

DOMAIN DEFINITION FOR THE KERNEL LANGUAGE

Version 1.3

The objective of the Kernel Language is to allow people to describe their current and future practices and methods so that they can be composed, compared, evaluated, and measured by developers as well as taught and researched by academic and research communities. The principal intended users of the Kernel Language are projects looking for appropriate methods and practices to apply them effectively.

A *method* is composed of practices.

There are two kinds of *practices*:

1) A *peer practice* is a separate concern of software development. Examples are “development from requirements to test”, “iterative development”, or “component-based development”.

- A peer practice, unless explicitly defined as a continuous activity, has a clear beginning and an end.
- A peer practice includes criteria for its assessment and suitable metrics. Where applicable, the assessment criteria must include quantitative elements.

2) A *cross-cutting practice* denotes a general mode of operation that applies across practices. For example, many practices involve organizing workshops, or relying on self-organizing teams. By describing such concepts as cross-cutting practices, we avoid repeating their details in the description of peer practices.

Enactment is the creation of a set of activities from method and practice elements for the purpose of building a system together with a set of completion criteria (gates) for each task. Activities require certain skills and resources that have those skills¹. Activities may be required to precede other activities and may be associated with work products. Activities may be applied repeatedly to different parts of the system, requiring the specification of precedence between those tasks and the allocation of resources to them.

Validation is the comparison of the set of gates completed for a task with those defined by the enactment of the task. If the two differ, the description of the enactment should change to capture what is actually being done, or the activity should change to match what is supposed to be done.

The kernel language and the resulting descriptions must support:

- all relevant practices and their composition in *today's* methods.
- *extensibility*, allowing the description of yet-to-be-invented methods and practices.
- *composition*, in different ways, including merging and extension to describe new methods. It also must support the definition of the universals in the kernel.
- *comparing* methods and their practices.
- *enactment and validation*

The kernel language shall have an abstract syntax, static and operational semantics. It shall have at least two concrete syntaxes, a graphical form and optionally a textual.

¹ Capabilities may be factored out.