Essence Explained

Essence is a standard for the creation, use and improvement of software engineering Practices which is maintained and published by the OMG international open standards consortium.

Key Concepts

Essence describes a language and a kernel:

- The Essence Language enables practices and related knowledge to be expressed in a simple, visual way that ensures that they can be easily shared, understood, adopted, adapted and applied both independently and in combination with other Essence Practices.
- The Essence Kernel provides the common ground for defining software development Practices. It includes the essential elements that are always central to every software engineering endeavor. The Kernel helps practice authors to define good Practices and helps practitioners to make informed decisions about which Practices to adopt and how to apply and adapt them.

Kernel

A Kernel is a kind of domain model. It defines important concepts that are general to everyone working in a domain, like software engineering or product development.

A Kernel is a skeletal practice framework, within which many different Practices can be used successfully together. You can think of a Kernel as being like an electronic circuit board, with Practices being like the electronic components that plug into the circuit board.

Kernels can be extended for use in wider domains or related specialist domains, such as Systems Engineering and new kernels can be defined for other domains, such as Business Change.

The Essence Software Engineering Kernel

The Essence Software Engineering Kernel provides the common ground for defining software development practices. It includes the essential elements that are always central to every software engineering endeavor, including common:
- Activity Spaces
- Alphas
- Competencies
- Competency Levels.

Essence Kernel Alphas

The Essence Kernel defines seven core, common Alphas for software engineering, which together:

Capture the key concepts involved in software engineering
Allow the progress and health of any software endeavor to be tracked and assessed
Provide common ground for the definition of software engineering practices.

Note that the Kernel Alphas each belong in one of the three Areas of Concern and are color-coded accordingly.

Essence Kernel Activity Spaces

As with the Kernel Alphas, these can be used both independently and to help define practices:

The set of Activity Spaces can be used to assess coverage and do gap analysis across a team's current way of working (do we have an agreed approach to doing these things?)
Activities in practices can be placed within Kernel Activity Spaces to give a clear indication of what general kinds of things the practice and its activities will help the team to achieve – e.g. a Find User Stories Activity is located within the Understand the Requirements Activity Space.

Essence Kernel Competencies

The Essence Kernel defines six core Competencies that are the commonly needed for any software endeavor.

Note that, as with other aspects of the Essence Kernel, the set of Competencies described by the Kernel is the core, minimum set that is generally always expected to be required on any software development endeavor. Specialist Practices can extend this core minimum set by introducing their own additional Competencies as needed for particular types of endeavor to address specific types of challenge, such as Operations, Coaching, Database Administration or Hardware Design for example.

Essence Kernel Competency Levels

The Essence Kernel defines five standard Competency Levels that can be used across all Competencies as follows:

Assists - demonstrates a basic understanding of the concepts involved and can follow instructions
Applies - able to apply the concepts in simple contexts by routinely applying the experience gained so far
Masters - able to apply the concepts in most contexts and has the experience to work without supervision
Adapts - able to apply judgment on when and how to apply the concepts to more complex contexts. Can make it possible for others to apply the concepts
Innovates - a recognized expert, able to extend the concepts to new contexts and inspire others.

Practice

A Practice is a repeatable approach to doing something with a specific objective in mind. Examples might include Scrum or User Stories.

A good practice:

Helps a team to do something well
Can either be used independently or in combination with other Essence practices

Is of manageable size (perhaps 5 to 15 elements being the normal limits for this)
“Tells a story” of how we achieve some valuable outcomes, i.e. (typically) has: one or more Activities, that progress...one or more Alphas, that are captured and communicated using ... one of more Work Products, the whole being supplemented/joined-up with ...one or more Patterns, and more information is sign-posted, or sources cited, using ...references to one or more Resources.

Area of Concern

Essence subdivides the territory of software engineering into three broad areas that any software endeavor has to pay attention to. Each has its own color-coding.
Areas of concern:

Customer – contains everything to do with the use and exploitation of the solution to generate value for the customers and users

Solution – contains everything to do with the specification and development of the solution to meet the needs of the customers

Endeavor – Contains everything to do with the team and the way that they approach their work to deliver the solution to the customer.

Activity

An Activity is the doing of some work by one or more people, for example in a workshop or meeting, or as an individual, pair or larger group collaboration. Examples might include Daily Stand-Up Meeting, Backlog Refinement or Develop a Component.

Activities are what we do. Activities are important because, unless we collectively actually do something (successfully), nothing is ever achieved or produced.

A good Activity description will include guidance on:
- What outcomes the Activity produces or achieves
- What we should do to achieve these outcomes
- What kinds of Competencies at what Levels are needed to undertake the activity successfully.

Activity Space

An Activity Space is a placeholder for something to be done in an endeavor, such as to Understand the Requirements.

The Essence Kernel defines a number of Activity Spaces that together represent the kinds of things that we need to do to progress any software engineering endeavor.

Activity Spaces can be used to group together related Activities that different Practices define.

Alpha

An Alpha is a key aspect or element that we need to progress, the State of which is a key indicator of the overall progress and health of an endeavor. Examples might include a Team achieving a State of Performing or a User Story being progressed to a Done State.

Alphas are what we progress. These are of central importance in Essence because they ensure we remain focused on the valuable outcomes we are trying to achieve, not on secondary concerns such as what physical documents or other artefacts may or may not help us to achieve these outcomes.

A good Activity should be defined is by what progress it achieves. For example, a Backlog Refinement Activity might produce a new User Story and progress others so they are Ready for Development.

Alpha State

Alphas have States that describe how far they have been progressed. A State is a specification of the state of progress of an Alpha. Examples might include a User Story Alpha being in a State of Done or an Impediment being in a Resolved State.

We can more accurately define Activities in terms of what State they progress Alphas to (and possibly from). For example a Develop User Story Activity might progress a User Story from State Ready to State Done. An Alpha is defined in terms of the set of States that it is progressed through, "from top-to-bottom", for example as shown aside for a User Story Alpha.

Alpha States can in turn be concisely and accurately defined in terms of the set Checklist Items that need to be achieved for the State to be achieved. For example, the Done State of a User Story Alpha might have Checklist Items such as:
- Implemented in the product
- Tested against the acceptance criteria
- Accepted as meeting the stakeholder needs.

Work Product

A Work Product is an artifact of value and relevance for a software engineering endeavor. A Work Product may be a document or a piece of software, or any physical artefact such as a planning board, progress chart, screen map, paper prototype or design diagram. Examples might include a Project Plan, a Kanban Board or a Story Card.

For important things that we will invest significant effort in progressing, it may be helpful and wise to have something physical and visible to capture and communicate more widely what we have discussed and agreed about them. For example, User Stories are often described using physical index cards.

Note that if a Practice does not include a Work Product to describe an Alpha this does not mean it is forbidden to physically capture any information about the Alpha. It just means there is no explicit guidance being provided by the practice on whether or how you should do this.

Work Product Level of Detail

A Level of Detail is the range of content in a Work Product. Examples include a Story Card having its Value Expressed or additionally having its Acceptance Criteria Listed. Just as Alphas have States which describe how far they have been progressed, so Work Products have Levels of Detail which describe how much information of what kind has been captured.

A Work Product is primarily defined in terms of the set of Levels of Detail that it can progressed through, “from top-to-bottom”, for example as shown here for a Story Card Work product. Note that some Levels of Detail may be optional (indicated by a dashed outline to the Level of Detail symbol – as shown here for a Conversation Captured Level of Detail). This means that stopping at the previous Level of Detail is normally sufficient. Subsequent Levels of Detail may or may not be created depending on the context or circumstances in which the practice is being used.

Work Product Levels of Detail can also be concisely and accurately defined in terms of a set Checklist Items. For example, the Value Expressed Level of Detail for a Story Card might have Checklist Items:

The name of the User Story is visible on the front of the card.
There is a headline description on the front of the card that briefly describes what will be done to the product in a way that clearly conveys its value.
Activities can now be more fully defined in terms of what Levels of Detail they progress Work Products to as well as what State they progress Alphas to, for example as shown here for a Prepare User Story Activity.

Competency

A Competency encompasses the abilities, capabilities, attainments, knowledge, and skills necessary to do a certain kind of work, such as Leadership, Development and Testing.

Teams are central and critical to the success of endeavors and Activities. In particular, it is important to understand what kind of team members we need, with what kinds of Competencies.

Competency Level

A Competency Level defines what level of capability a team member has with respect to a Competency, for example Leadership Level 1 or Development Level 4.

The Essence Kernel has a standard set of five named Competency Levels listed above.   A key way in which we can describe Essence Activities is to define what Competencies we need at what minimum Competency Level to do the Activity well. This helps teams plan their Activities by agreeing, for example, which team members should participate in which Activities. For example:

A Backlog Refinement Activity might require a Stakeholder Representation Competency Level of at least Level 3: Masters, to successfully represent the stakeholders and communicate their needs
A Daily Stand-Up Activity might need a minimum Management Competency Level of at least Level 2: Applies, to ensure the meeting is scheduled, communicated, and has the available room and/or virtual communication facilities needed for it to work well.
Note that, if a team lacks the required Competency Level, then this indicates that they could either:

Acquire it, e.g. by coaching, training or recruiting, or
Look for alternative Practices that require lower Competency Levels.

Pattern

A Pattern is a generic mechanism for describing practice guidance that may relate to many other associated Essence elements.

Patterns are a flexible and invaluable way to provide additional supporting guidance of any kind that relates to the other Practice elements, and how these elements work together.

Examples of common types of Patterns include:

Roles – such as Product Owner or Project Manager which can be defined by referencing required Competencies and Competency Levels, and defining their responsibilities with respect to the Activities, Alphas, Work Products etc.
Milestones – such as Ready to Start or Ready to Release, which can be defined in terms of the Alpha States and Checklist Items that must be achieved for the milestone to be achieved
Games – that use different combinations of elements of the Essence Kernel and/or Essence Practices to help teams to plan, do, assess and adapt their endeavors
Techniques – such as Brainstorming or Root-Cause Analysis that can be described and related to the Activities that might benefit from using them.

Resource

Since an Essence Practice focuses primarily on the “bare-bone” essentials, it is useful to be able to reference other Resources where more supporting guidance can be found.

A Resource is a source of information or content, such as a website, that is outside the Essence model and referenced from it, for instance by a URL or a book reference.