

The Essence Language and SE Theory



**Idaho State
University**

Computer
Science

Dr. Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will:

- Understand the kernel of the Essence Standard
- Understand how theory and some ideas for SE theory





The Essence Kernel

CS 3321

ROAR

The Essence Kernel

- The Essence kernel is the set of Essence elements that would always be found in all types of software system endeavors.
 - For instance, the element architecture was discussed as a kernel element.
 - The opinion was that while for many systems it is critical to identify an architecture there are many simpler systems where architecture is not an issue.
 - Since it is not common to all projects, architecture is not a concern that every endeavor has to face, it didn't qualify as a kernel element.
- In the following slides we will illustrate the elements that are part of Essence Kernel

Areas of Concern

- The Essence kernel elements are organized around 3 areas of concern (as we have already seen):

Customer – This area of concern contains everything to do with the actual use and exploitation of the software system to be produced.

Solution – This area of concern contains everything related to the specification and development of the software system.

Endeavor – This area of concern contains everything related to the development team and the way that they approach their work.

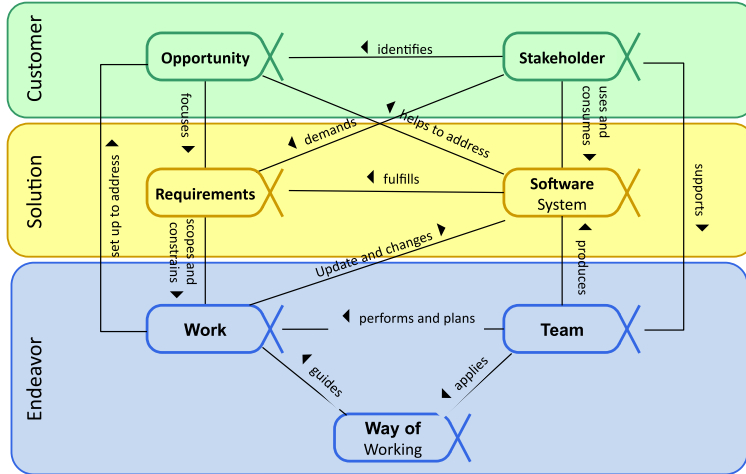
The Essence Kernel

The kernel elements are fundamental of four kinds:

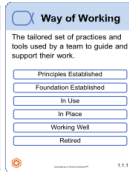
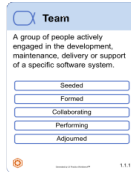
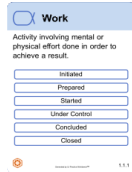
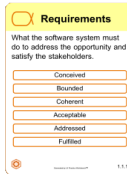
- ❶ The essential things to work with – **Alphas**
 - ❷ The essential things to do – **Activity Spaces**
 - ❸ The essential capabilities needed – **Competencies**
 - ❹ The essential arrangements of elements – **Patterns**
- Finding the right elements is crucial
 - They must be universally acceptable, significant, relevant and guided by the notion that:
 - “You have achieved perfection not when there is nothing left to add, but when there is nothing left to take away.”

1. The Alphas

We have already seen the Kernel Alphas



Essence Kernel Alpha States



- The OMG standard defines the states for each kernel alpha shown
- The details of each state can be found in the Essence standard, and we will not go deeper into each of them here
- You can download these from Moodle

2. The Activities and Activity Spaces

- In every software development endeavor you carry out a number of activities
 - **Essence does not define any activities**
 - how your team goes about capturing and communicating the requirements can be very different from team to team
 - **Essence defines a number of activity Spaces**

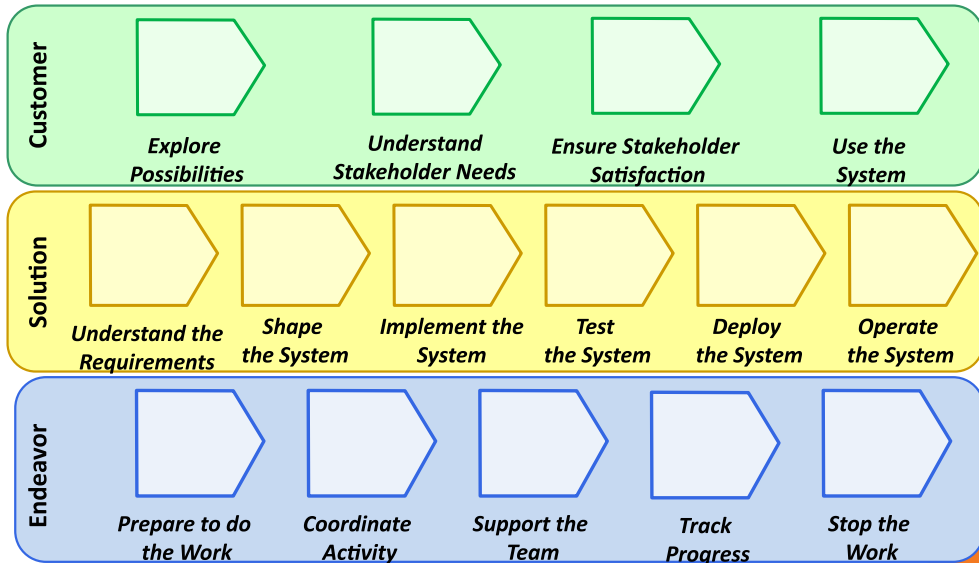
Definition

Activity Spaces are generic placeholders for specific activities

- Since the activity spaces are generic
 - They are method-independent
 - They could be standardized and are thus part of the Essence standard.
 - Each activity space can be extended with concrete activities that progress one or more alphas
- The activity spaces are packages used to group activities, which are related to one another.
- The activity spaces represent the essential things that have to be done to develop software.



Standard Activity Spaces



Standard Activity Space Descriptions

Customer

- **Explore Possibilities** – Explore the possibilities presented by the creation of a new or improved software system. This includes the analysis of the opportunity and the identification of the stakeholders.
- **Understand Stakeholder Needs** – Engage with the stakeholders to understand their needs and ensure that the right results are produced. This includes identifying and working with the stakeholder representatives to progress the opportunity.
- **Ensure Stakeholder Satisfaction** – Share the results of the development work with the stakeholders to gain their acceptance of the system produced and verify that the opportunity has been addressed.
- **Use the System** – Observe the use of the system in a live environment and how it benefits the stakeholders.

Solution

- **Understand the Requirements** – Establish a shared understanding of what the system to be produced must do.
- **Shape the system** – Shape the system so that it is easy to develop, change and maintain, and can cope with current and expected future demands. This includes the architecting and overall design of the system to be produced.
- **Implement the System** – Build a system by implementing, testing and integrating one or more system elements. This includes bug fixing and unit testing.
- **Test the System** – Verify that the system produced meets the stakeholders' requirements.
- **Deploy the System** – Take the tested system and make it available for use outside the development team

Endeavor

- **Prepare to do the Work** – Set up the team and its working environment. Understand and commit to completing the work.
- **Coordinate Activity** – Co-ordinate and direct the team's work. This includes all ongoing planning and re-planning of the work, and re-shaping of the team.
- **Support the Team** – Help the team members to help themselves, collaborate and improve their way of working.
- **Track Progress** – Measure and assess the progress made by the team.
- **Stop the Work** – Shut-down the software engineering endeavor and handover of the team's responsibilities.



Activity Space Card



Implement the System

Build a system by implementing, testing and integrating one or more system elements. This includes bug fixing and unit testing.

On Entry



Requirements



Way of Working



Software System: Architecture Selected

On Exit



Software System: Demonstrable - Usable - Ready



Activity name



Very brief
Activity description



Inputs for activity



Outputs of activity

3. Competencies

Definition

Competencies are generic containers for specific skills

- Specific skills, for example Java programming, are not part of the kernel because this skill is not essential on all software engineering endeavors.
- But competency is always required and it will be up to the individual teams to identify the specific skills needed for their particular software endeavor.
- A common problem on software endeavors is not being aware of the gap between the competencies needed and the competencies available.
 - The kernel approach will raise the visibility of this gap.

Standard Competencies

- Competencies are aligned to their focus areas
- Essence Kernel Standard competencies are needed for any Software Engineering Endeavor, independently of methods and techniques adopted



Standard Competencies Descriptions

Customer

- **Stakeholder Representation** – This competency encapsulates the ability to gather, communicate, and balance the needs of other stakeholders, and accurately represent their views.

Solution

- **Analysis** – This competency encapsulates the ability to understand opportunities and their related stakeholder needs, and to transform them into an agreed upon and consistent set of requirements.
- **Development** – This competency encapsulates the ability to design, program and code effective and efficient software systems following the standards and norms agreed upon by the team.
- **Testing** – This competency encapsulates the ability to test a system, verify that it is usable and that it meets the requirements.

Endeavor

- **Leadership** – This competency enables a person to inspire and motivate a group of people to achieve a successful conclusion to their work and to meet their objectives.
- **Management** – This competency encapsulates the ability to coordinate, plan and track the work done by a team

Competency Levels

Competency levels of achievement:

- 1 **Assists** – Demonstrates a basic understanding of the concepts and can follow instructions
- 2 **Applies** – Able to apply the concepts in simple contexts by routinely applying the experience gained so far.
- 3 **Masters** – Able to apply the concepts in most contexts and has the experience to work without supervision.
- 4 **Adapts** – Able to apply judgment on when and how to apply the concepts to more complex contexts. Can enable others to apply the concepts.
- 5 **Innovates** – A recognized expert, able to extend the concepts to new contexts and inspire others.

- Each of the competencies has a competency level
- The competency level is the same across all of the kernel competencies

4. Patterns

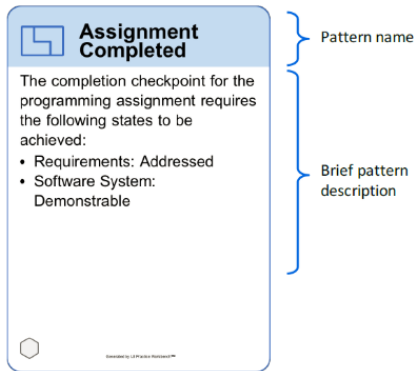
Definition

***Patterns* are generic solutions to typical problems**

- Patterns is the way **Essence** allows arrangements of elements to solve a specific problem.
- Patterns are optional elements (not required element of a practice definition) that may be associated with any other language element.
- *Patterns* examples exist in our daily life as well as in Software Engineering:
 - In a classroom, we often see the teacher in front, with rows of desks and chairs for students. This is a common teaching pattern.
 - In SW Eng we use patterns very often. Some examples are:
 - CheckPoints, Student Pairs, etc.
- *Roles* are a special form of *Pattern*

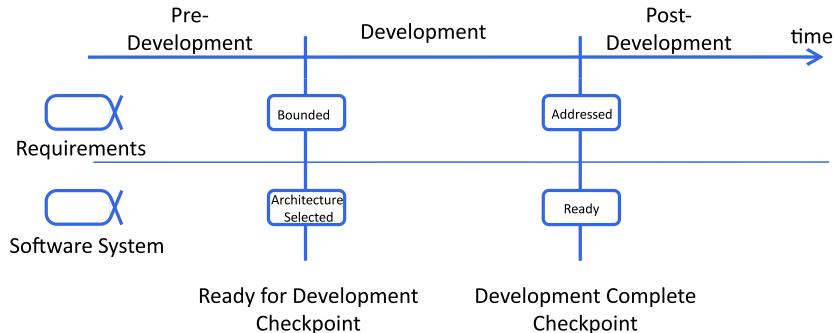
Pattern Example: Checkpoint

- A *checkpoint* is a set of criteria to be achieved at a specific point in time where an important decision is to be taken.
 - 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.



Using the Checkpoint Pattern

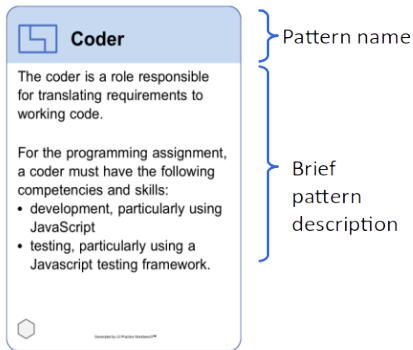
- Let's use Checkpoints to decide when to start and when to finish development of a software project.



- In this example, there are two checkpoints.
 - What are the checkpoints?
- The criteria for these two checkpoints are expressed using alpha states.
 - What are the Alpha States for each Checkpoint?


Roles: A Special kind of Pattern

- Roles represent the set of competencies needed to do a job effectively
 - Roles are a special kind of pattern that apply to people
 - Example roles are Coder, Analyst, Tester
- Responsibilities to achieve a task are assigned to the task owner, that could be playing a role, but the responsibilities are not part of the role definition




Essence Element Cards Summary

Alpha


 **Software System**

A system made up of software, hardware, and data that provides its primary value by the execution of the software.

- Architecture Selected
- Demonstrable
- Usable
- Ready
- Operational
- Retired


 1.1.1


Work Product

 **Code**


Good code that not only implements requirements, but also in a self-explanatory way.

- Pseudo Coded
- Code Completed
- Code Explained

Describes:  Software System


 1.1.1

Activity

 **Write Code**

Collaborate together to produce good quality code that meet requirements.

- Requirements: Bounded
- Implement the System
- Development
- Requirements: Addressed
- Software System: Ready
- Code: Code Completed

 1.1.1

Competency


 **Development**

The ability to design and program effective software systems following the standards and norms agreed by the team.

- Innovates
- Adapts
- Masters
- Applies
- Assists


 1.1.1

Pattern


 **Assignment Completed**

The completion checkpoint for the programming assignment requires the following states to be achieved:

- Requirements: Addressed
- Software System: Demonstrable


 1.1.1

Activity Area

 **Implement the System**

Build a system by implementing, testing and integrating one or more system elements. This includes bug fixing and unit testing.

- On Entry
- Requirements
- Way of Working
- Software System: Architecture Selected
- On Exit
- Software System: Demonstrable - Usable - Ready

 1.1.1



Software Engineering Theory

CS 3321

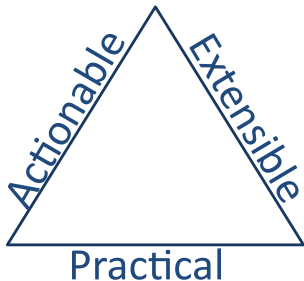
ROAR

What is Essence?

- Essence provides a common ground for Software Engineering
 - It is very important to have such common ground
 - It is more than a conceptual mode
 - It allows to represent any software engineering method
- Essence Kernel is
 - A **thinking framework** for teams to reason about the **progress** they are making and the **health** of their **endeavors**.
 - A **framework** for teams to **assemble** and continuously improve their **way of working**.
 - The **common ground for improved communication**, standardized measurement, and the sharing of best practice.
 - A **foundation for accessible, inter-operable method** and practice definitions.
 - And most importantly, a way to **help teams understand where they are, and what they should do next**.



Essence Guiding Principles



- Alphas help assess & drive progress and health of the project
- Each state has a checklist
 - Criteria needed to reach the state
- Alphas are method and practice independent

- Practices are distinct, separate, modular units
- Kernel allows us to create or tailor and compose practices to new methods
- Additional Alphas can be added

- Tangible through the cards
 - Cards provide concise reminders
- Practical through Checklists and Prompts
 - Utilizable on a daily basis helping making decisions

Essence and Agile (or other approaches)

- Essence Kernel doesn't compete with existing methods
- Essence kernel can be used with all the currently popular management and technical practices:
 - Scrum
 - Kanban
 - risk-driven
 - Iterative
 - Waterfall
 - Use-Case driven development
 - Behavior Driven development
 - Continuous Integration
 - Test Driven Development
- It will help all sizes of teams
 - form one-man bands to 1,000 strong software engineering programs
- The kernel supports the values of the Agile Manifesto
 - It values the 'use of methods' over 'the description of method definitions'

What is a Theory?

- Most theories share three characteristics
 - they attempt to generalize local observations and data into more abstract and universal knowledge
 - they generally have an interest in causality (cause and effect)
 - They often aim to explain or predict a phenomena
- Gregor proposes 4 goals for a theory:
 - 1 Describe the studied phenomenon
 - Function Point and SWEBOK could serve as an example
 - 2 Explain the how, why, and when of the topic
 - theory of cognition is aimed at explaining the human mind's limitations
 - 3 Besides explaining what has already happened also predict what will happen next
 - Cocomo attempts to predict the cost of software projects
 - 4 Prescribe how to act based on predictions
 - Alan Davis's 201 principles exemplify this goal

Where is the Theory for SE?

- Most academic disciplines are very concerned with their theories.
- Why is it that the software engineering community seems so uninterested in discussing its theories?
 - ***Software Engineering Doesn't Need Theory?***
 - Software engineering *isn't* doing fine.
 - All engineering fields need theory
 - The maturity of scientific disciplines can be measured by the unity of their theories
 - ***Software Engineering Already Has Its Theory?***
 - A discipline's significant theories should be able to provide answers to that discipline's significant questions...
 - ***Software Engineering Can't Have a Theory?***
 - Software engineering is a practical engineering discipline without scientific ambitions where rules of thumb and guidelines assume the role of theory
 - We don't believe that there's any rational reason for the lack of theoretical focus in software engineering
 - Without the predictive and prescriptive support of theory, software engineering would be relegated to the horribly costly design process of trial and error

Essence is founding the theory of SE

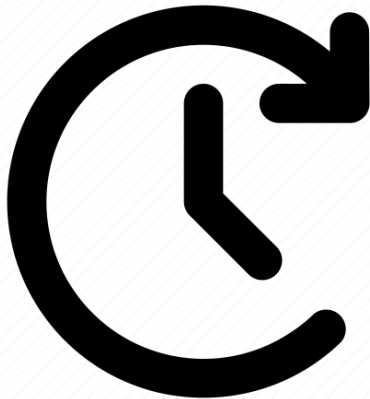
- Theory is generally used to
 - ① describe a phenomenon of interest
 - ② to explain and predict the phenomenon
- Description precedes prediction and to describe something, a language is needed
- There is currently no widely accepted predictive general theory of software engineering
 - However, the Essence takes the first step by proposing a coherent, general, *descriptive* theory of software engineering, i.e., a language of software engineering
 - But a complete consideration of the causality between concepts and thus prediction is beyond the current version of the Essence.

Conclusions

- Essence kernel is a spring board towards more mature software engineering practices and a more mature software engineering discipline
- Throughout the remainder of this course, we will demonstrate how Essence helps you and your teams collaborate more effectively

For Next Time

- Review Essentials Chapters 6 - 7
- Review this Lecture
- Come to Class
- Read Essentials Chapter 8 and 9
- Review Lecture 04
- Watch Lecture 04





Are there any questions?