Software Engineering Essentialized
Teaching material

The *Essence* Kernel

Giuseppe Calavaro, Ph.D.
IBM Big Data Practice Leader
External Professor at University of Rome “Tor Vergata”

www.semat.org

The Essence Kernel

- The Essence kernel is the set of Essence elements that would always be found in all types of software system endeavours.
  - 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 Concerns

- The Essence kernel elements are organized around 3 areas of concerns, that 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 fundamentally of four kinds:

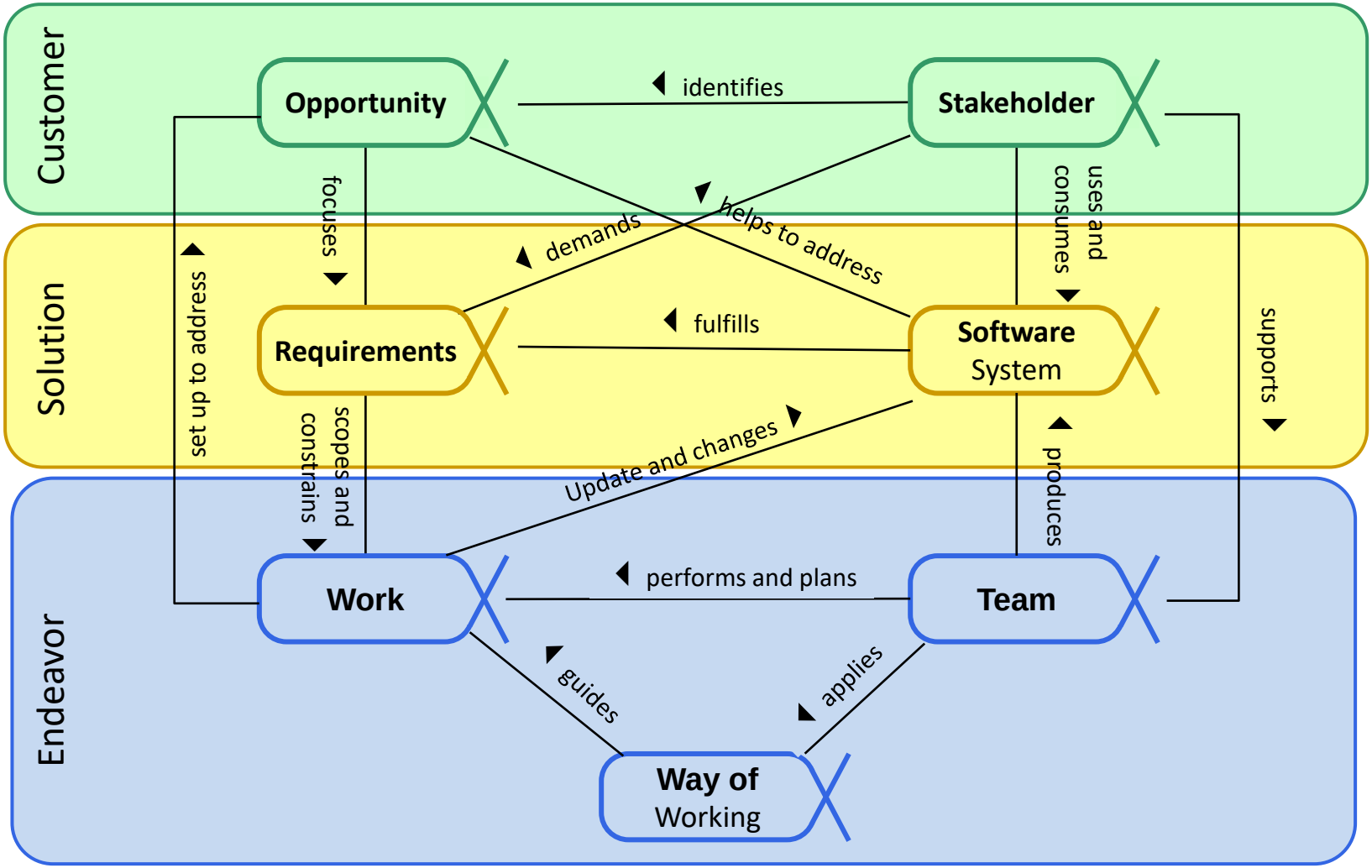
1. The essential things to work with – the alphas
2. The essential things to do – the activity spaces
3. The essential capabilities needed – the competencies
4. The essential arrangements of elements – the patterns.

- Finding the right elements is crucial.
- They must be universally acceptable, significant, relevant and guided by the notion that,

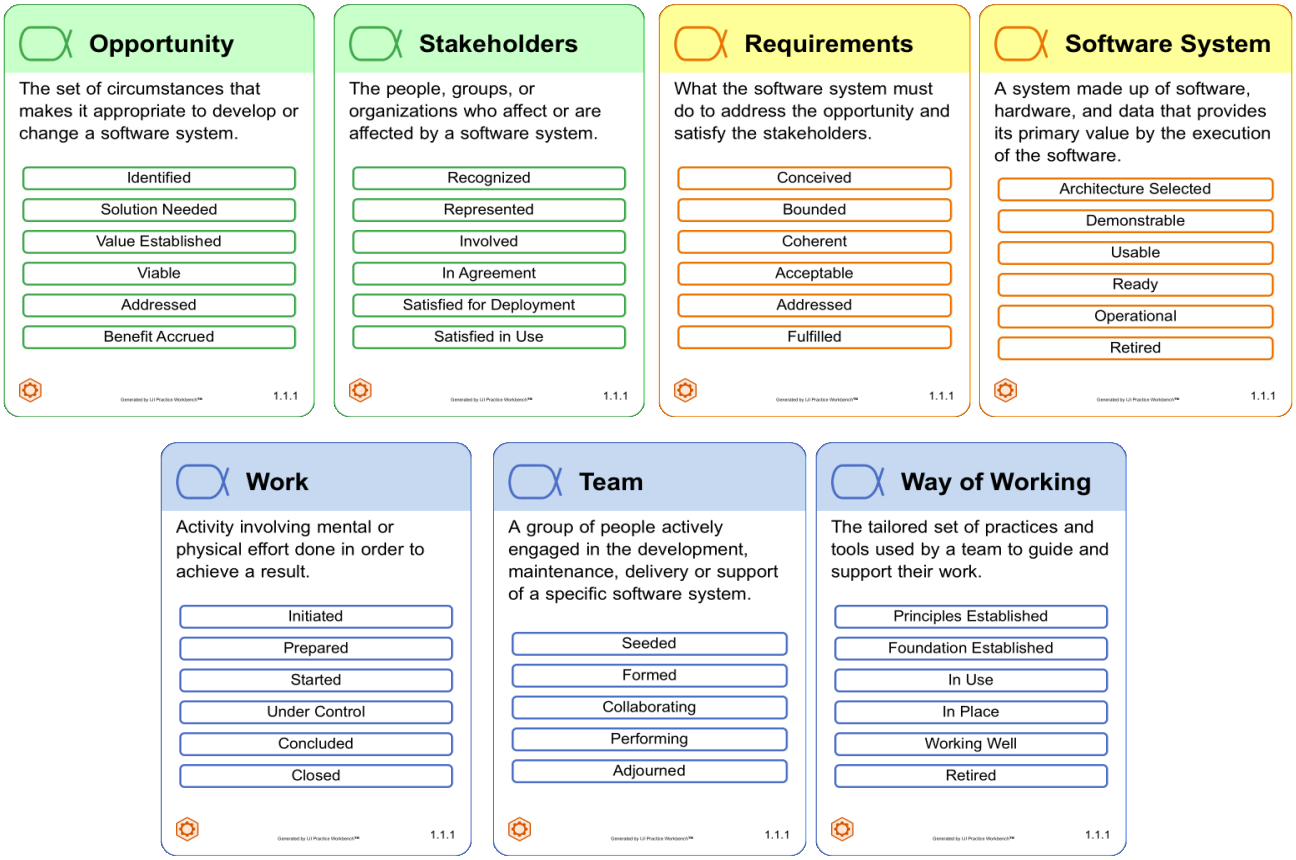
  "You have achieved perfection not when there is nothing left to add, but when there is nothing left to take away."*

1. The alphas

We have already seen the Kernel Alphas



States of the Alphas in the Essence Kernel



- 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 should be able to download them from the web site of the Essence book

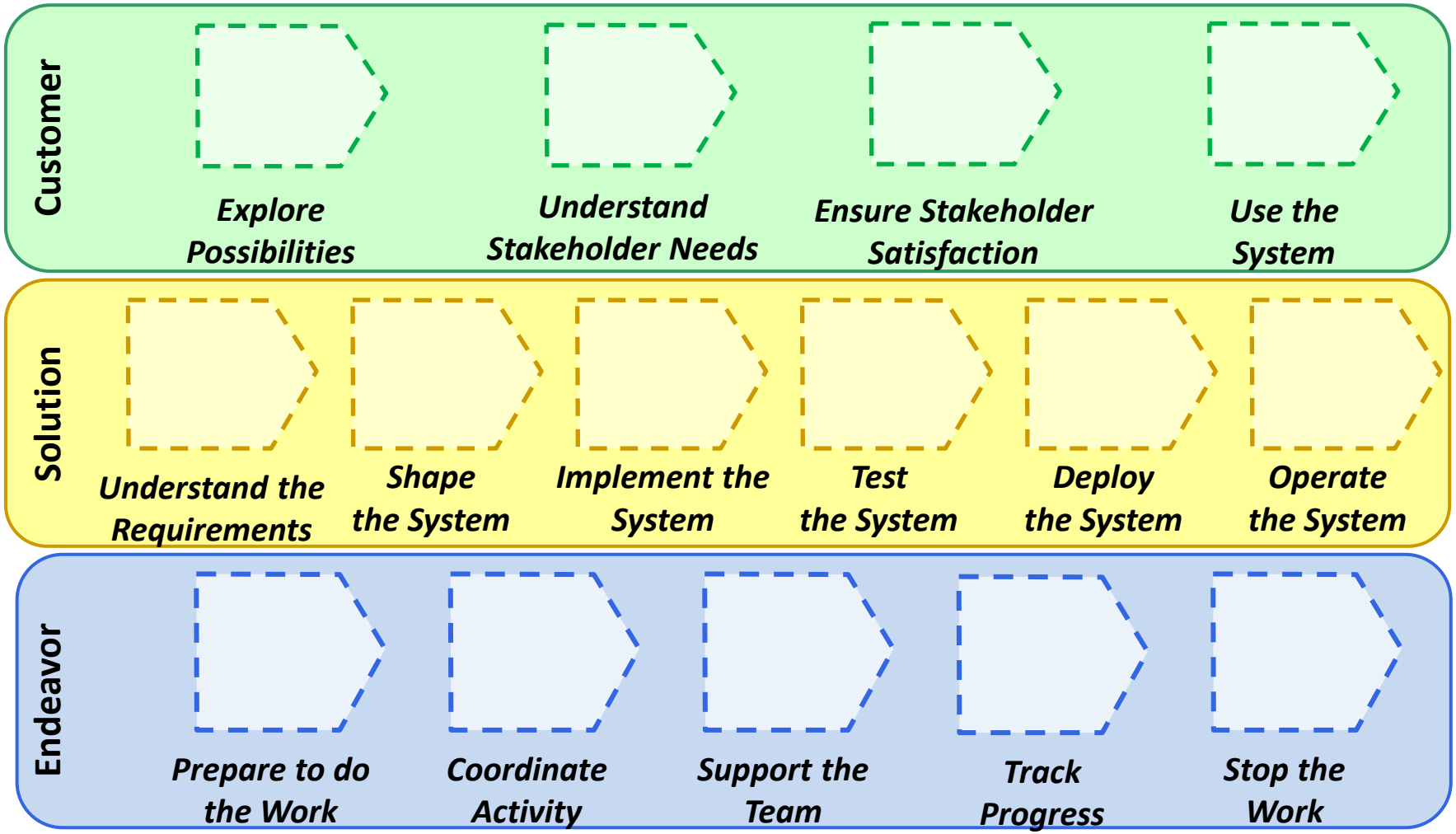
This picture is a snapshot and has problems. Remember to replace

2. The Activities and Activity Spaces

- In every software development endeavour 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.**
- **Def. 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

Activity Spaces in Kernel Standard

These are the Activity Space from Essence Standard



Activity Spaces Essence Standard Desc.

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 the system in a live environment and how it benefits the stakeholders.

Solution

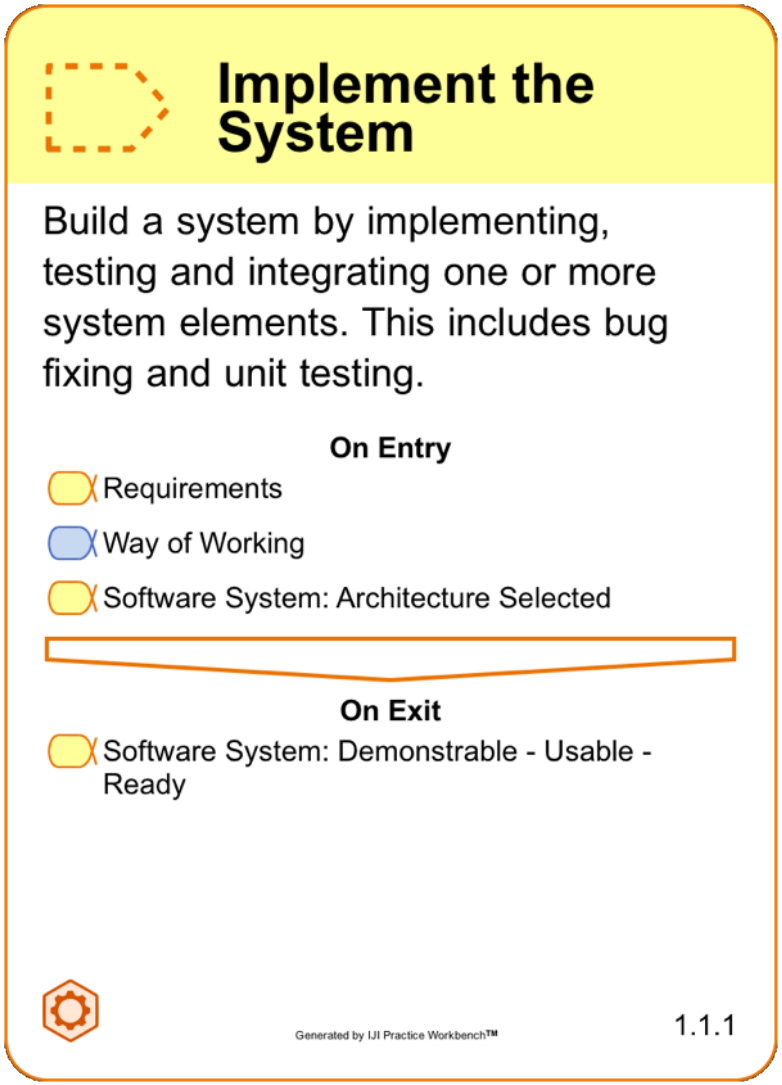
- **Understand the Requirements**
  Establish a shared understanding of what the system to be produced must do.
- **Shape the system**
  Shape the system so that it is easy to develop, change and maintain, and can cope with current and expected future demands. This includes the architecting and overall 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

Endeavour

- **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 endeavour and handover of the team's responsibilities.

Activity Space Card

- Activity space cards have very similar contents as activity cards



Activity name

Very brief Activity description

Inputs for activity

Outputs of activity

3. The Competencies

Def. *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 endeavours.
- But competency is always required and it will be up to the individual teams to identify the specific skills needed for their particular software endeavour.

- A common problem on software endeavours 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.

Competences in Essence Kernel Standard

- Competencies are aligned to the three focus areas
- Essence Kernel Standard competencies are needed for any Software Engineering Endeavour, independently then methods and techniques adopted

Competences Essence Standard Desc.

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.

Endeavour

- **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

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

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.

4. Patterns

Def. *Patterns* are generic solutions to typical problems

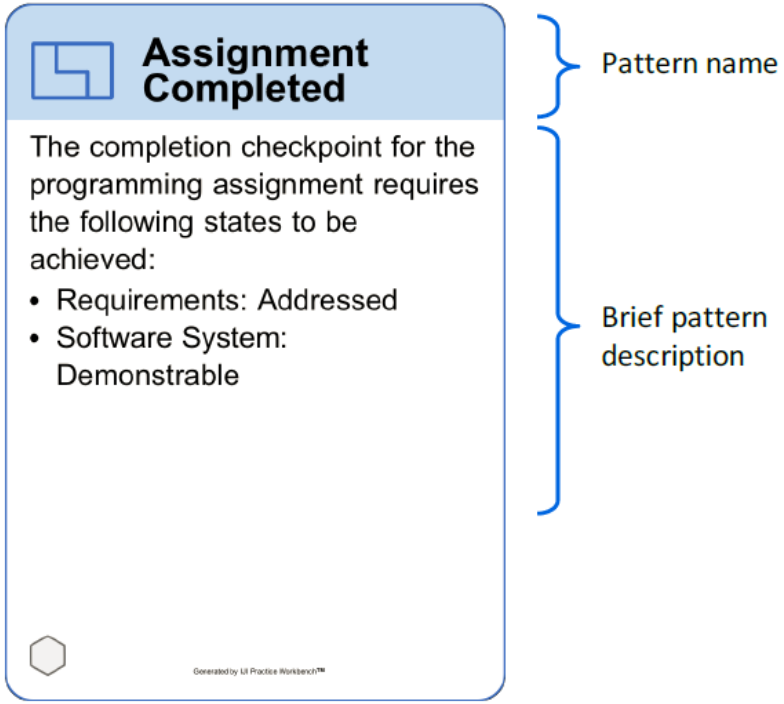
- Patterns is the way *Essence* allows arrangements of elements to solve a specific problem

- 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 special type of *Patterns*

A 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.

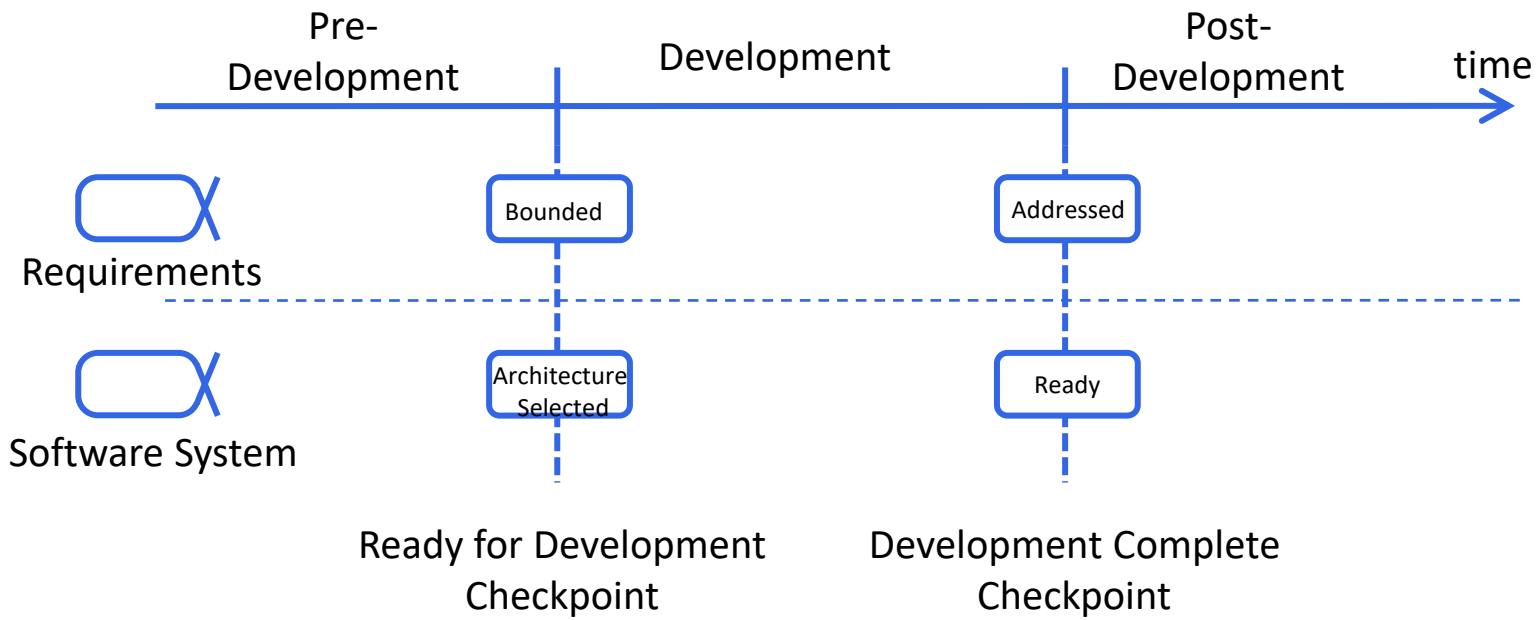
- This pattern can be reused for other similar endeavours trying to get to the same checkpoint.



This picture wasn't provided. This is a snapshot of the book. Remember to replace.

Using Checkpoint Pattern Example

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

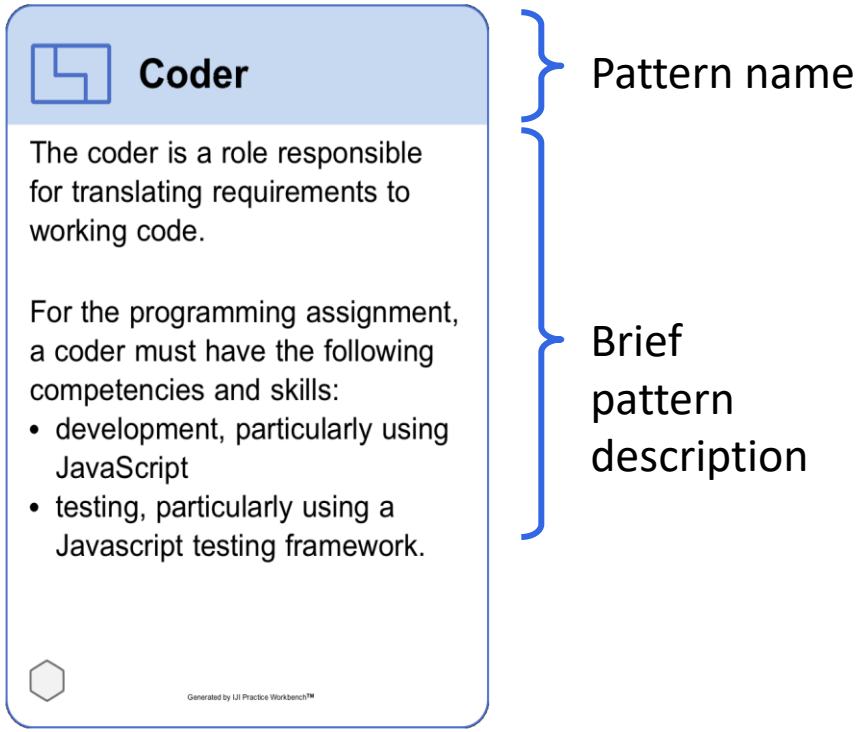


- In this example, there are two checkpoints.
  - What are the checkpoints?
- The criteria for these two checkpoints are expressed using alpha states.
  - What are the Alpha States for each Check Point?

Roles: A Special kind of Pattern

- Roles represent the set of competencies needed to do a job effectively
  - Roles are a special kind of patter that apply to people
  - Example of 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



Summary of Essence Elements and Cards

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

Generated by Ivi Practice Workbench™ 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

Generated by Ivi Practice Workbench™

Activity

Write Code

Collaborate together to produce good quality code that meet requirements.

Requirements: Bounded

Implement the System

1 Development

Requirements: Addressed

Software System: Ready

Code: Code Completed

Generated by Ivi Practice Workbench™

Competency

Development

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

- Innovates 5
- Adapts 4
- Masters 3
- Applies 2
- Assists 1

Generated by Ivi Practice Workbench™

Pattern

Assignment Completed

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

- Requirements: Addressed
- Software System: Demonstrable

Generated by Ivi Practice Workbench™

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

Generated by Ivi Practice Workbench™ 1.1.1

There is a snapshot that need to be replaced.

For Next Time

- Review Essentials Chapter 6
- Review this Lecture
- Read Essentials Chapter 7
- Come to Lecture