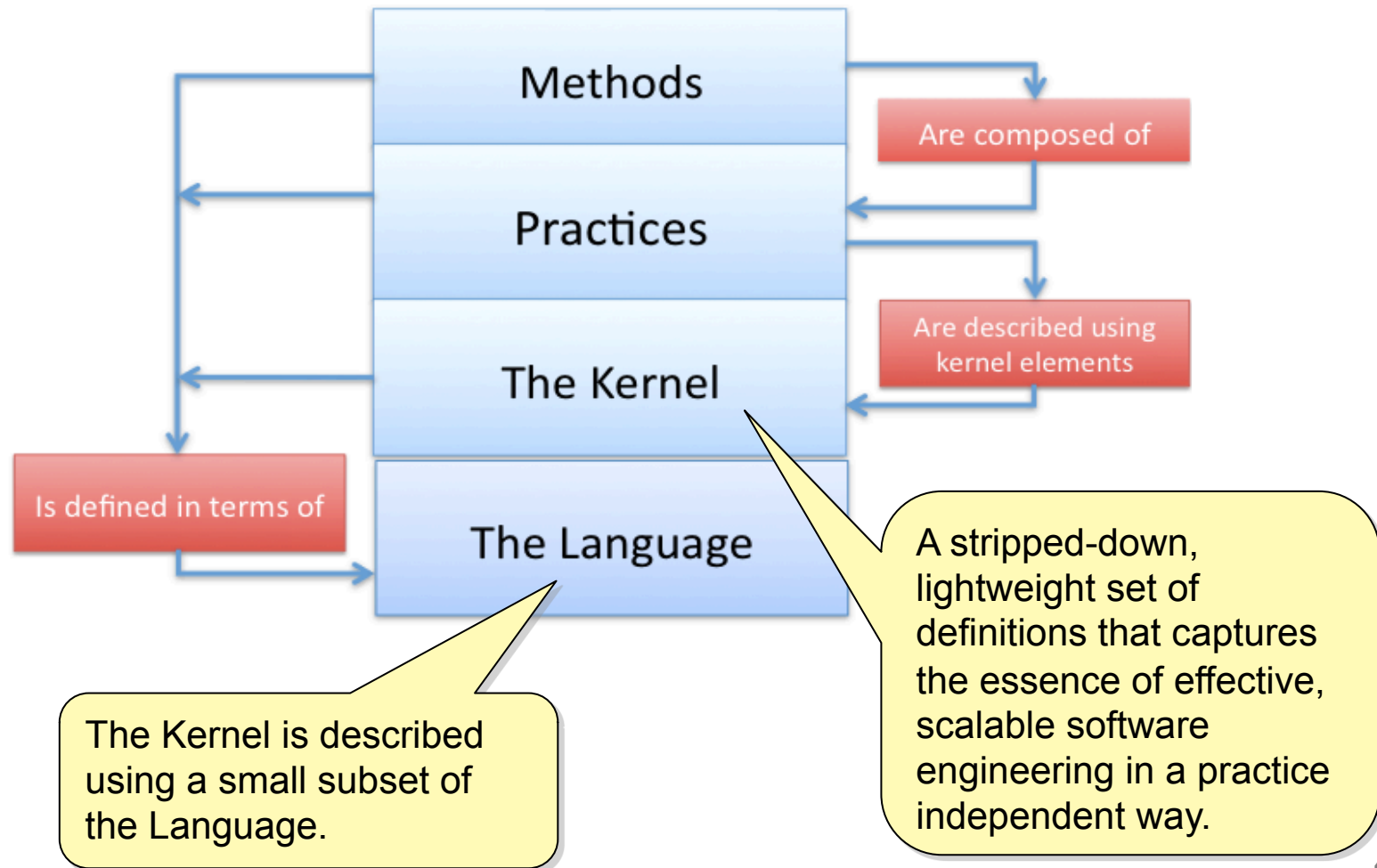Ed Seidewitz, Model Driven Solutions

www.semat.org

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

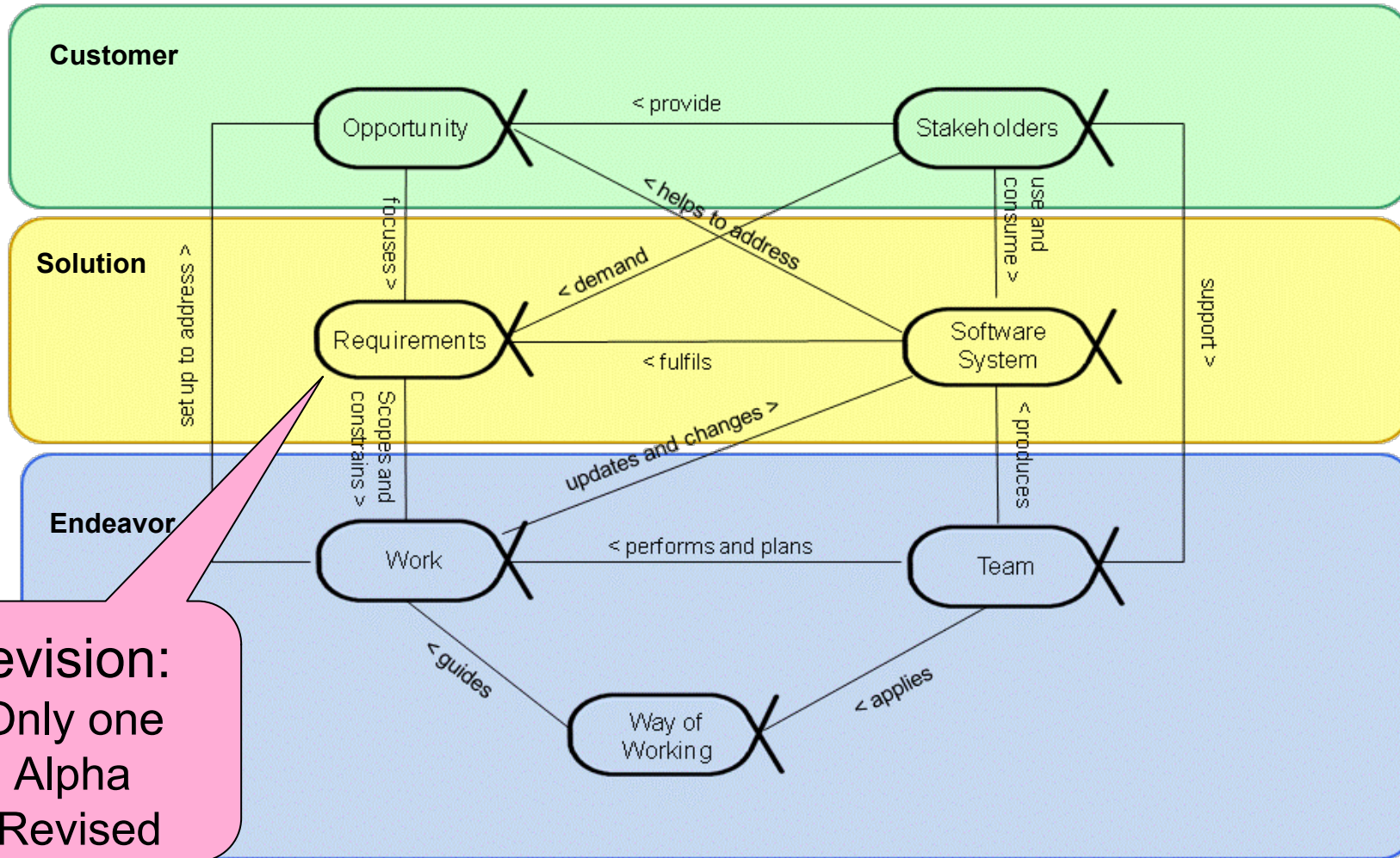- Recommendations

Introduction to Essence

Ivar Jacobson, IJI

The Kernel

Methods
Practices
The Kernel
The Language

Are composed of

Are described using kernel elements

Is defined in terms of

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
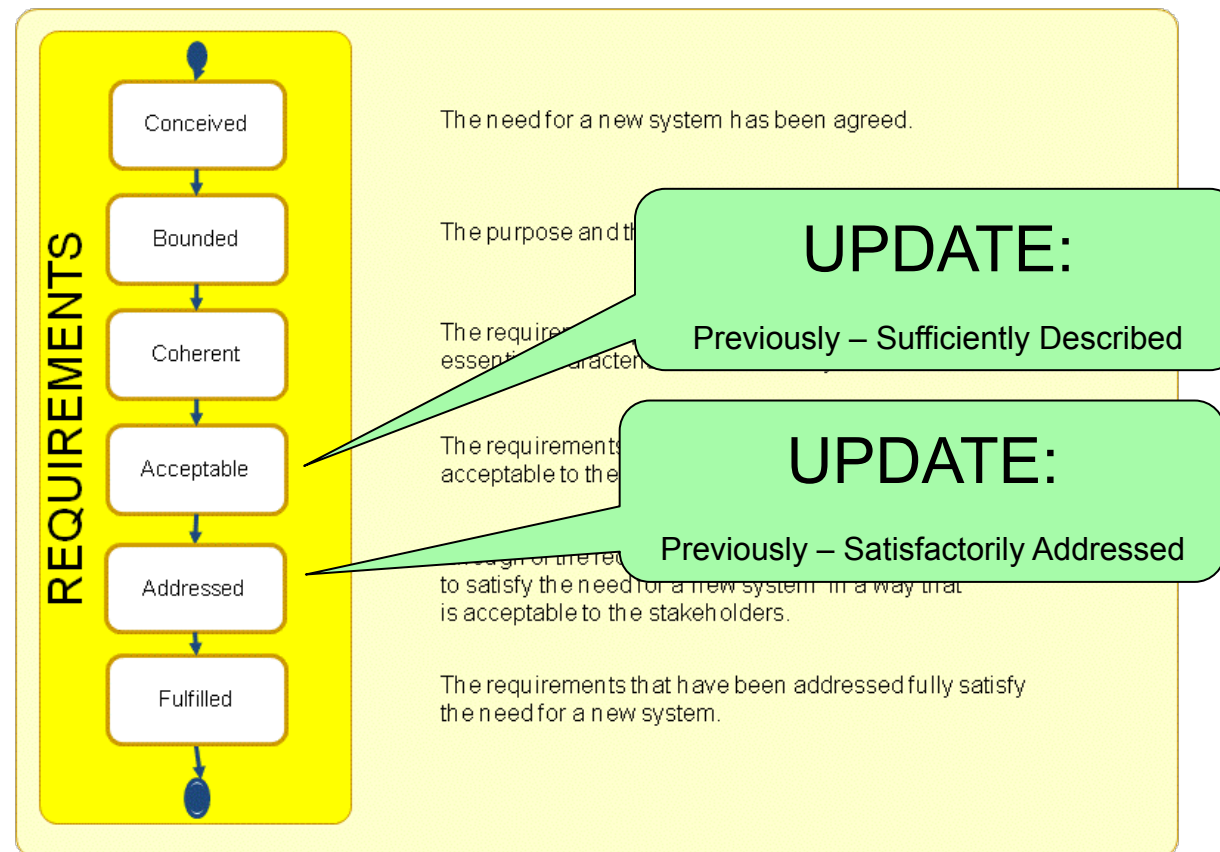
Alphas: Example

Requirements

Description

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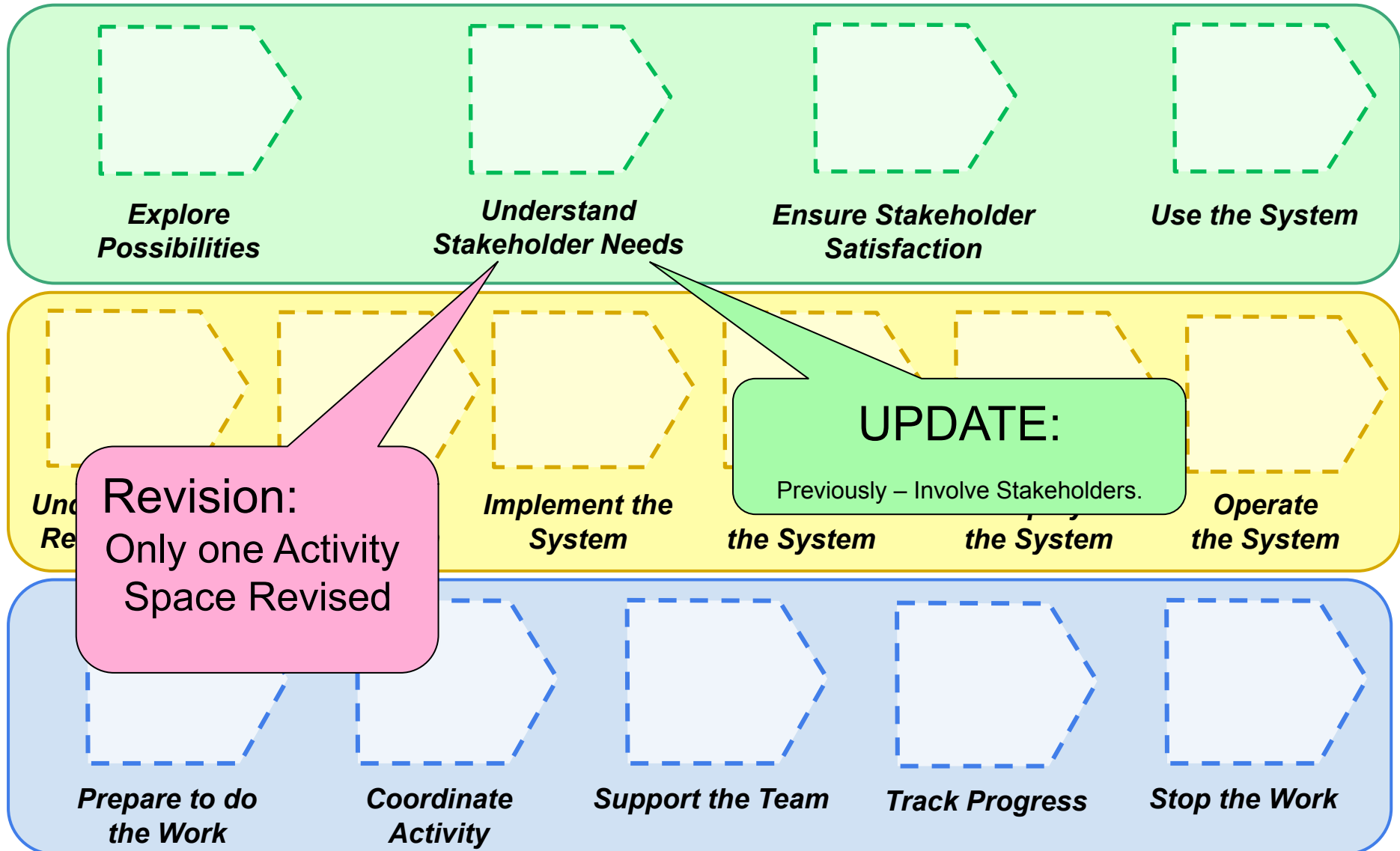
It is important to discover what is needed from the software system, share this understanding among the stakeholders and the team members, and use it to drive the development and testing of the new system.
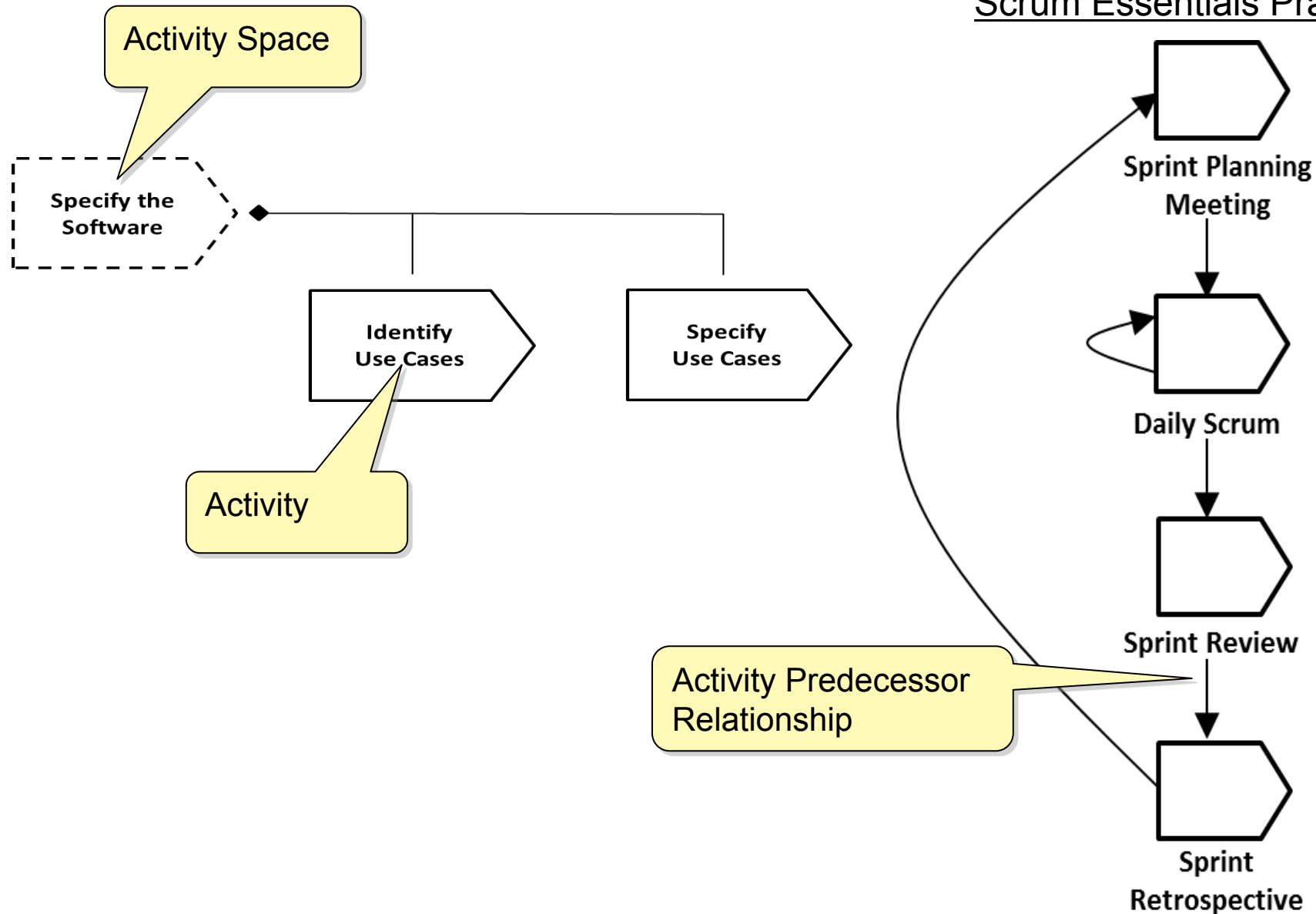
Associations

scopes and constrains : Work

Activity Spaces: The Essential Things to Do

Explore Possibilities

Understand Stakeholder Needs

Ensure Stakeholder Satisfaction

Use the System

Und... Re...

Implement the System

... the System

... the System

Operate the System

Revision: Only one Activity Space Revised

UPDATE:

Previously – Involve Stakeholders.

Prepare to do the Work

Coordinate Activity

Support the Team

Track Progress

Stop the Work

Activity Spaces: Examples

Activity Space

Specify the Software

Activity

Identify Use Cases

Specify Use Cases

Scrum Essentials Practice

Sprint Planning Meeting

Daily Scrum

Sprint Review

Activity Predecessor Relationship

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.
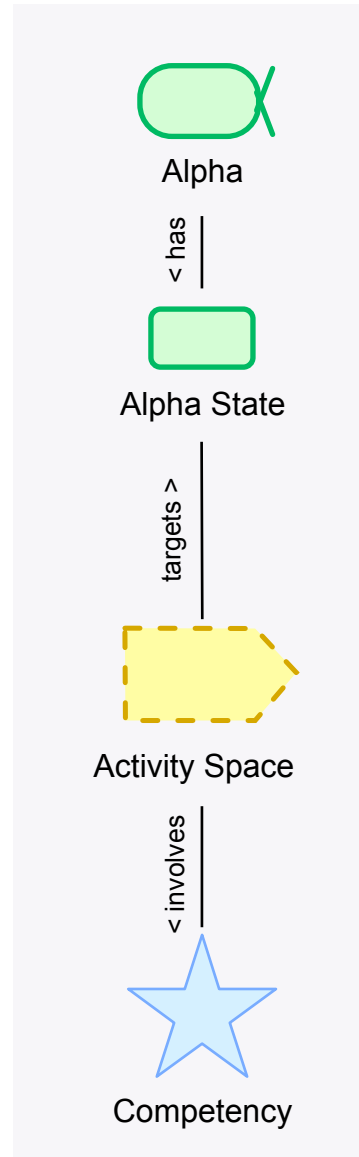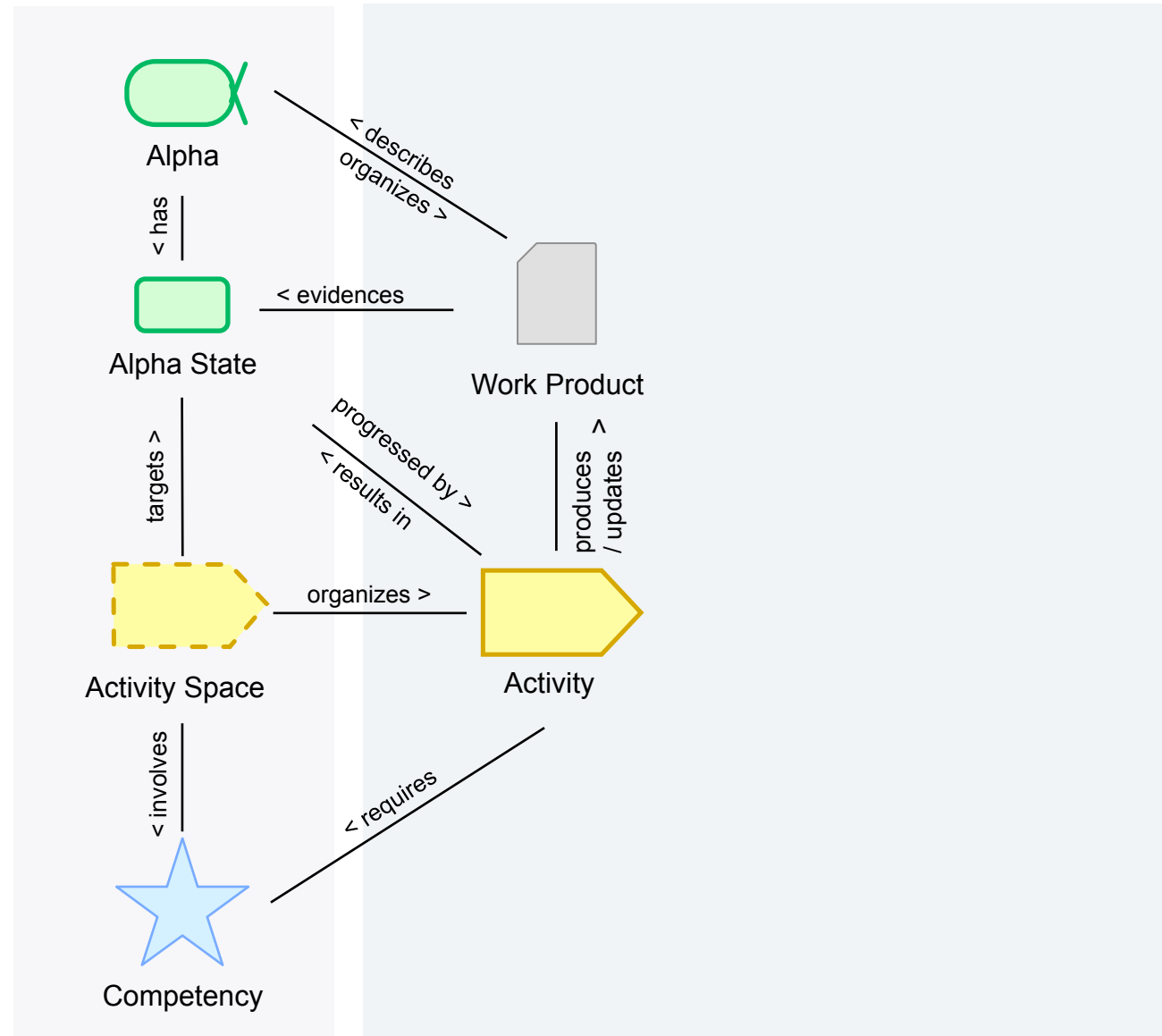 - intuitive and concrete graphical syntax, not formal semantics.
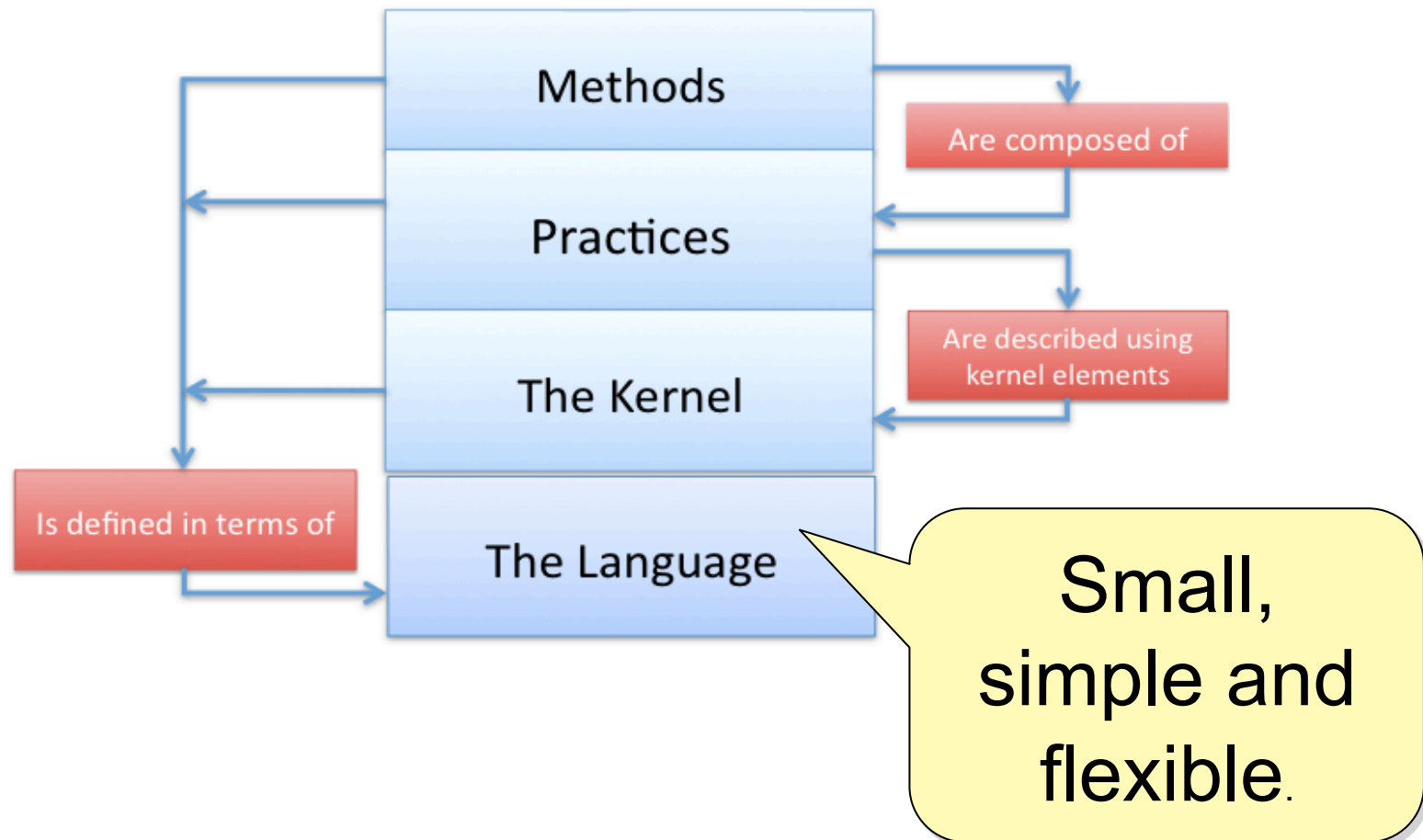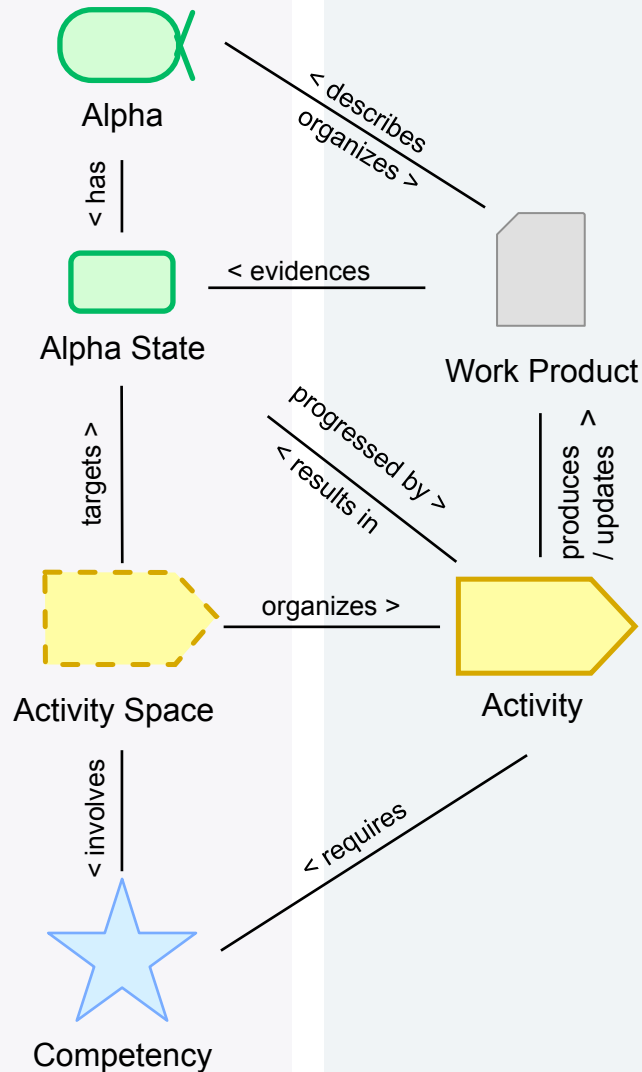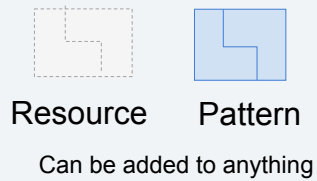
The Language: Small, Simple and Flexible
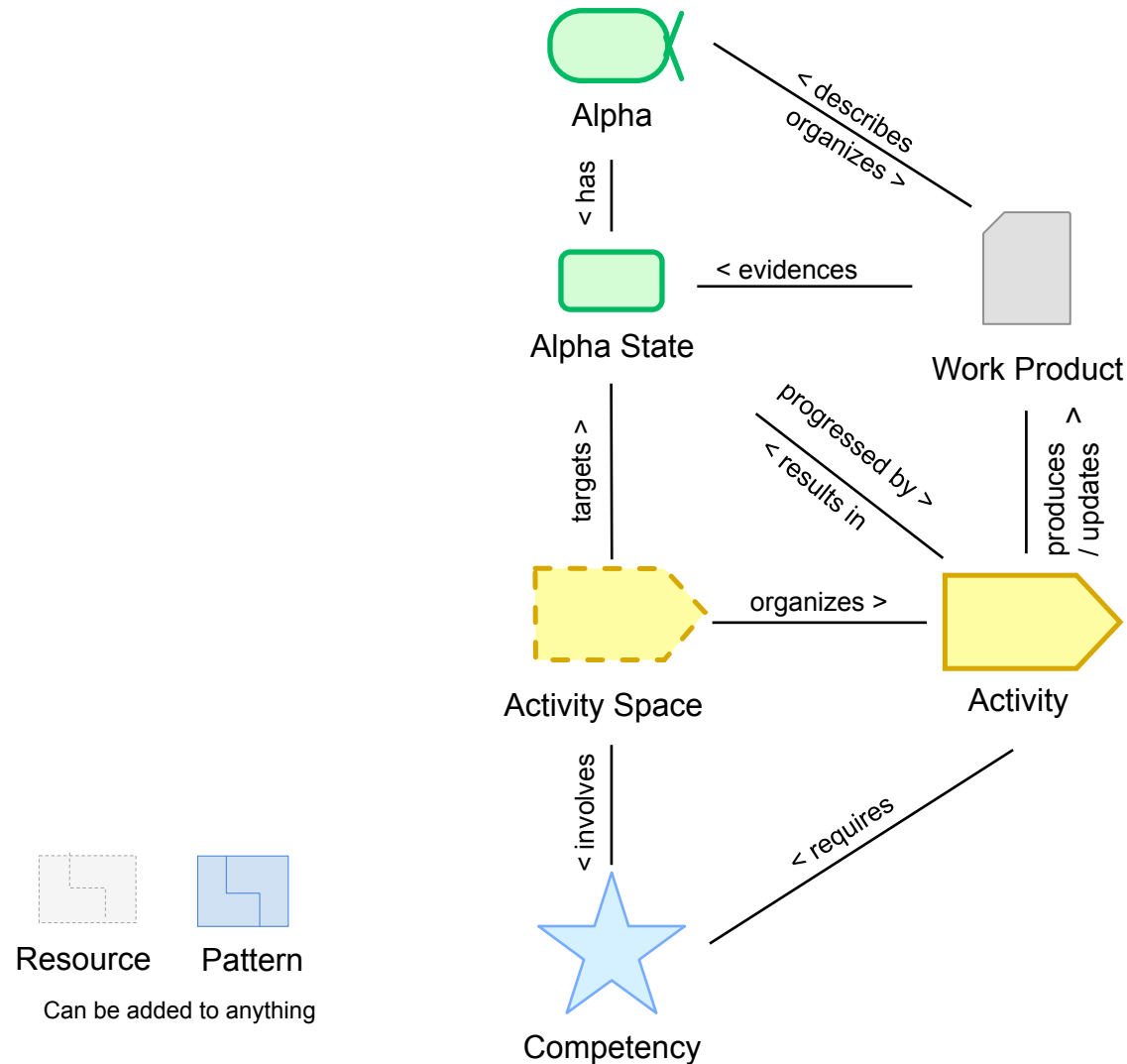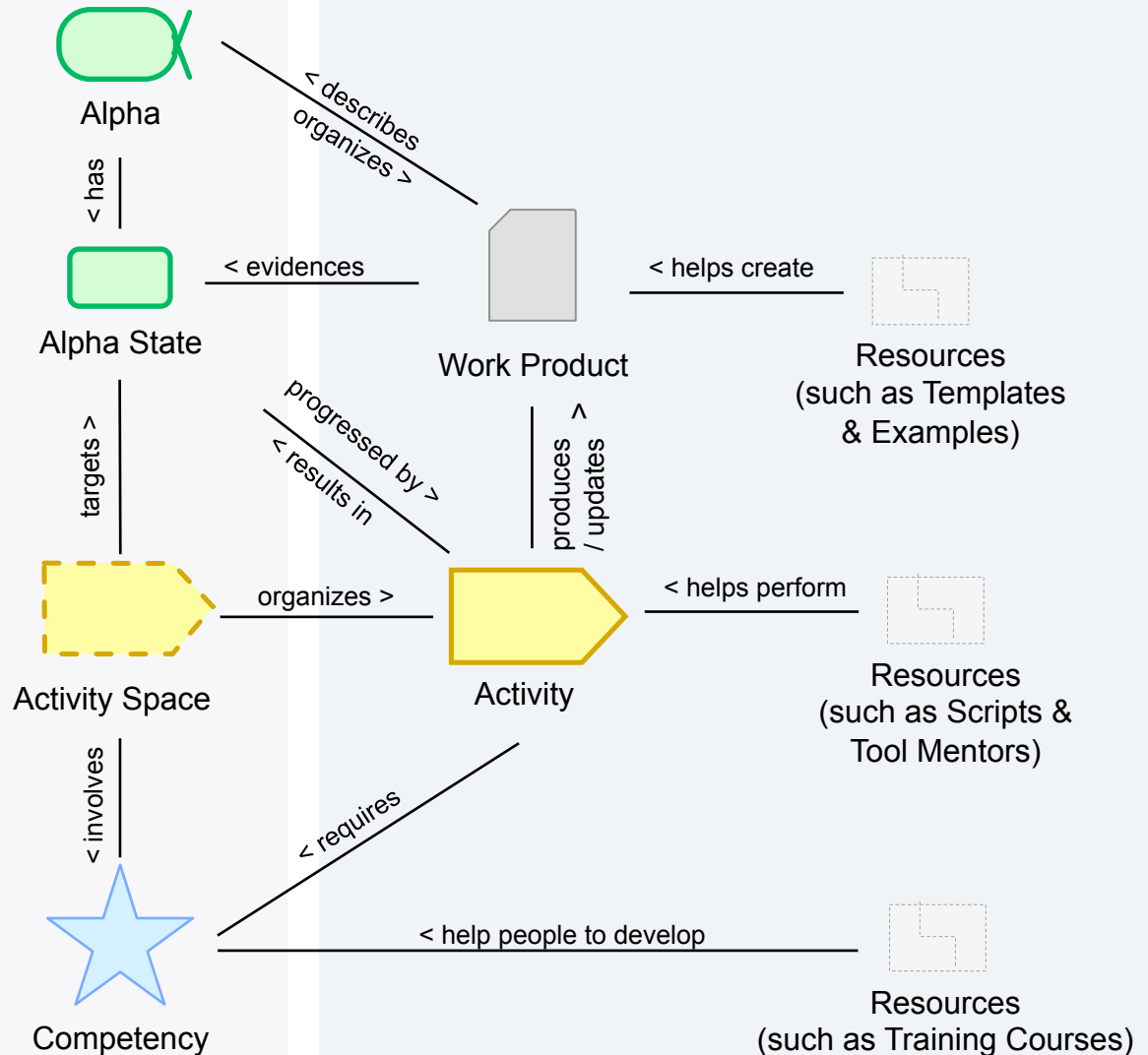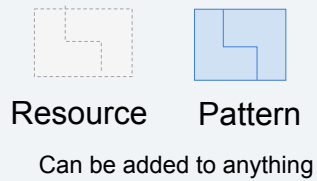
The Kernel provides the blue print

Practices add the detail

Resources and Patterns enable extension

Some example uses of resources

Some example uses of Patterns

Alpha

Alpha State

Work Product

Activity Space

Activity

Competency

e.g. Checkpoint or Milestone
Patterns that synchronize Alpha State progression

e.g. Phase
Patterns that sequence and filter activities

e.g. Different Approaches
Patterns that describe alternative ways to perform and activity. .

e.g. Team Role
Patterns that qualify team membership such as Team Roles and Team Structures.

Resource   Pattern
Can be added to anything

synchronizes progress of>
< describes
organizes >
< has
aligns a set of >
< evidences
ends with >
results in set of >
targets >
progressed by >
produces / updates >
< helps perform
sequences >
< involves
< requires
< qualify responsibilities
< helps up profile teams and team members

Language: Structure of the Metamodel
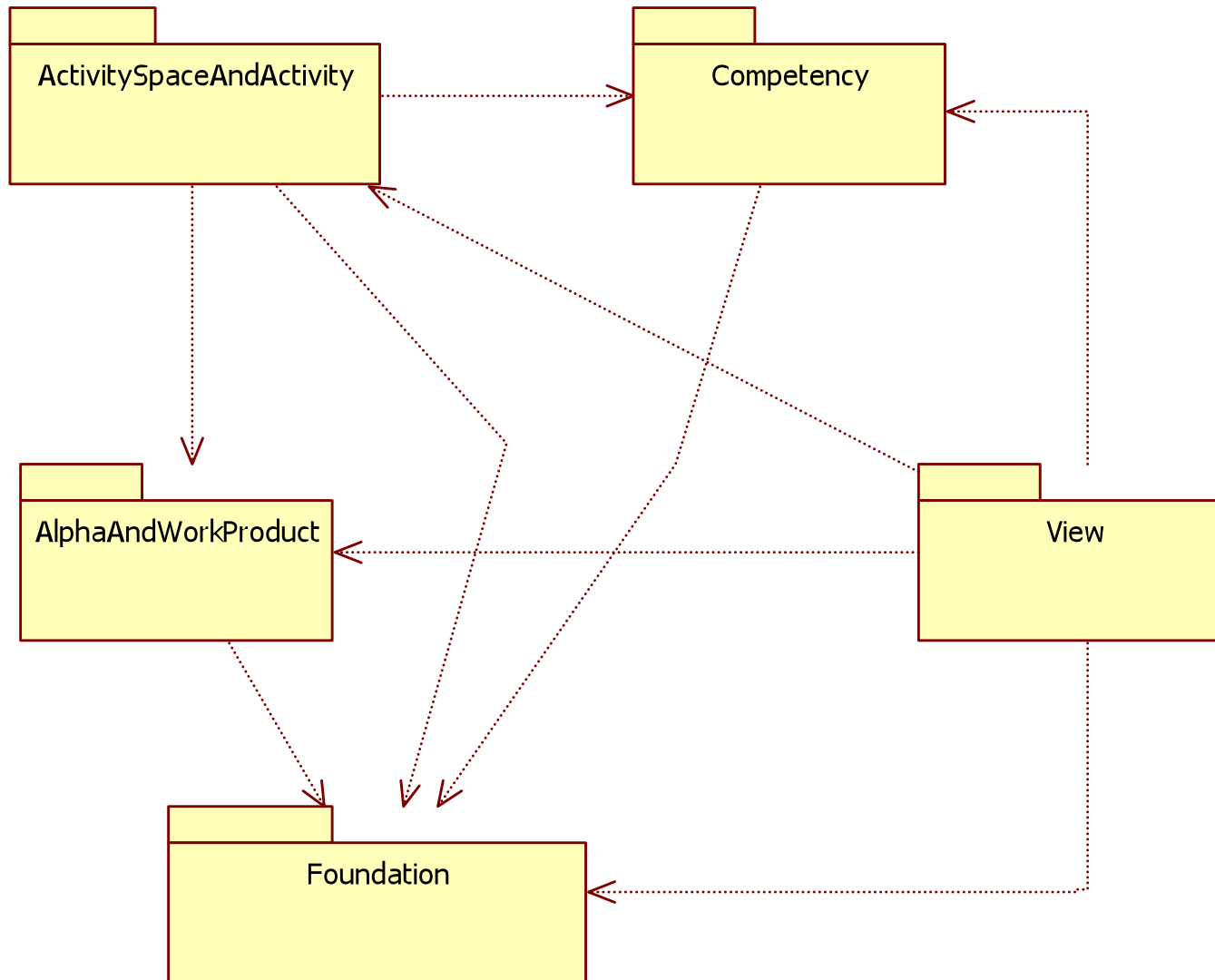
Language: Foundation
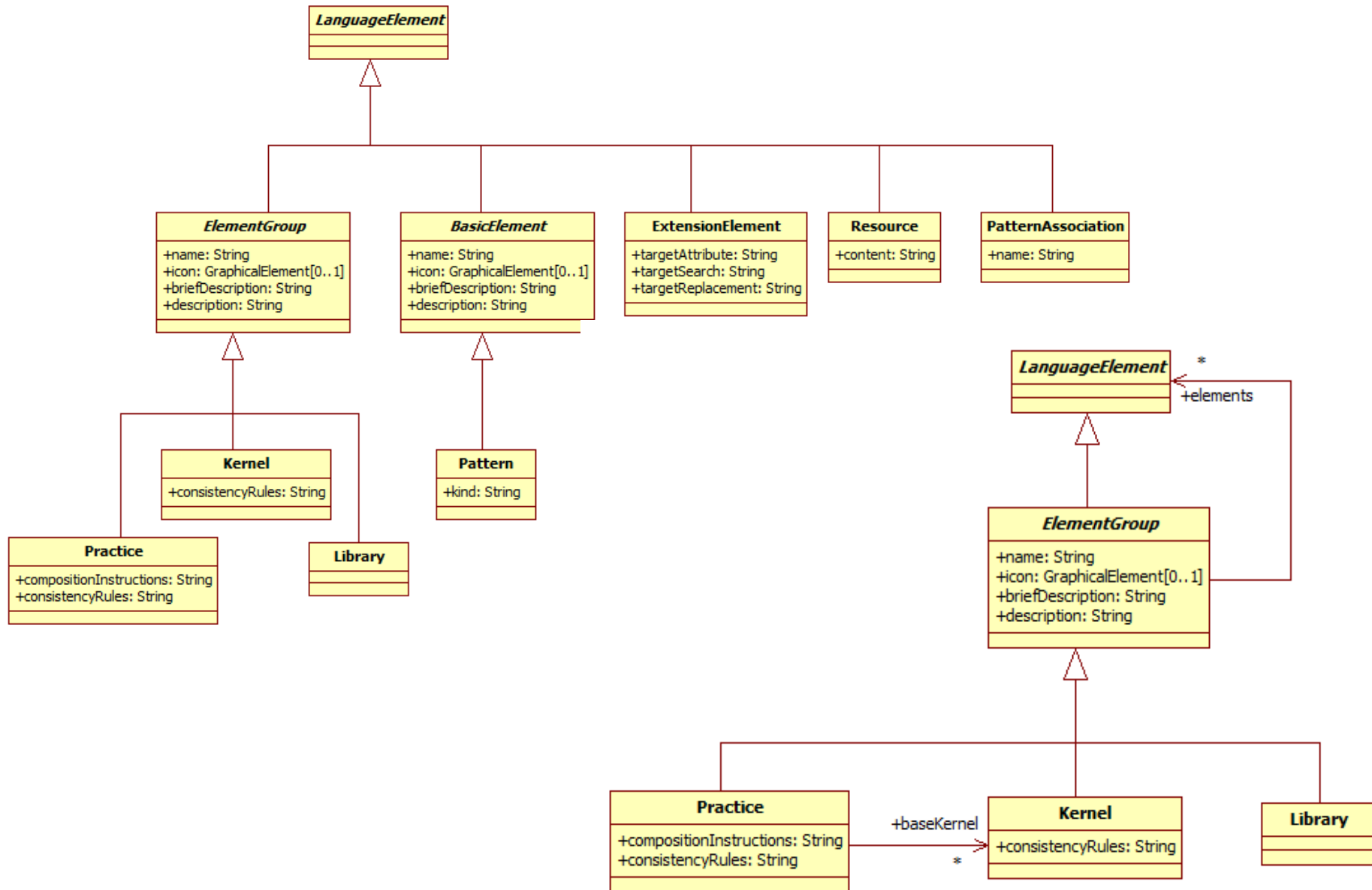
Language: Alpha and Work Product

Language: Activity Space and Activity

Language: Competency
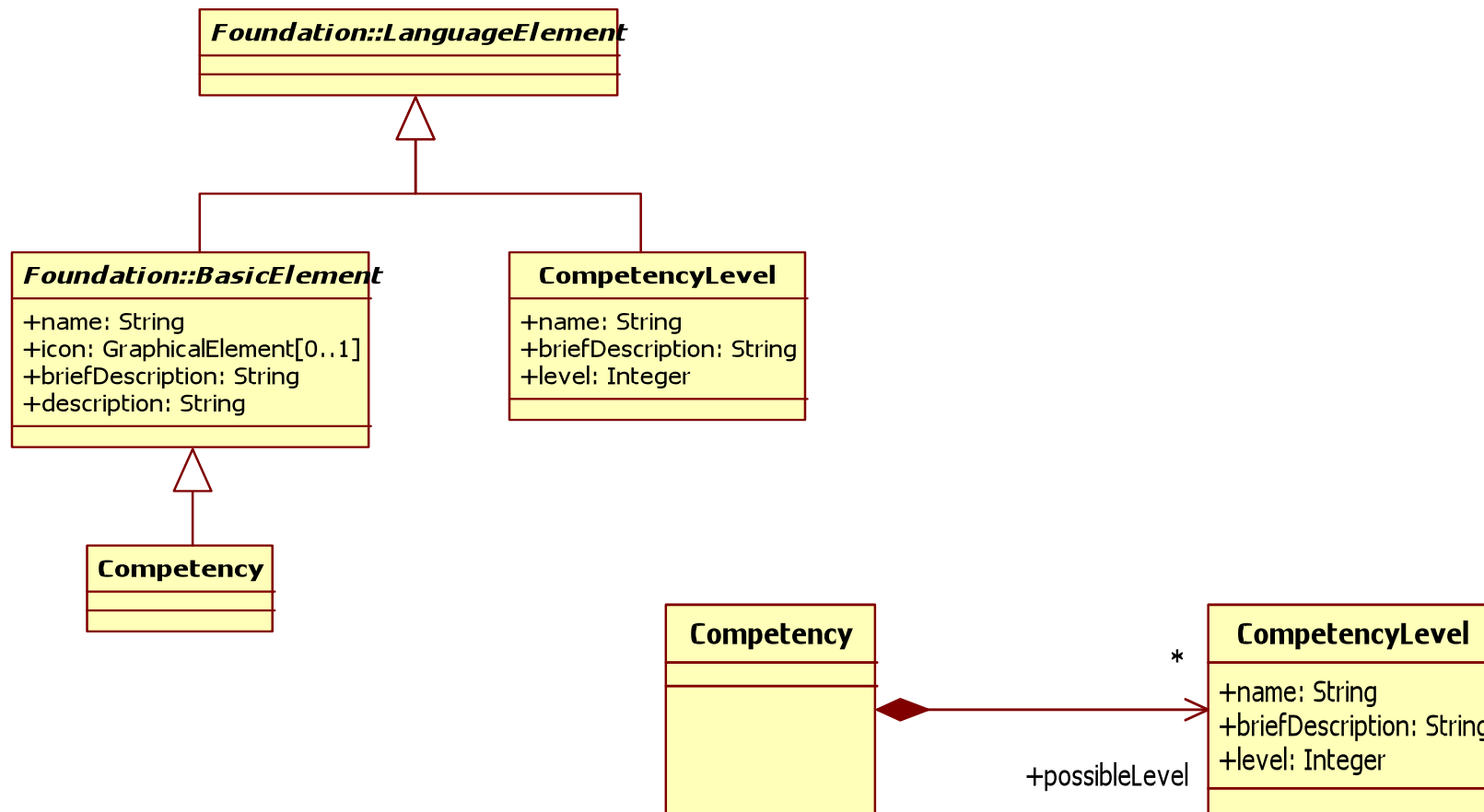
Language: View
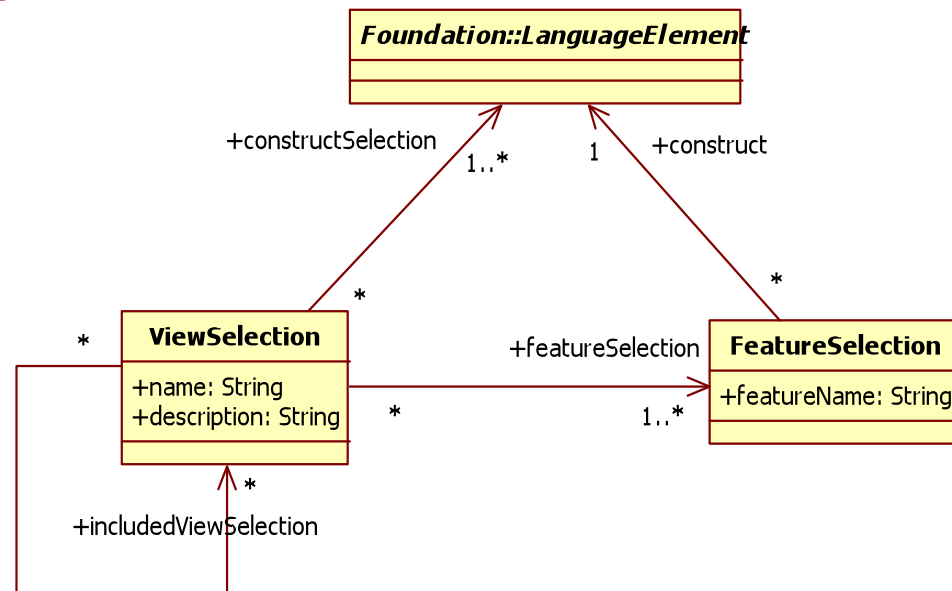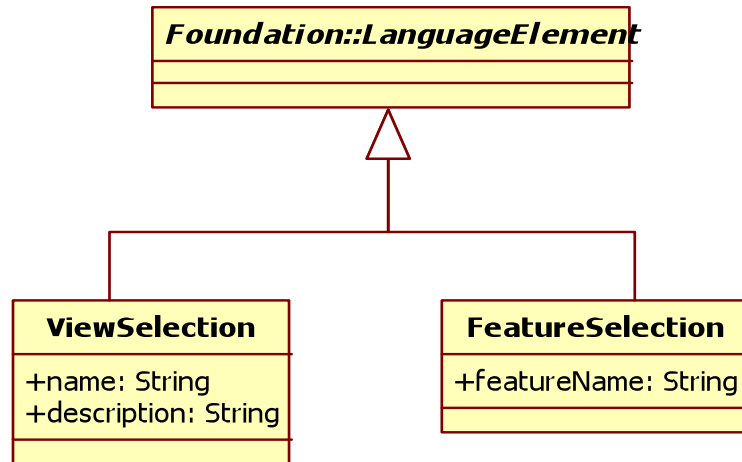
Ian Spence, IJI
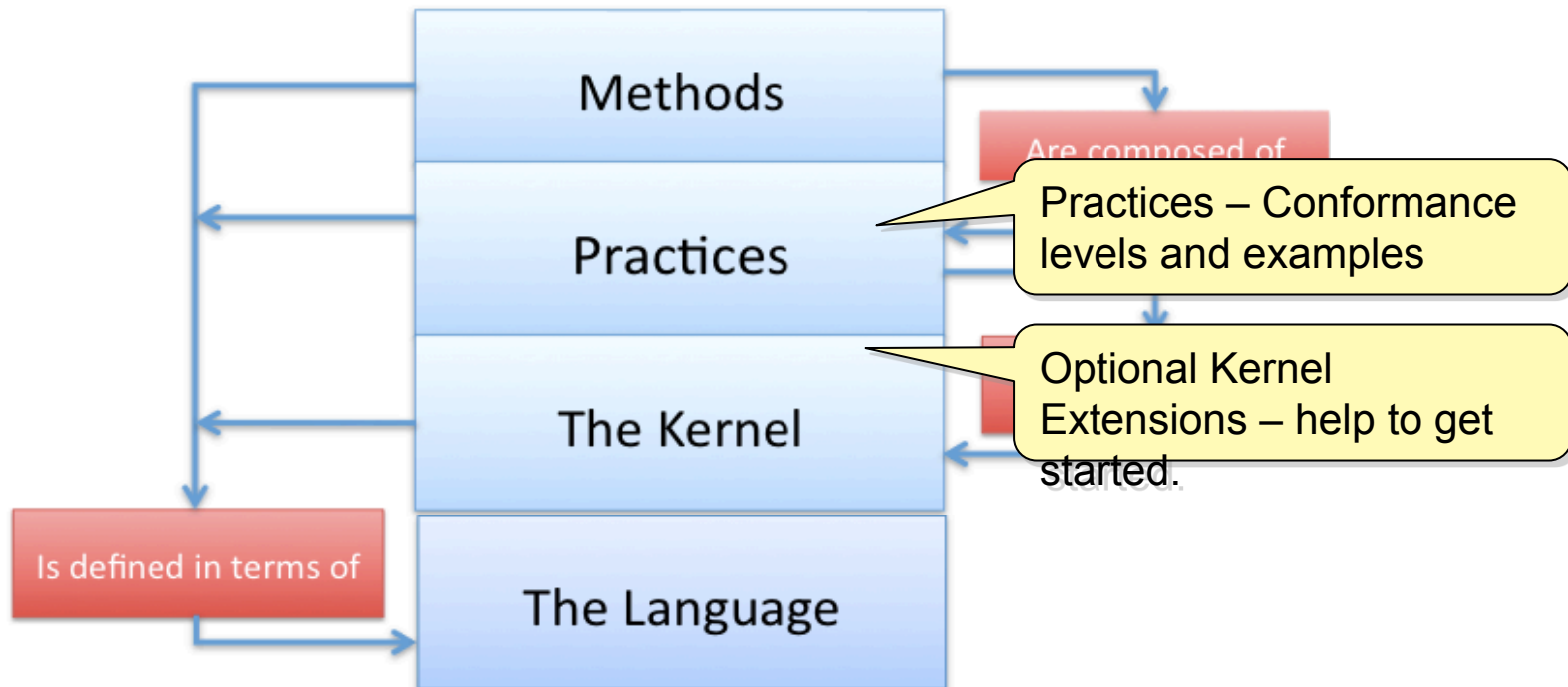
Optional Kernel Extensions

- **Business Analysis Extension** – adds two Alphas, Need and Stakeholder Representative, to drive forward the Opportunity and the Stakeholders.
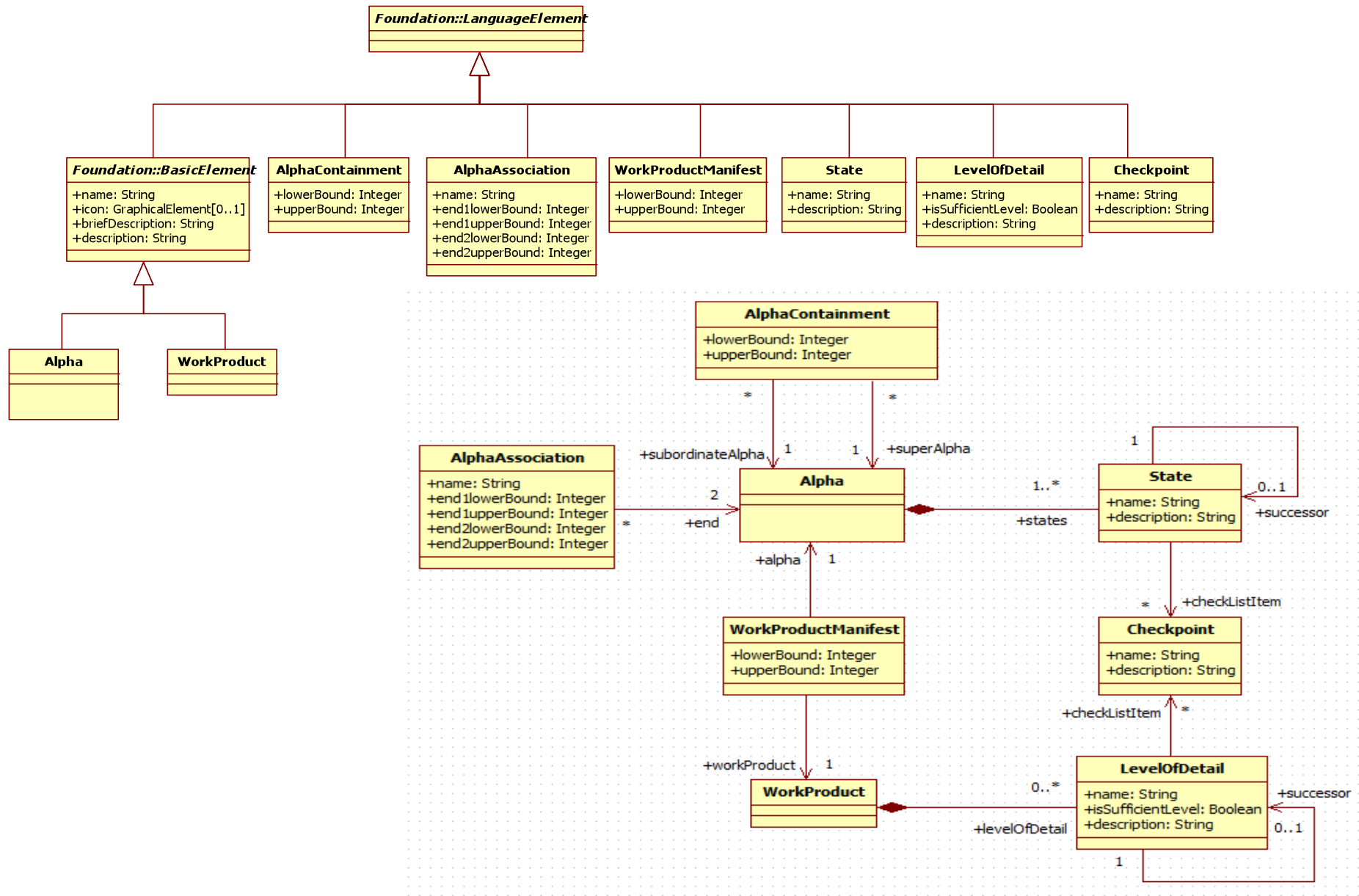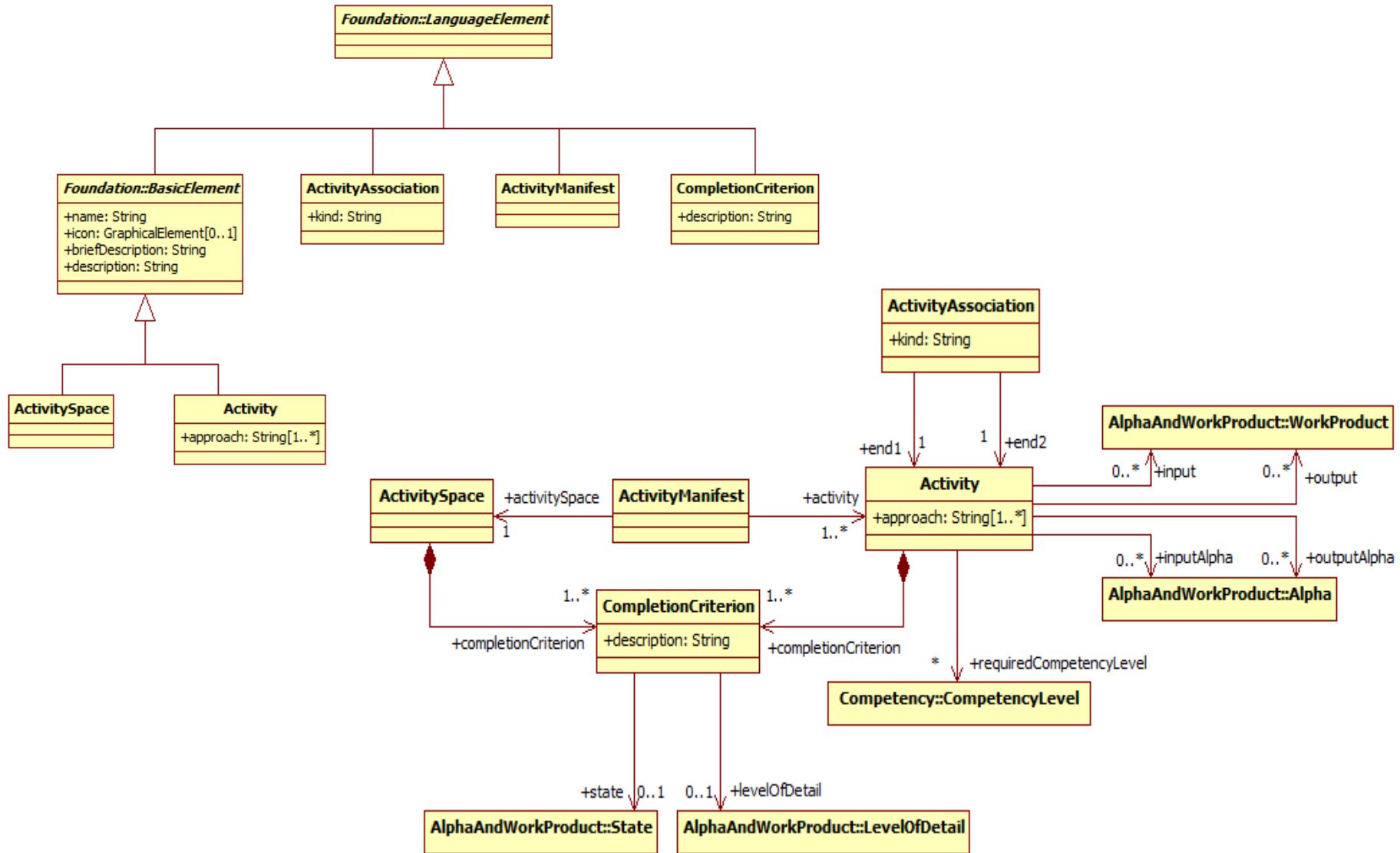
- **Development Extension** – adds two Alphas, Requirement Item and System Element to drive forward the Requirements and the Software System. As well as System Element it also adds Bug to monitor the health of the Software System. Bugs are an important thing to monitor, track and address in any software 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

Requirement Item

- **Identified** — A specific condition or capability that the Software System must address has been identified.

- **Described** — The Requirement Item is ready to be implemented.

- **Implemented** — The Requirement Item is implemented in the Software System and demonstrated to work.

- **Verified** — Successful implementation of the Requirement Item in the Software System has been confirmed.

Drives

Requirements

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

Development Extension (2)

Bug

Detected — An error, fault or flaw in the Software System is observed and logged.

Located — The cause of the Bug in the Software System has been found.

Fixed — The Bug has been removed from the Software System.

Closed — The removal of the Bug from the Software System has been confirmed.

Inhibits

Software System

Architecture Selected

Demonstrable

Usable

Ready

Operational

Retired

Development Extension (3)

System Element — Identified	A system element has been identified as part of the Software System and its responsibilities and its position in the Software System are clear.
System Element — Interfaces Agreed	The System Elements interfaces have been agreed.
System Element — Developed	The System Element has been implemented and tested, and is believed to be ready for integration into the Software System.
System Element — Ready	The System Element has been verified and is ready for live use as part of the Software System..

Drives

Software System: Architecture Selected → Demonstrable → Usable → Ready → Operational → Retired

# 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
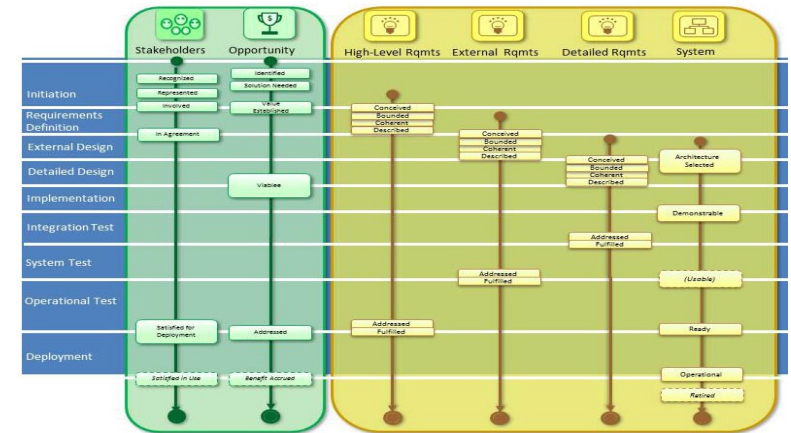
- With more to come…..
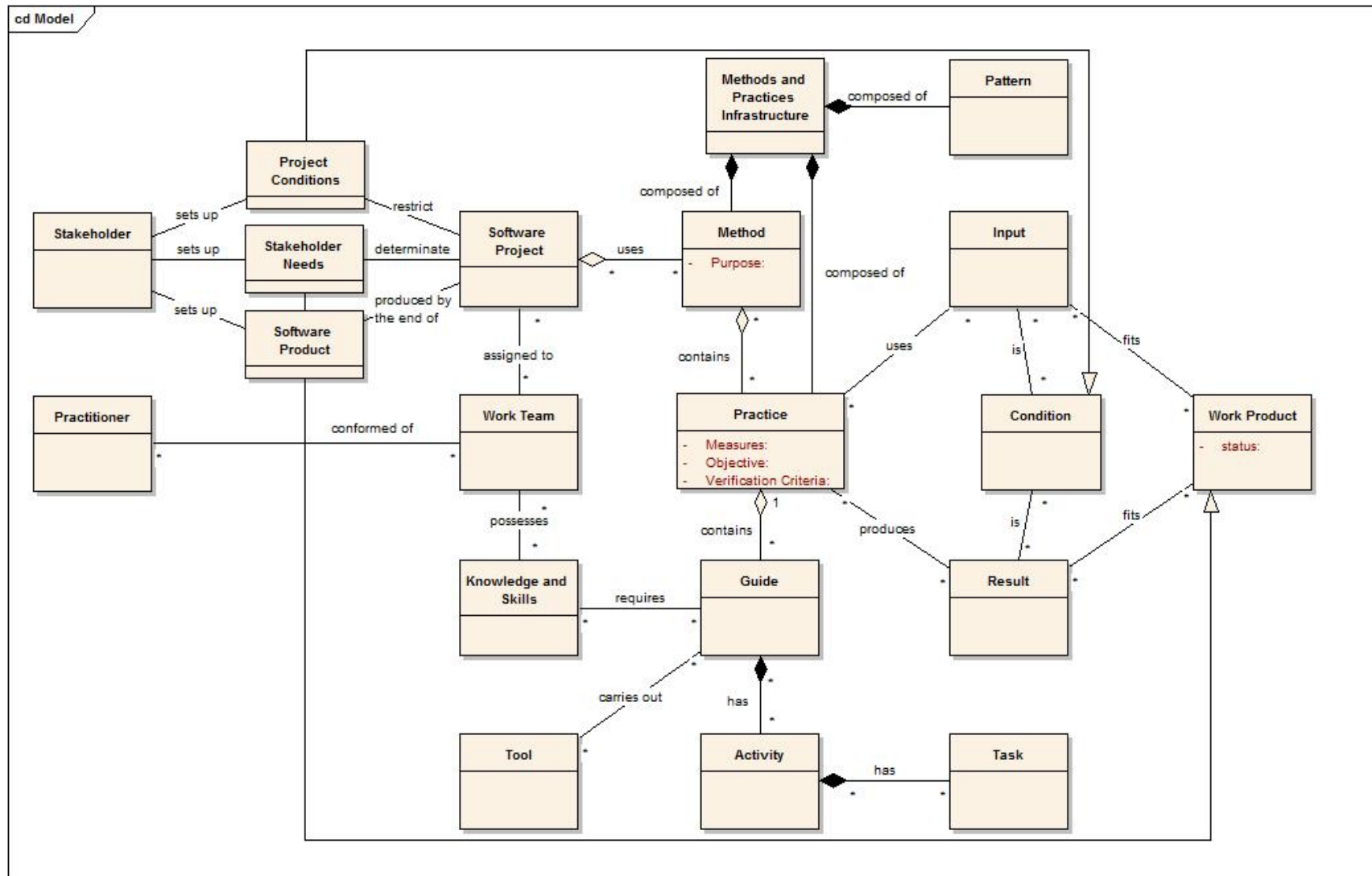 - Catalogue of short-form one-page descriptions in development
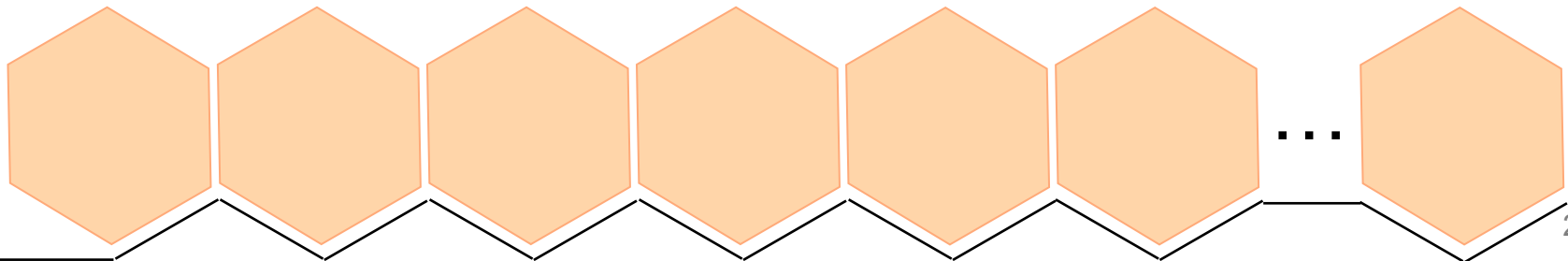
Alignment with KUALI-BEH

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

[Identifier]	Practice		
[name]			
Objective			
[objective]			
Source		**Result**	
[expected characteristics]		*[expected characteristics]*	
Verification Criteria			
[criterionA, criterionB,…]			
Instance States			
[stateA, stateB,…]			
Guide			
Activity	*[activity]*		
Input		**Output**	
Tasks (optional)	**Resource (optional)**	**Knowledge and Skills**	**Measures**
[toDoThis, …, toDoThat, …]	*[list of proposed tools]*	*[abilities, competences, attainments, …]*	*[measureA, measureB, …]*

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
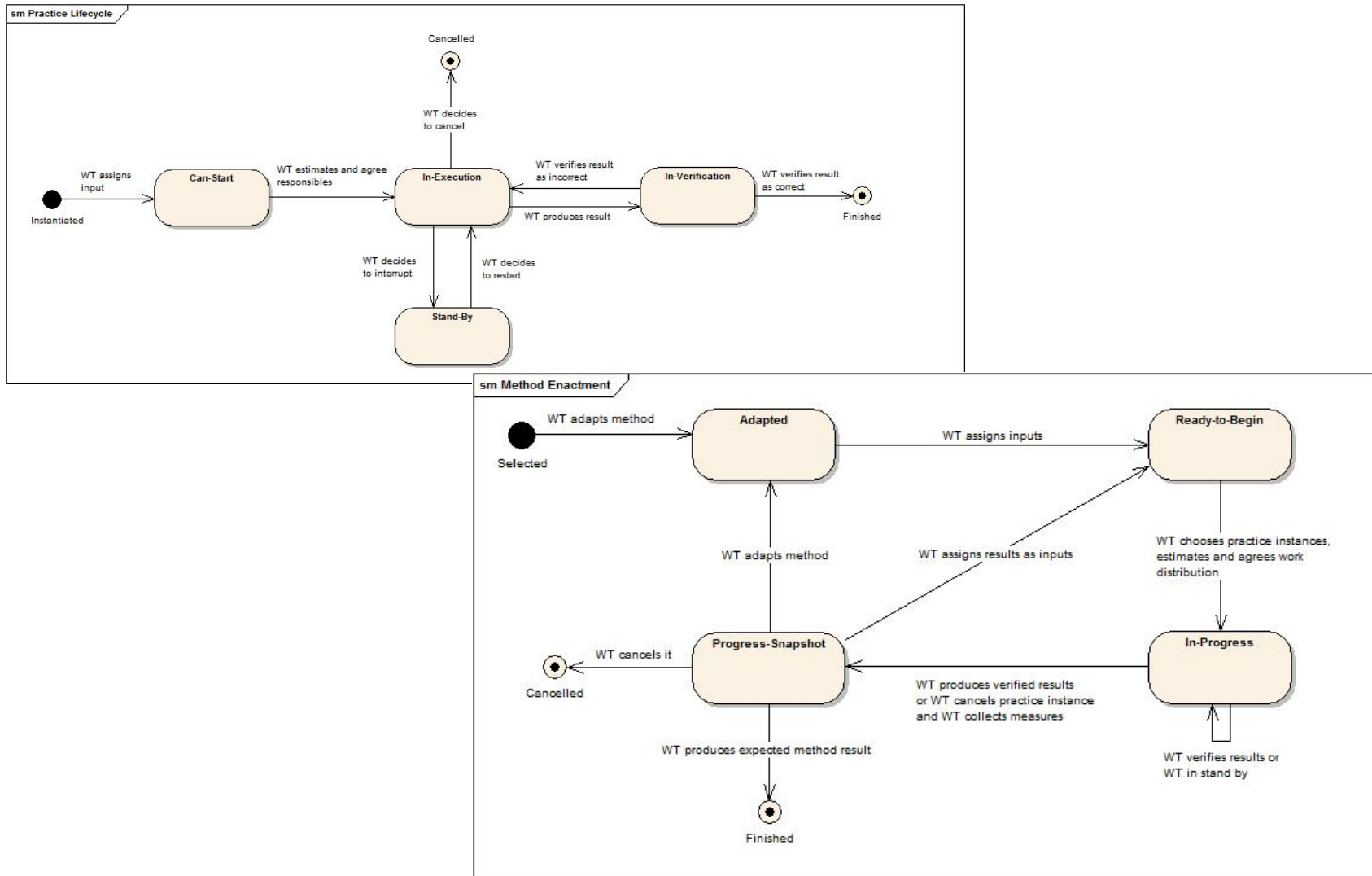
Alignment with KUALI-BEH

- 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

Alignment with SEMDM (ISO 24744)

Brian Elvesæter, SINTEF

- 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

Level 2 — WorkProduct

Level 1 — my_WorkProduct ◁— Use Case narrative

Level 0 — Withdraw Cash

Defined in the Essence language
Defined by a user of the Essence language
Instantiation (lower box is an instance of the upper box)
Sub-classing (lower box is a sub-class of the upper box)
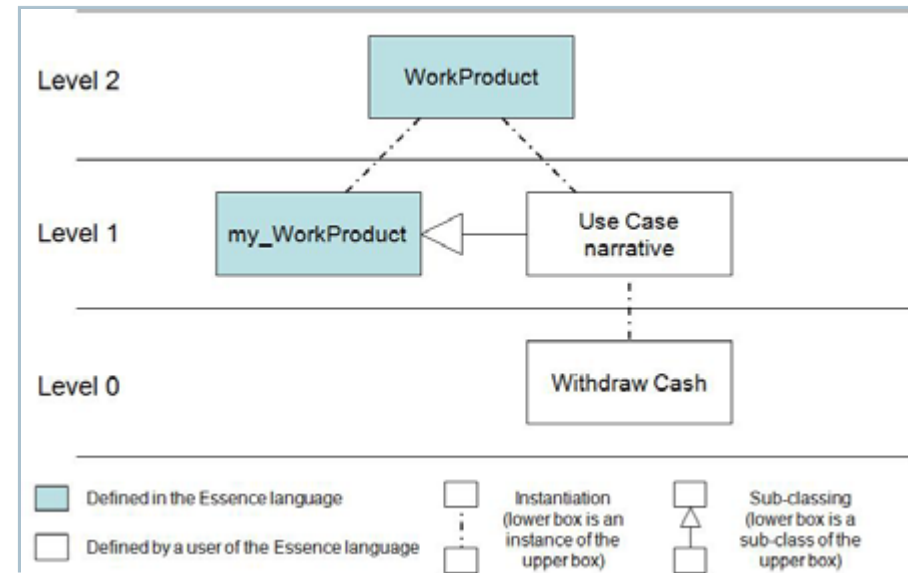
- **ISO 24744 and Essence expresses the same, but in two different ways.**
- **Essence is MOF-based, ISO 24744 is not!**

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
  - WorkProduct**Kind** 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

Book is available now – Safaribooksonline/Addison Wesley