CSL2060
ESSENCE KERNEL AND ADDITIONAL PRACTICES

VOILA

CHAT APPLICATION

Essence Kernel:

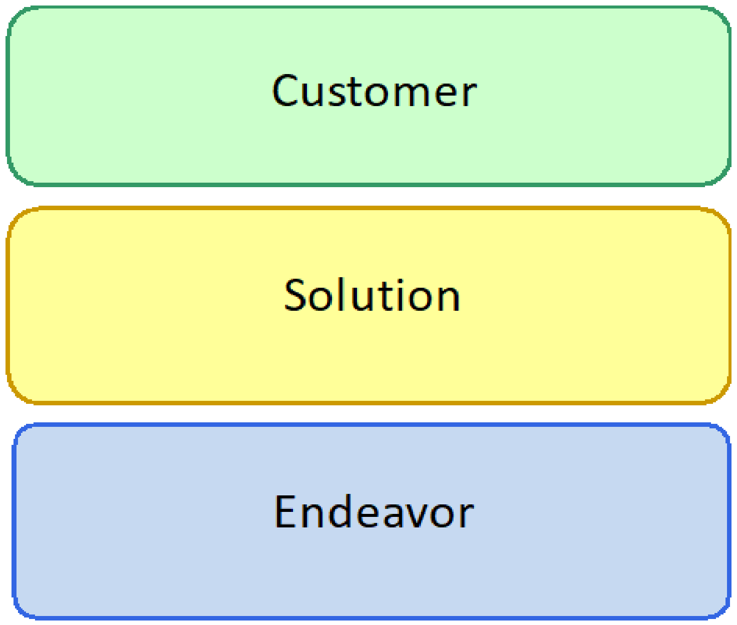
The Essence kernel captures the idea of a common ground based on the things we always have, the things we always do, and the skills we always need in order to conduct software engineering endeavours. The objective of the kernel is to provide a set of universal elements to define, use and adapt methods and practices supporting the day-to-day activities of the software development team in a dynamic way while fostering communication and collaboration. In order to establish a common ground, the kernel addresses the customer, solution and endeavour facet of software engineering. Within these three areas of concern, the kernel defines alphas, activity spaces and competencies.

Area of Concerns:

The kernel is structured alongside three areas of concern: The customer area of concern, the solution area of concern, and the endeavour area of concern.

- The **customer area of concern** deals with the actual use and exploitation of the software system to be developed. Within this area, the team understands the opportunity to build software, i.e. the value the software system provides to the other stakeholders.
- The **solution area of concern** addresses the specification, design and implementation of the software system. Within this area, the team establishes a shared understanding of the requirements and implements, builds, tests, deploys and supports the software system.
- The **endeavour area of concern** engages in the concerns of the team and the management of its work. Within this area, the team is formed and work is being

planned and organized; the team progresses the work in alignment with the agreed way-of-working.



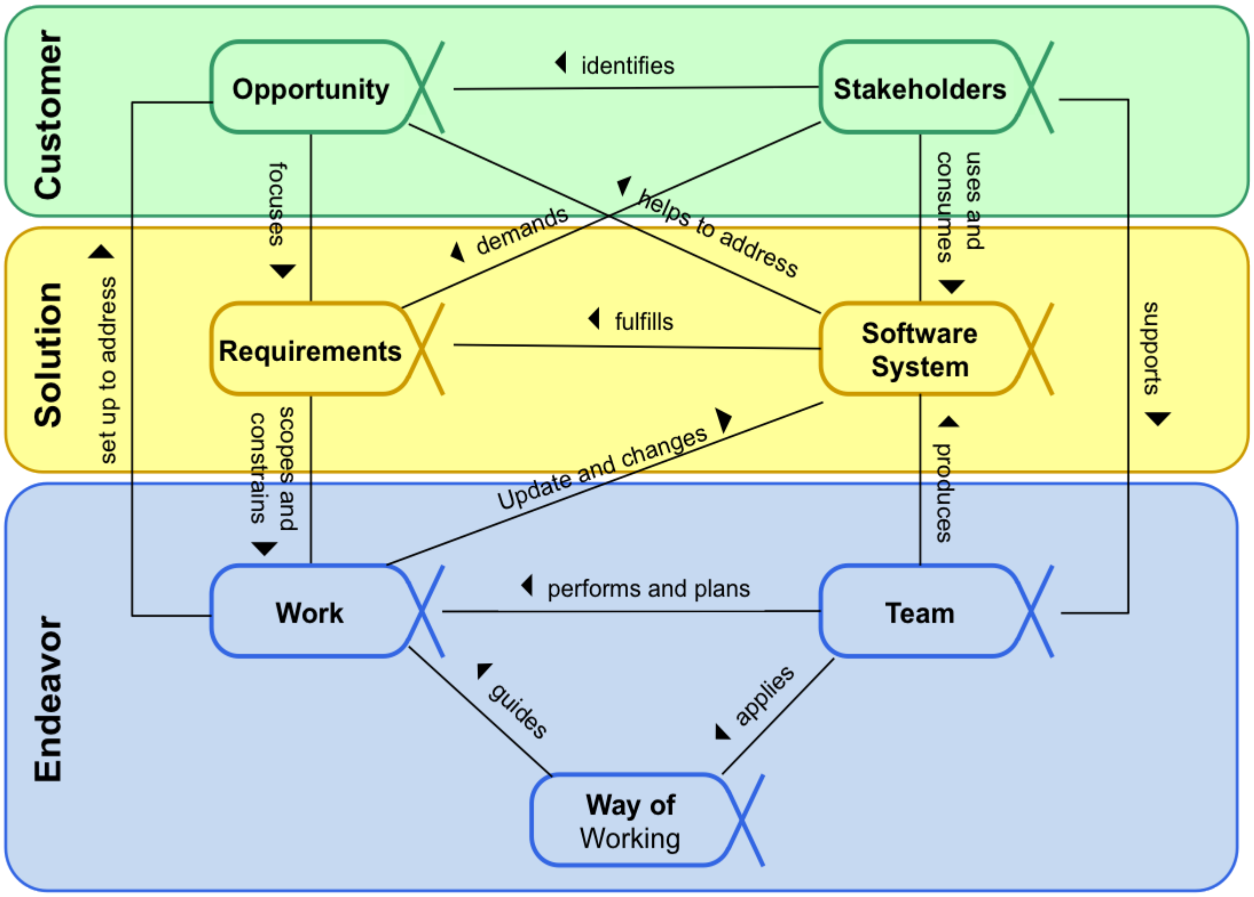
Alphas:

Alphas can be regarded as one of the most central concepts of the kernel. Alphas represent the essential things to work with when conducting a software engineering endeavour, i.e. the things a team produces, manages and uses in the process of developing, maintaining and supporting software. As such, they form the base of the common ground for the definition of software engineering methods and practices. The kernel defines a set of seven alphas. Each alpha belongs to a specific area of concern. The customer area of concern specifies the **Opportunity** and **Stakeholders** alpha. Within the solution area of concern, the **Requirements** and the **Software System** alpha are defined. The endeavour area of concern includes the **Team**, the **Work** and the **Way of Working** alphas. During the execution of an endeavour, the team progresses the alphas from an initial state to a target state. Therefore, each alpha contains a set of pre-defined states.

Alpha states allow to assess the progress and health of the alphas during the execution of the endeavour. Furthermore, the states allow to determine where the team currently stands and what to do next in order to progress. The assessment of each state is supported with a defined checklist associated with each state.

Overview

Overview of all Essence kernel alphas across the three areas of concern.



The Alphas should not be viewed as a physical partitioning of your endeavor or as just abstract work products. Rather they represent critical indicators of the things that are most important to monitor and progress.

Stakeholders: Those people or group of people who are affected or affect the software system.Team members, service providers, and users of the software are stakeholders.

Stakeholders demand opportunity, are supported by a team, provide funding and ensure acceptance of software systems.

Recognised: Stakeholders have been identified.

Represented: There Representatives have been authorised.

In Agreement: Minimal Expectations have been agreed and comprehensible.

Satisfied in use and deployment: Feedback by stakeholder is recorded and Ready for deployment. Systems meet expectation

Involved: Representative assists the team to work.

Opportunity: The set of circumstances that makes it appropriate to develop or change a software system. It is the opportunity that unites the stakeholders and provides the motivation for producing a new or updated software system. It is by understanding the opportunity that you can identify the value and the desired outcome that the stakeholders hope to realize from the use of the software system, either alone or as part of a broader business or technical solution.

Identified: Ideas behind the opportunity and other stakeholders to be identified.

Value Established: Solution value and impact has been quantified.

Solution Needed: Solution is identified, established and needed for a solution proposed.

Benefit Accrued: Benefits have been made from the deployed solution.

Addressed: Opportunities are addressed and stakeholders are satisfied.

Viable: Solution is possible considering all the risks.

Requirements: 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. Requirements must be addressed and satisfied stakeholders.

Conceived: Users identified, clear opportunity and agreement by stakeholders to be produced.

Coherent: Consistent, clear, rational requirement and shared. Teams know and agree on what to be delivered.

Bounded: Clear success of system and purpose agreed.

Fulfilled: Requirements fully satisfied and accepted by stakeholders.

Addressed: System and requirements to be matched and system worth making operational.

Acceptable: Value to be realized clear and testable.

Software System: A system made up of software, hardware, and data that provides its primary value by the execution of the software.A software system can be part of a larger software, hardware or business solution. We use the term software system rather than software because software engineering results in more than just a piece of software.

Architecture Selected: Architecture selection criteria agreed and technologies are selected.

Usable: Software systems are usable from the point of view of the user.

Demonstrable: System exercised and performance are measured.

Ready: Stakeholders want the system and operational support in place.

Retired: The software system is retired and no longer supported or updated.

Operational: Systems available for use and live.

Work: In the context of software engineering, work is everything that the team does to meet the goals of producing a software system matching the requirements, and addressing the opportunity presented by the stakeholders. The work is guided by the practices that make up the team's way-of-working.

Initiated: Required clear results and clear priorities.

Started: Development is started and Progress is monitored.

Prepared: Cost and efforts are estimated and risk exposure understood.

Closed: Lessons are learned and everything is achieved.

Concluded: System accepted and results achieved.

Under Control: Tasks are completed, progress measured and risk under control.

Team: A group of people actively engaged in the development, maintenance, delivery or support of a specific software system. One or more teams plan and perform the work needed to create, update and/or change the software system.Normally a team consists of several people. Occasionally, however, work may be undertaken by an individual creating software purely for their own use and entertainment. A team requires at least two people, but most of the guidance provided by Team Alpha can also be used to help single individuals when creating software.

Seeded: Mission has been defined and responsibilities are outlined.

Collaborating: Team works as a unit and open to communication.

Formed: Members are introduced and accept work. Members are committed to learn.

Adjourned: responsibilities are fulfilled and mission is concluded.

Performing: Problems are addressed and continuously adapting to change.

Way-of-Working: The tailored set of practices and tools used by a team to guide and support their work.

Principle established: Principles followed being identified.

Foundation Established:Basic foundation of the work is being laid.

In Use: Practice and tools are adapted.

In Place: All things are beings done in order

Retried:Work has been accomplished and team left.

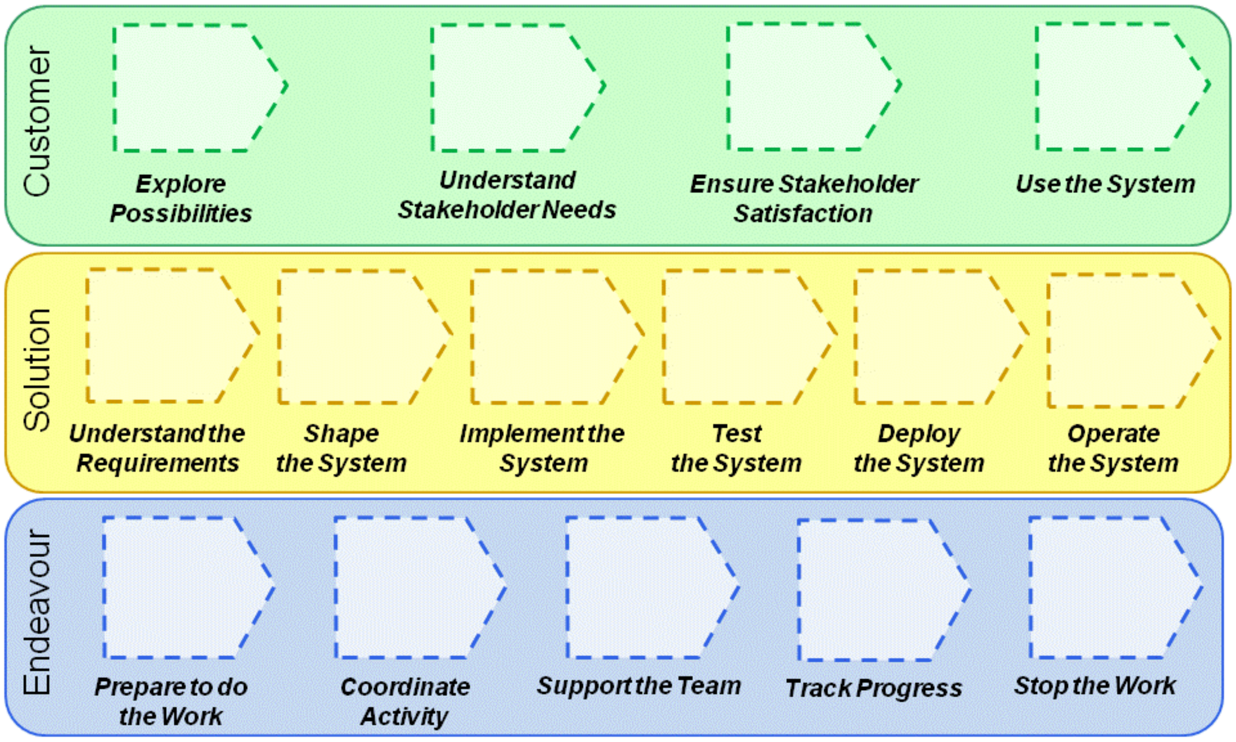
Working well: Way of working is adopted well.

Activity Spaces:

Activity spaces represent placeholders for the essential activities in software engineering endeavours. They complement the alphas and provide an activity based view on software engineering. An activity space serves as placeholders for one or more concrete activities. Each activity space has a set of objectives and is related to certain kernel alphas which are required as input or provided as output. Each activity space is defined including completion criteria which relate to certain alpha states.

Overview

Overview of all defined kernel activity spaces across the three areas of concern.



Project Specific Kernel

<u>Customers</u>: Coordinate with Stakeholders

<u>Solution</u>: Maintaining the System, Validate the System

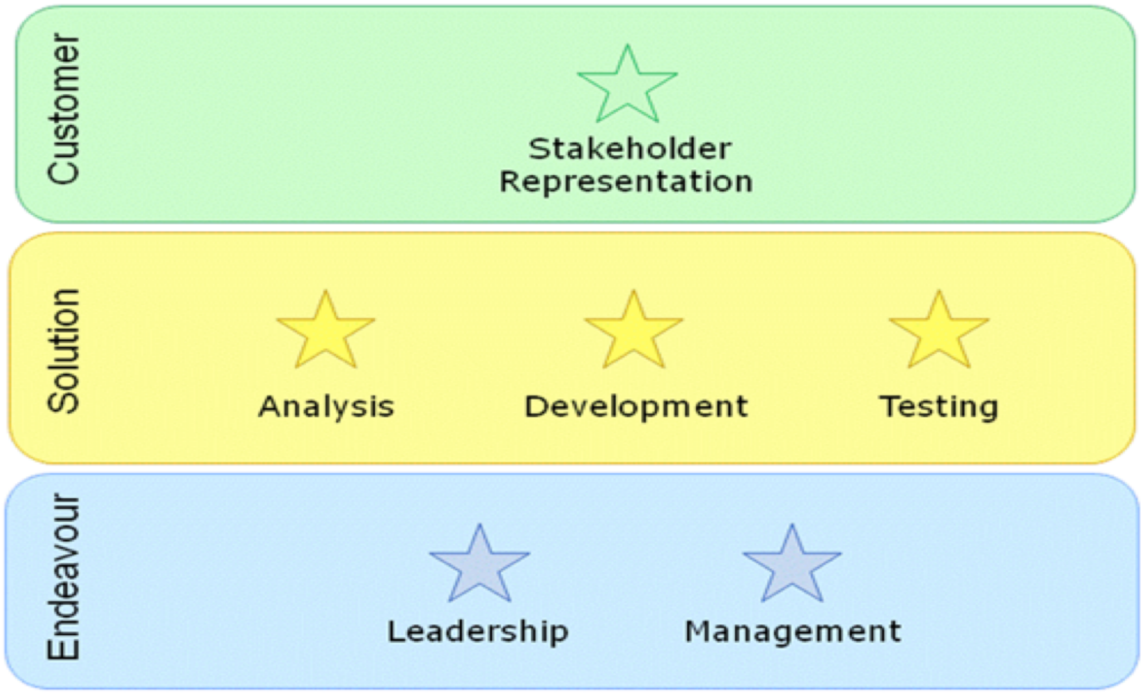
<u>Endeavour</u>:Work Distribution

Competencies:

The kernel competencies complement the kernel alphas and activity spaces with the key competencies required to conduct software engineering endeavours. The defined kernel competencies consist of communication, engineering and management capabilities.

Overview

Overview of specified competencies across the three areas of concern.



Each competency can be assessed by five levels of achievements which build upon each other. Team members with competencies at level 1 (“Assists”) demonstrate a basic understanding of required concepts and can follow instructions. At level 2 (“Applies”), the concepts are applied in simple contexts based on first experiences. Level 3 (“Masters”) defines the competency to apply the concepts in most contexts. Team members possessing this competency are considered to have enough experience to work without supervision. Competencies at level 4 (“Adapts”) allow you to judge when and how to apply the concepts in more complex contexts. Competencies at Level 5 (“Innovates”) represent recognized experts who extend concepts, apply them to new contexts, and inspire others.

Project Specific Kernel

<u>Customers</u>: Stakeholder Identification

<u>Solution</u>: Dart (Syntax) Knowledge, Flutter Knowledge, Firebase Knowledge, Maintenance, Debugging

<u>Endeavour</u>: Learning Skills, Problem Solving, Coordination, Communication, Critical Thinking

Patterns:

Patterns are generic solutions to typical problems. In our daily life, we see patterns everyday. In a classroom, we expect to have the teacher in front, with rows of desks and chairs for students. Such a pattern is designed for the teacher to transmit knowledge as efficiently as possible. Some classrooms are designed such that students are arranged in circles for greater discussion and discovery.

Roles

Many pieces of work require more than one competency. For instance, when developing software you need more than just a Development competency. You also need to know how to find out what to develop, which requires an Analysis competency. Thus, when assigning an individual this work, he/she needs to have these competencies so a common solution to the problem of assigning work is to define a special kind of pattern, which does that. We call such a pattern a role. A role requires a minimum level of each competency to do the job effectively.

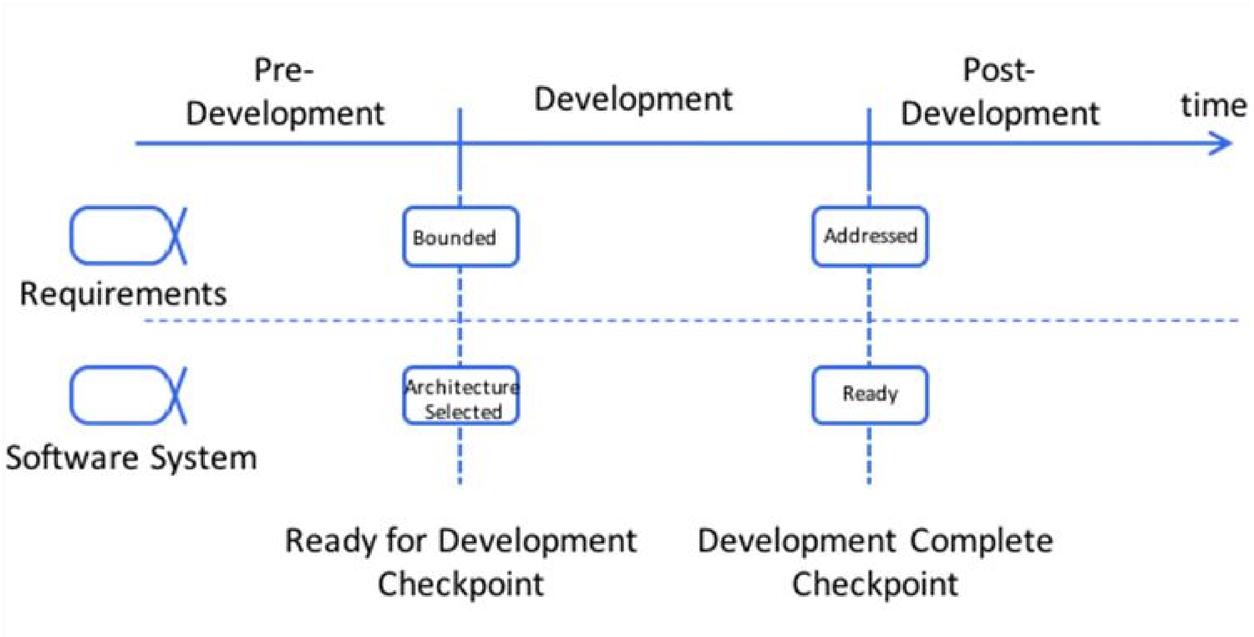
When someone is given a role she/he should have the needed competencies to succeed. If not, he/she should be offered additional training and coaching to carry out the role successfully.

Checkpoint Pattern

A pattern could be used to define a checkpoint. A *checkpoint* is a set of criteria to be achieved at a specific point in time in a software engineering endeavor. They are frequently used within all types of software engineering endeavors. Checkpoints are key points in the life cycle of a software endeavor where an important decision is to be taken. They are used by organizations as points where they stop and think about whether it makes sense to proceed or not depending on if there is or isn’t faith in what has been done. A decision could also be made to move forward but to go in a different direction because of a known problem or risk.

A checkpoint is simply expressed by a set of alpha states that must have been achieved in order to pass the checkpoint. This pattern can be reused for other similar endeavors trying to get to the same checkpoint.

In the simplest case, the timeline of a software engineering endeavor can be divided into three phases, pre-development, development, and post-development. In the figure, the rounded boxes are states associated with the alphas.



Pre-development could be defined as the phase when the team and the stakeholders are determining what they need to do. The end of the pre-development phase needs to include any essential items that need to be achieved before the team can begin developing the software system. One example might be funding approval to proceed with development.

Development is the phase when the team works collaboratively to create a software system that addresses the agreed needs and objectives. The phase ends when the software has reached a point where it can be used by real users.

Post-development is the phase when the software system is delivered to the user community and the software is used with real clients.