Software Engineering Essentialized
Teaching material

Conclusions: *Reflections on Essence Goals and Theory of Software Engineering*

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

www.semat.org

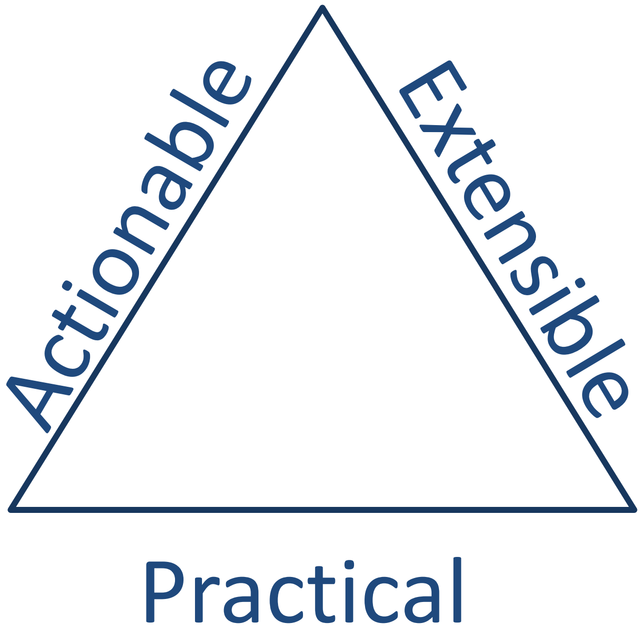
What is Essence?

- 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.

What makes Essence more than a conceptual framework?

Essence Guiding principles

- Alphas helps assess & drive progress and health of 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 allow 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

SEMAT

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
  - acceptance test driven development
  - continuous integration
  - test driven development
  - Etc.
- It will help all sizes of teams
  - from 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 phenomenon.
- Gregor[REF] 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. Beside 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[REF]

Where is the Theory for SW Engineering?

- Most academic disciplines are very concerned with their theories.
- Why 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

SEMAT

Essence is founding the theory of SW Eng

- Theory is generally used to
  1. describe a phenomenon of interest,
  2. to explain and predict that 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.

SEMAT

Conclusions

- Essence kernel is a spring board towards more mature software engineering practices and a more mature software engineering discipline.

- In the remaining parts of this course, we will demonstrate how Essence helps you and your teams collaborate more effectively.

Software Engineering Essentialized
Teaching material

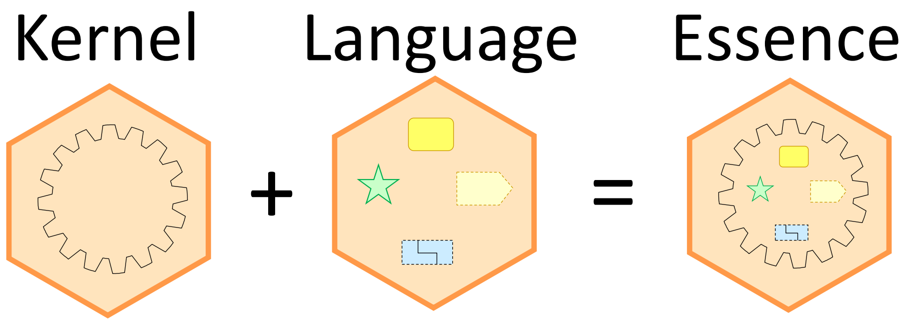
A summary of *Essence* Concepts and Elements

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

www.semat.org

The Essence of “Essence”

- Essence is made of 2 parts:

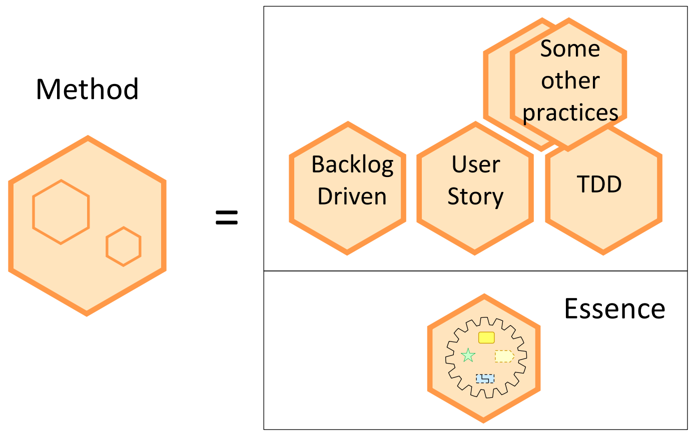


Essentialized Methods are

composition of **Essentialized practices**

That are described using Kernel Elements

Using Essence Language



SEMAT

Areas of Concerns

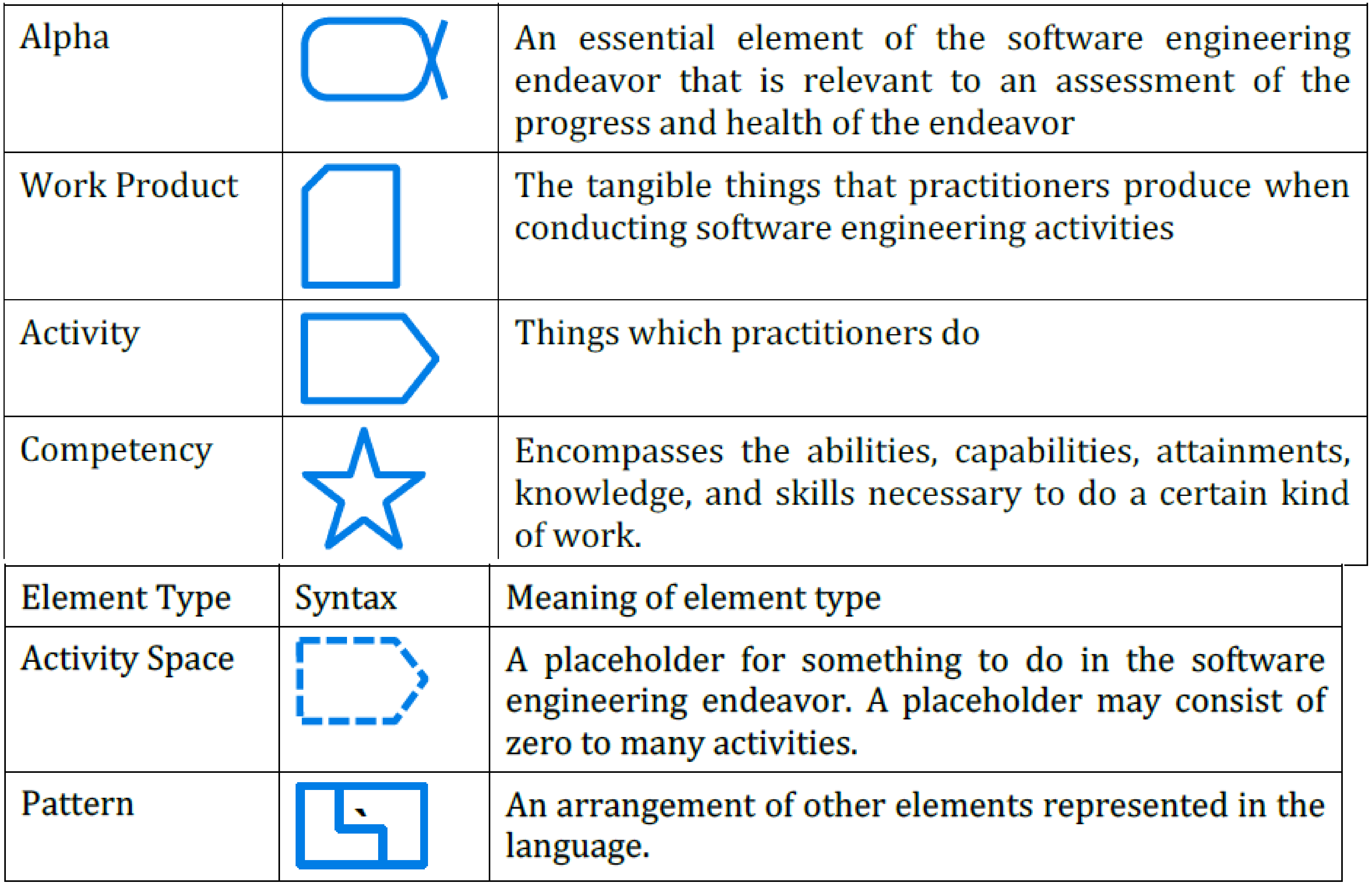
- The Essence kernel elements are organized around 3 areas of concerns:

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

Essence Language Element Types

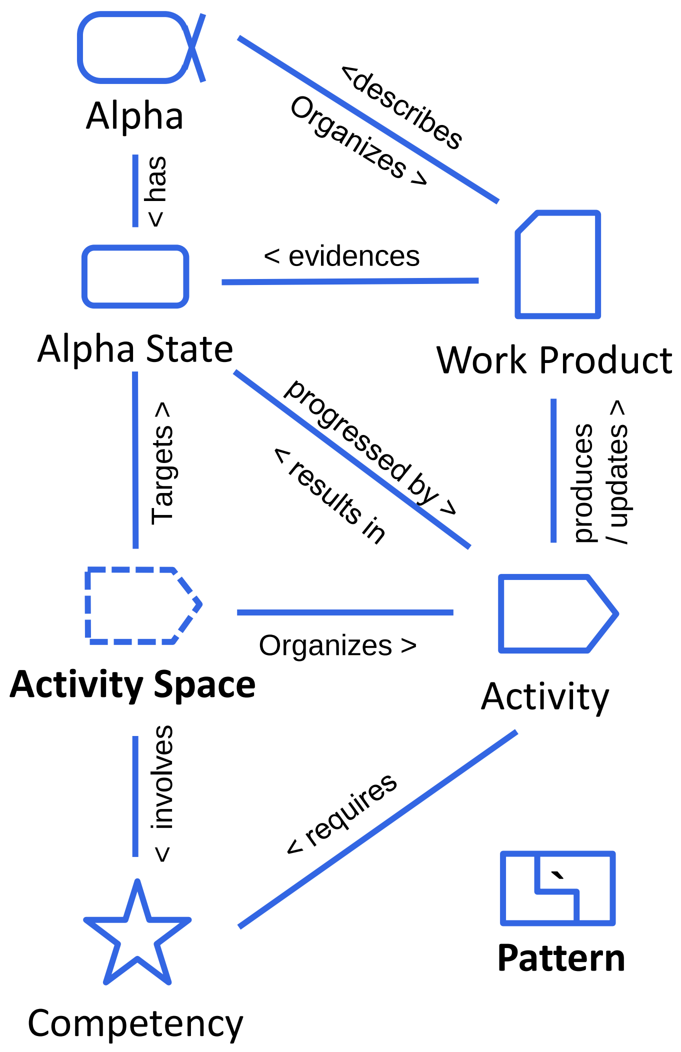


Element Type	Syntax	Meaning of element type
Alpha		An essential element of the software engineering endeavor that is relevant to an assessment of the progress and health of the endeavor
Work Product		The tangible things that practitioners produce when conducting software engineering activities
Activity		Things which practitioners do
Competency	☆	Encompasses the abilities, capabilities, attainments, knowledge, and skills necessary to do a certain kind of work.
Activity Space		A placeholder for something to do in the software engineering endeavor. A placeholder may consist of zero to many activities.
Pattern		An arrangement of other elements represented in the language.

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

Essence Language

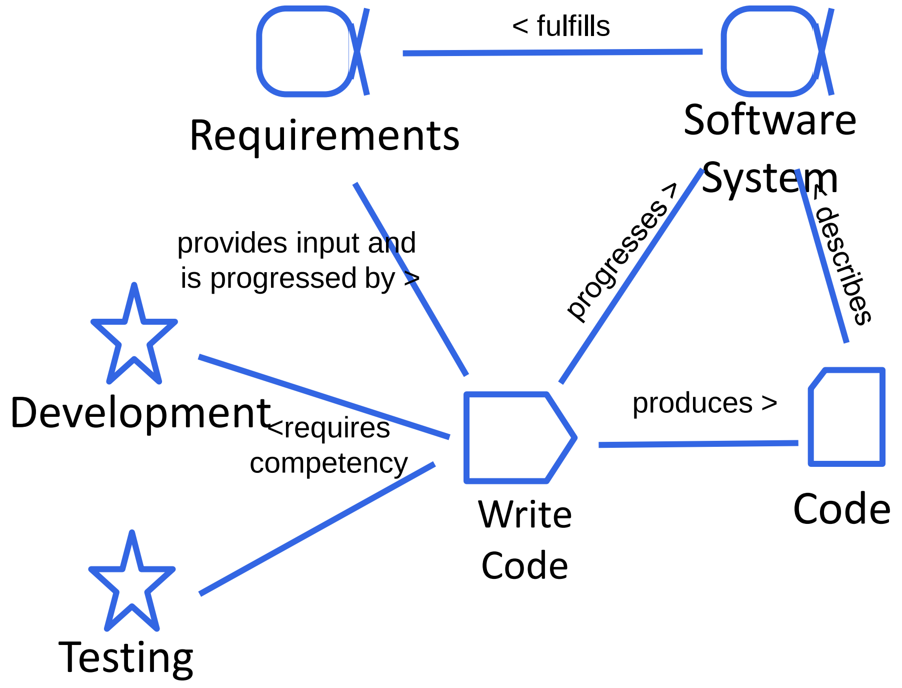
- These are the elements of ESSENCE LANGUAGE and their relationships
- Essentializing a Practice, means to describe a practice using the Essence language.



SEMAT

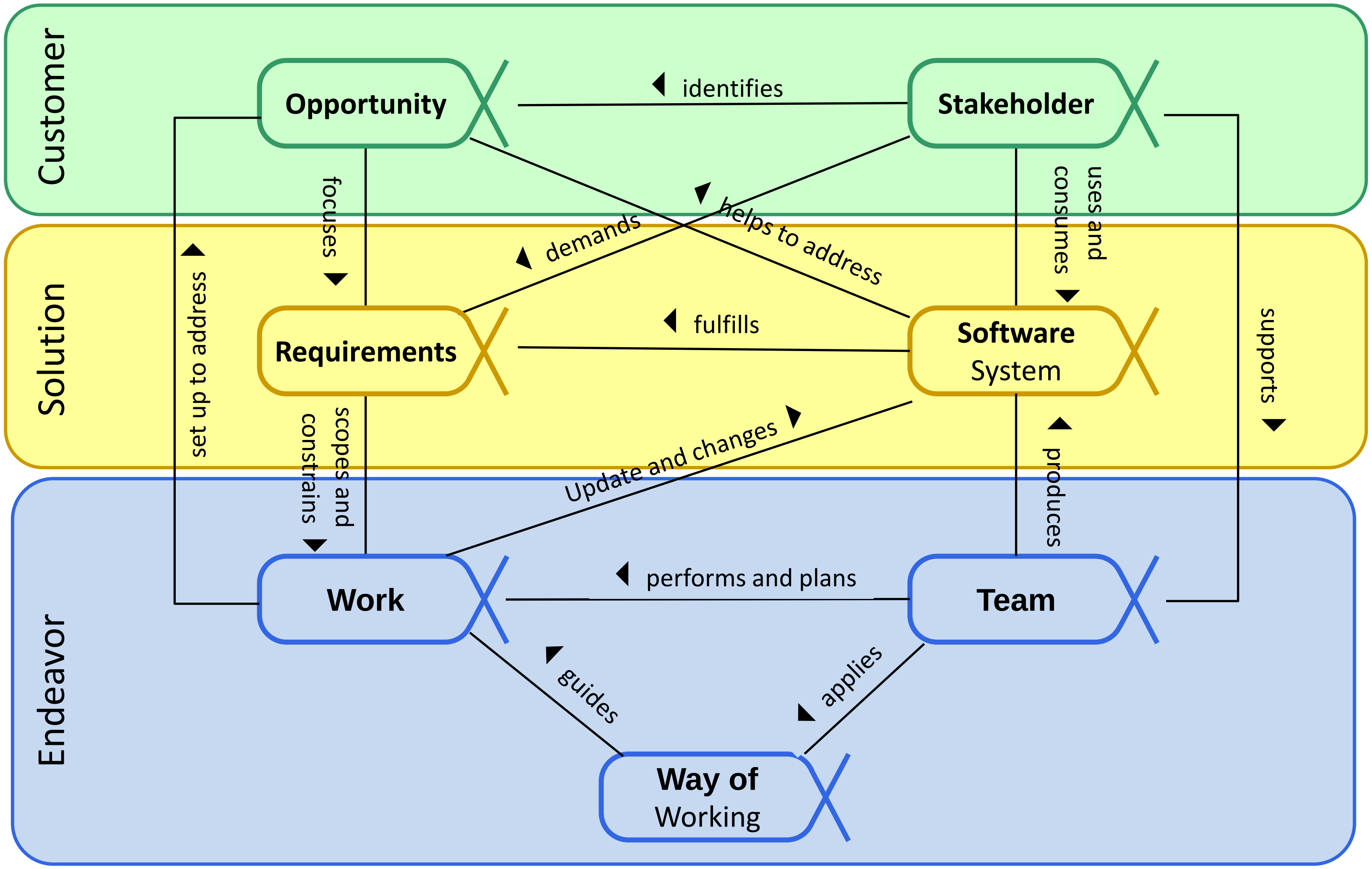
Essence describing Programming Practice

- The purpose of this practice is to produce high quality code.
  - In this case, we define code quality as being understandable by the different members of the team.
- Writing code is part of implementing the system.



1. The alphas

Essence Kernel Alphas



SEMAT

Kernel Alpha States

Opportunity
The set of circumstances that makes it appropriate to develop or change a software system.

- Identified
- Solution Needed
- Value Established
- Viable
- Addressed
- Benefit Accrued

Generated by UI Practice Workbench™ 1.1.1

Stakeholders
The people, groups, or organizations who affect or are affected by a software system.

- Recognized
- Represented
- Involved
- In Agreement
- Satisfied for Deployment
- Satisfied in Use

Generated by UI Practice Workbench™ 1.1.1

Requirements
What the software system must do to address the opportunity and satisfy the stakeholders.

- Conceived
- Bounded
- Coherent
- Acceptable
- Addressed
- Fulfilled

Generated by UI Practice Workbench™ 1.1.1

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 UI Practice Workbench™ 1.1.1

Work
Activity involving mental or physical effort done in order to achieve a result.

- Initiated
- Prepared
- Started
- Under Control
- Concluded
- Closed

Generated by UI Practice Workbench™ 1.1.1

Team
A group of people actively engaged in the development, maintenance, delivery or support of a specific software system.

- Seeded
- Formed
- Collaborating
- Performing
- Adjourned

Generated by UI Practice Workbench™ 1.1.1

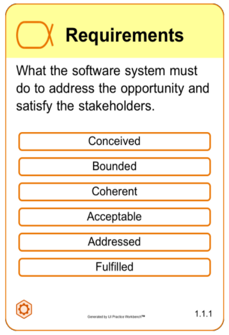
Way of Working
The tailored set of practices and tools used by a team to guide and support their work.

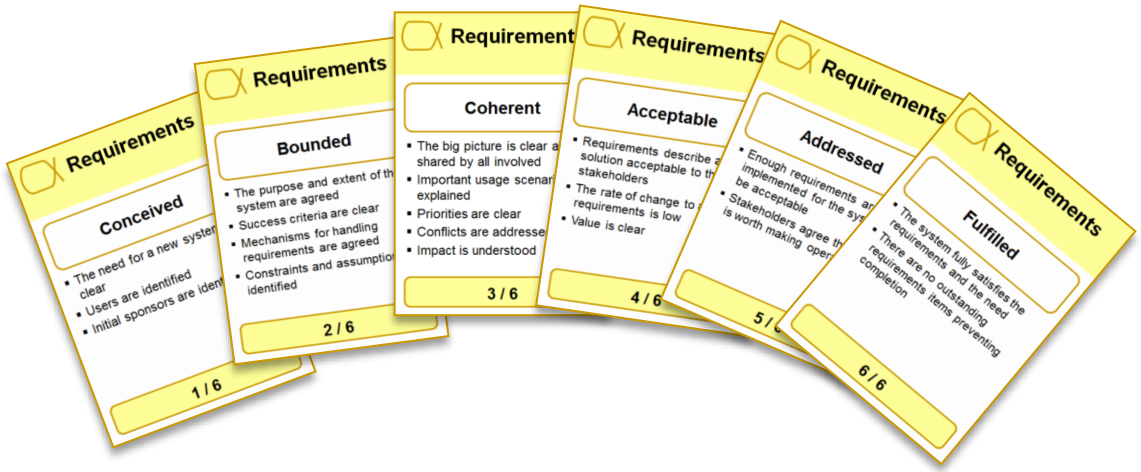
- Principles Established
- Foundation Established
- In Use
- In Place
- Working Well
- Retired

Generated by UI Practice Workbench™ 1.1.1

Cards make Kernel and Practices Tangible

- The Kernel can be "touched" and used through the use of cards
- The cards provide concise reminders and cues for team members
- By providing practical check lists and prompts, the kernel becomes something the team uses on daily basis if needed

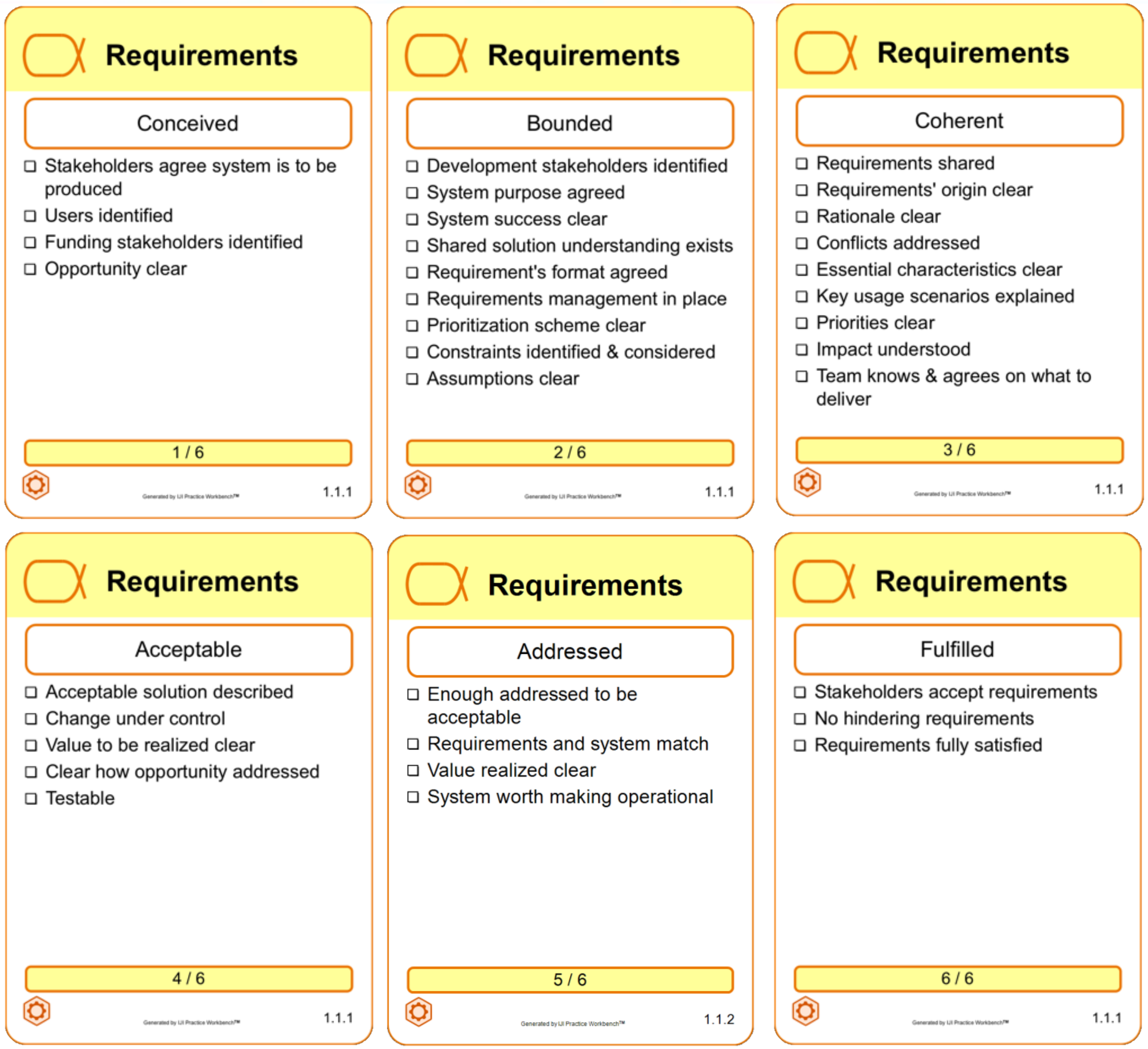

Requirements
What the software system must do to address the opportunity and satisfy the stakeholders.
Conceived
Bounded
Coherent
Acceptable
Addressed
Fulfilled
1.1.1


Requirements
Conceived
The need for a new system is clear
Users are identified
Initial sponsors are identified
1 / 6
Bounded
The purpose and extent of the system are agreed
Success criteria are clear
Mechanisms for handling requirements are agreed
Constraints and assumptions identified
2 / 6
Coherent
The big picture is clear and shared by all involved
Important usage scenarios explained
Priorities are clear
Conflicts are addressed
Impact is understood
3 / 6
Acceptable
Requirements describe a solution acceptable to the stakeholders
The rate of change to requirements is low
Value is clear
4 / 6
Addressed
Enough requirements are implemented for the system to be acceptable
Stakeholders agree the system is worth making operational
Fulfilled
The system fully satisfies the requirements and the need
There are no outstanding requirements items preventing completion
6 / 6

Alpha State Cards

For each State of an Alpha, there is a card describing the checklist criteria to achieve

- Checklists are an important and practical way to monitor and guide progress
- Checklist criteria are intentionally not expressed formally
- At the bottom of the card there is a bar indicating the sequence number and the total number of alpha states for this alpha



Requirements

Conceived

- ☐ Stakeholders agree system is to be produced
- ☐ Users identified
- ☐ Funding stakeholders identified
- ☐ Opportunity clear

1 / 6

Generated by IJI Practice Workbench™ 1.1.1

Requirements

Bounded

- ☐ Development stakeholders identified
- ☐ System purpose agreed
- ☐ System success clear
- ☐ Shared solution understanding exists
- ☐ Requirement's format agreed
- ☐ Requirements management in place
- ☐ Prioritization scheme clear
- ☐ Constraints identified & considered
- ☐ Assumptions clear

2 / 6

Generated by IJI Practice Workbench™ 1.1.1

Requirements

Coherent

- ☐ Requirements shared
- ☐ Requirements' origin clear
- ☐ Rationale clear
- ☐ Conflicts addressed
- ☐ Essential characteristics clear
- ☐ Key usage scenarios explained
- ☐ Priorities clear
- ☐ Impact understood
- ☐ Team knows & agrees on what to deliver

3 / 6

Generated by IJI Practice Workbench™ 1.1.1

Requirements

Acceptable

- ☐ Acceptable solution described
- ☐ Change under control
- ☐ Value to be realized clear
- ☐ Clear how opportunity addressed
- ☐ Testable

4 / 6

Generated by IJI Practice Workbench™ 1.1.1

Requirements

Addressed

- ☐ Enough addressed to be acceptable
- ☐ Requirements and system match
- ☐ Value realized clear
- ☐ System worth making operational

5 / 6

Generated by IJI Practice Workbench™ 1.1.2

Requirements

Fulfilled

- ☐ Stakeholders accept requirements
- ☐ No hindering requirements
- ☐ Requirements fully satisfied

6 / 6

Generated by IJI Practice Workbench™ 1.1.1

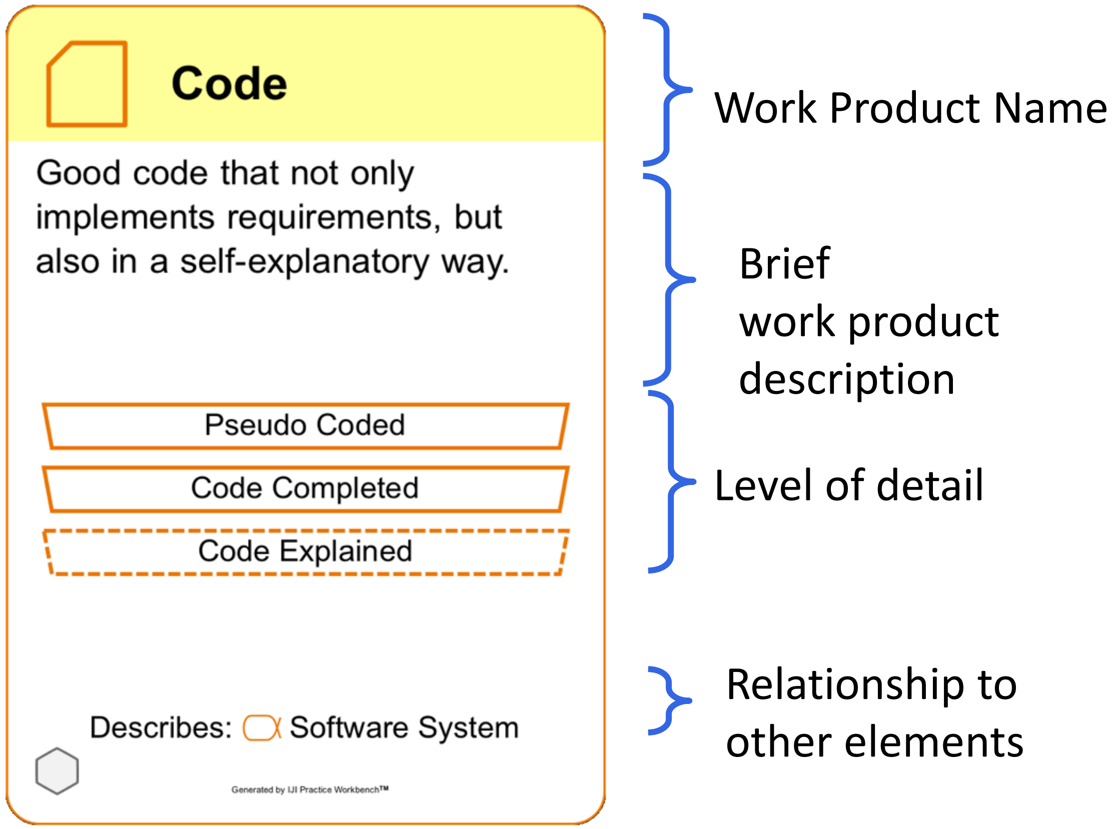
Work Products

Def: ***Work Products* are tangible things such as documents and reports**

Essence does not specify which work products are to be developed

But it does specify

- What work products are,
- How you represent them
- What you can do with them


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 IJI Practice Workbench™
Work Product Name
Brief work product description
Level of detail
Relationship to other elements

Activity and Activity Spaces

Def: *Activities* are things which practitioners do

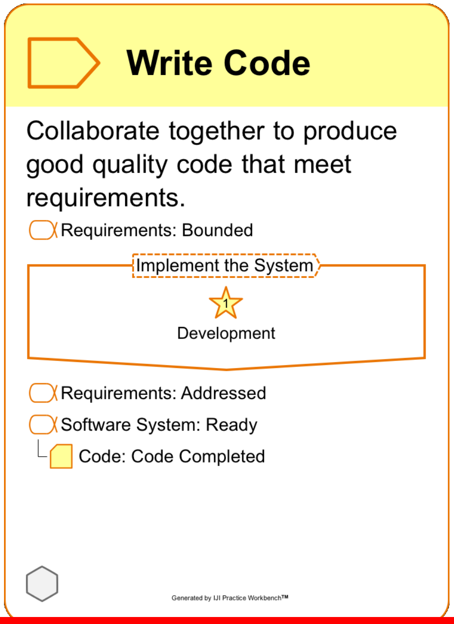
- Activities examples are: holding a meeting, analysing a requirement, writing code, testing or peer review
  - How your team goes about capturing and communicating the requirements can be very different from team to team

Def. Activity spaces are generic placeholders for specific activities

- Activity Spaces are method-independent
- Specific Activities will be added later, on top of the kernel

Essence does not define any activities

Essence defines a number of **activity spaces**


Write Code

Collaborate together to produce good quality code that meet requirements.

Requirements: Bounded

Implement the System

1

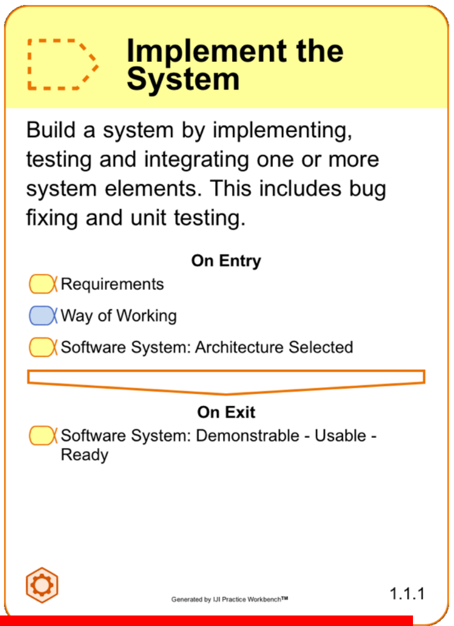
Development

Requirements: Addressed

Software System: Ready

Code: Code Completed

Generated by IJI Practice Workbench™


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

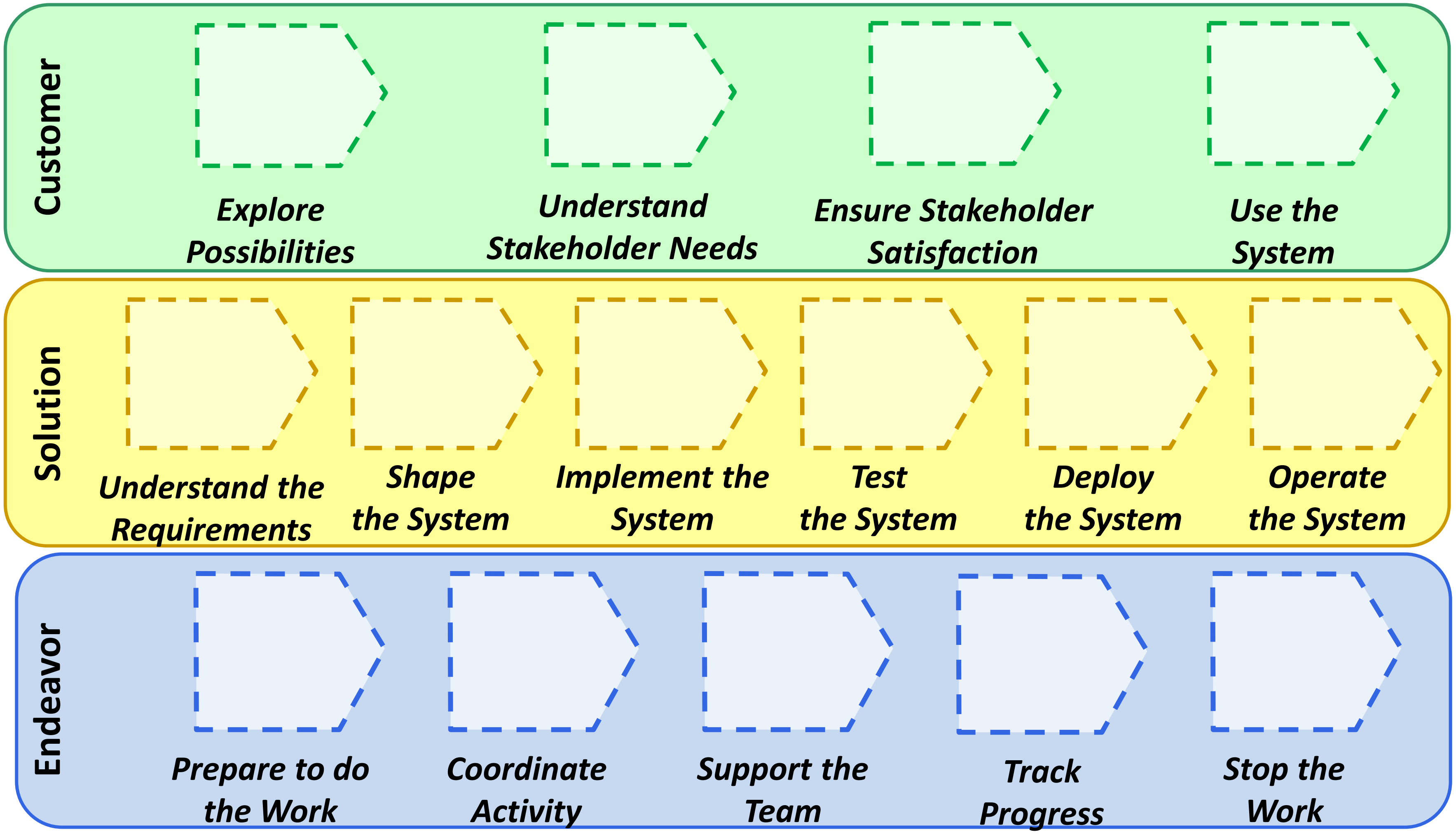
Generated by IJI Practice Workbench™

1.1.1

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

Activity Spaces in the Essence Kernel

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

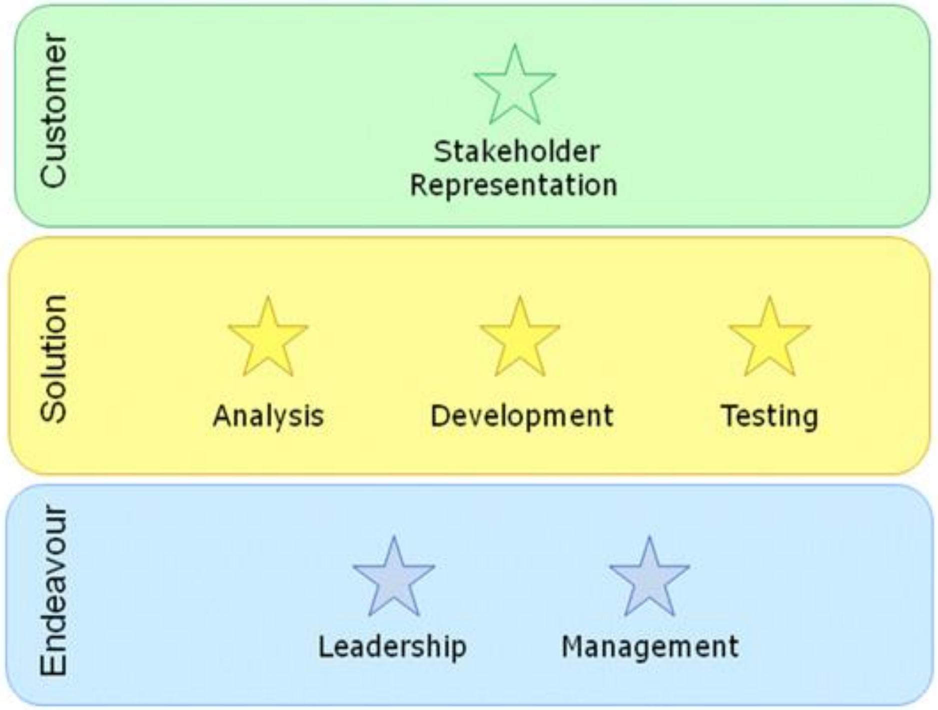
- **Understand the Requirements**
  Establish a shared understanding of what the system to be produced must do.
- **Shape the system**
  Shape the system so that it is easy to develop, change and maintain, and can cope with current and expected future demands. This includes the architecting and overall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

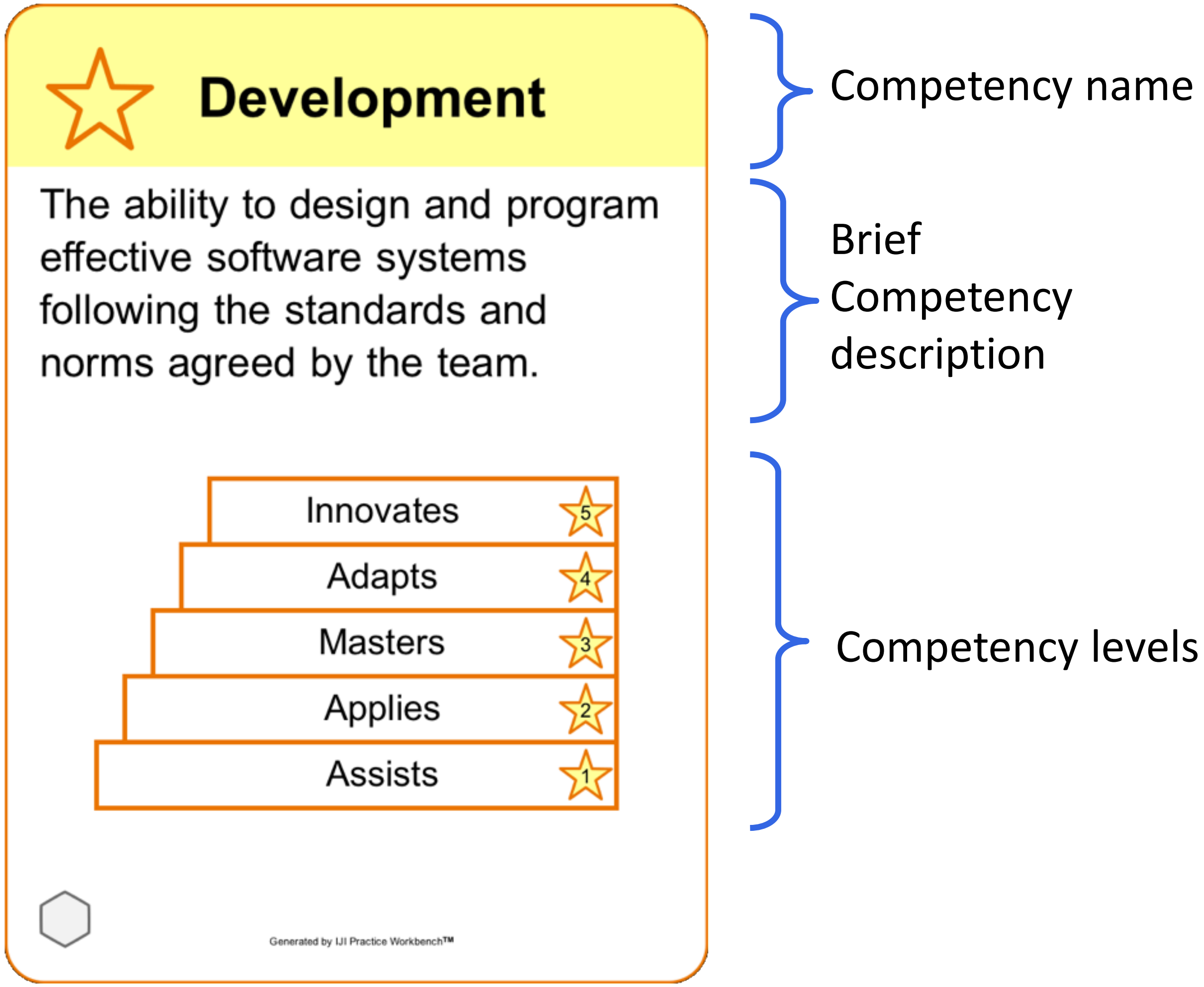
- **Prepare to do the Work**
  Set up the team and its working environment. Understand and commit to completing the work.
- **Coordinate Activity**
  Co-ordinate and direct the team's work. This includes all ongoing planning and re-planning of the work, and re-shaping of the team.
- **Support the Team**
  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.

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



Competency Card


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 IJI Practice Workbench™
Competency name
Brief Competency description
Competency levels

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.

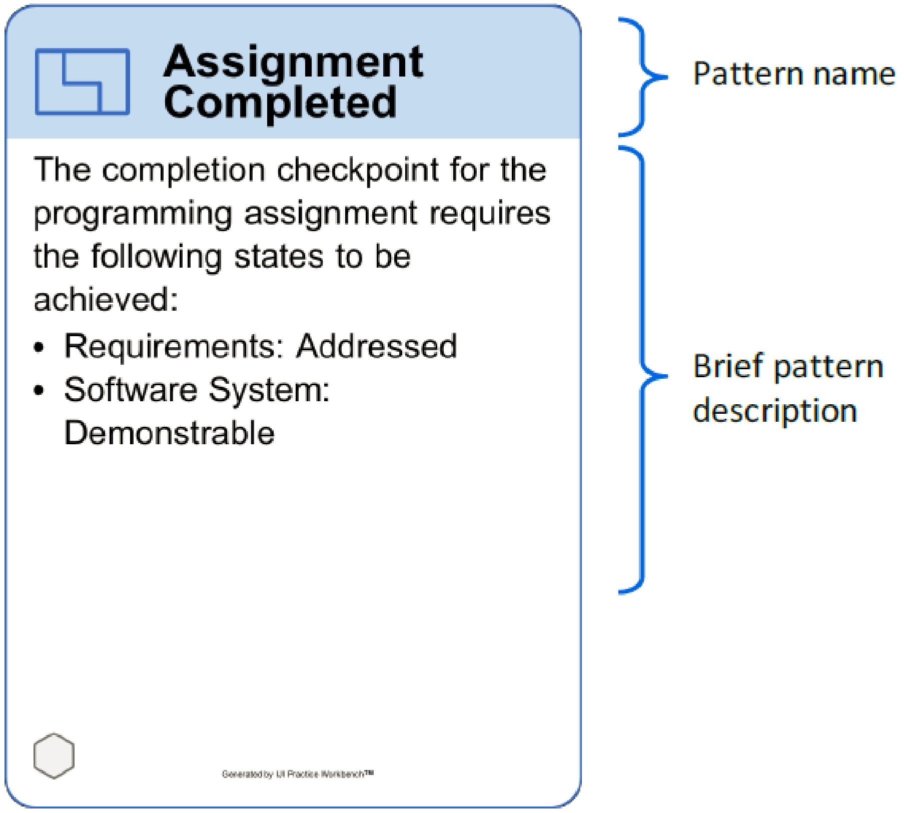
Solution

- **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

Pattern


Assignment Completed

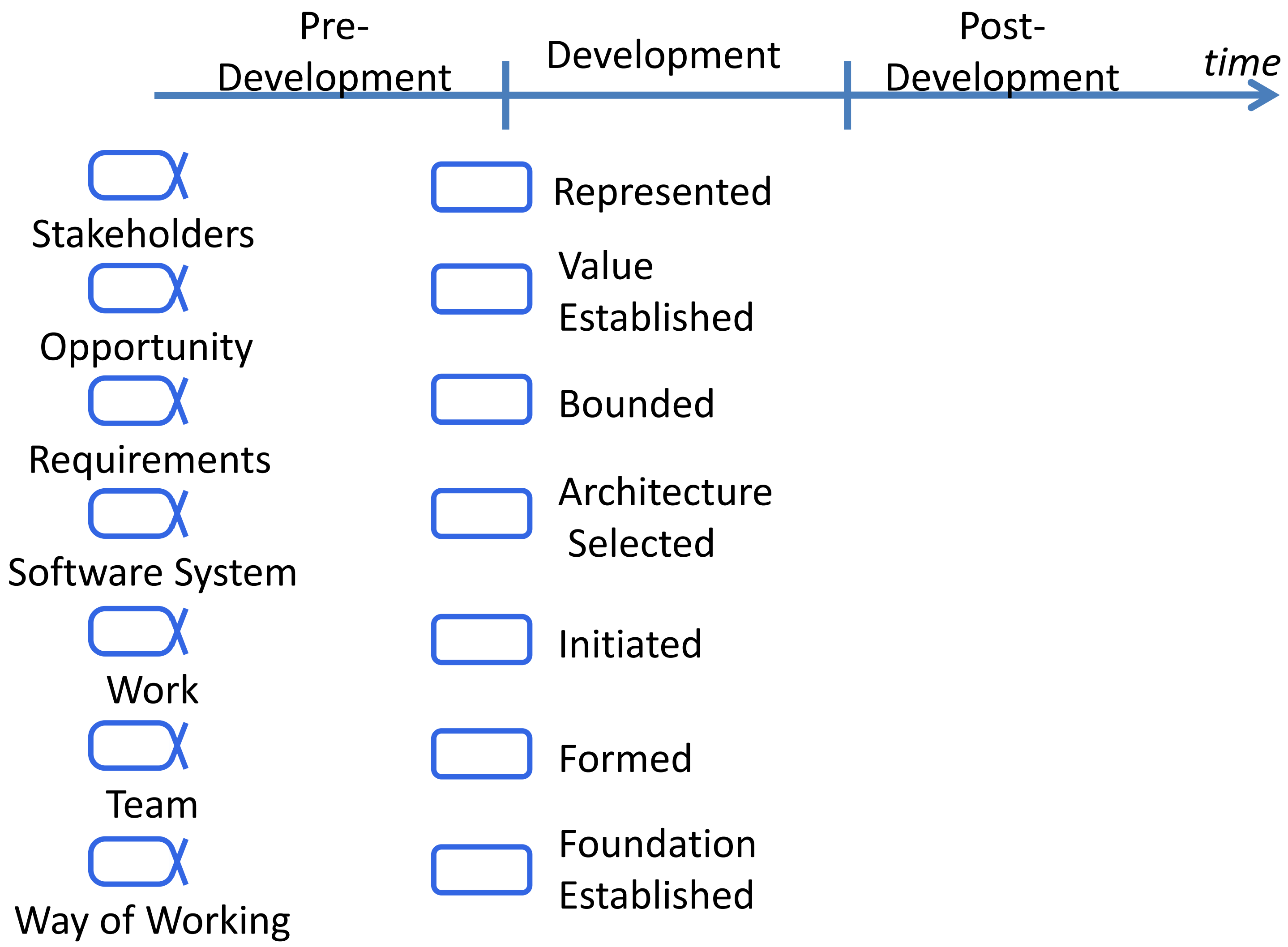
The completion checkpoint for the programming assignment requires the following states to be achieved:

- Requirements: Addressed
- Software System: Demonstrable

Generated by UI Practice Workbench™

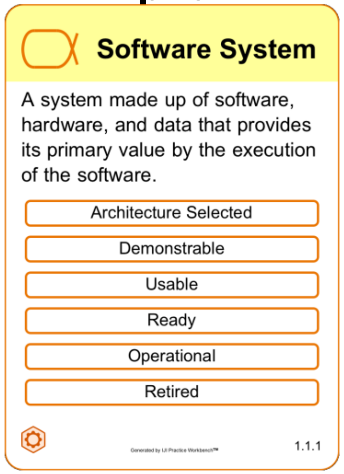
Example of Checkpoint Pattern for Enterprises

Just as an example, let's see how a large organization could approve a project start using a Check Point Pattern with further "gates"



Summary of Essence Elements and Cards

Alpha



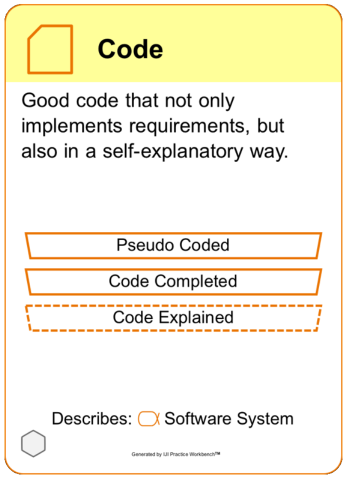
Software System

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 UI Practice Workbench™ 1.1.1

Work Product



Code

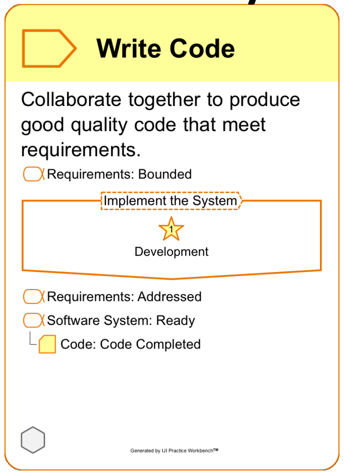
Good code that not only implements requirements, but also in a self-explanatory way.

- Pseudo Coded
- Code Completed
- Code Explained

Describes: Software System

Generated by UI Practice Workbench™

Activity



Write Code

Collaborate together to produce good quality code that meet requirements.

Requirements: Bounded

Implement the System

1 Development

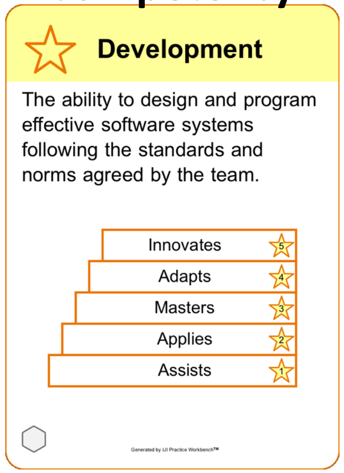
Requirements: Addressed

Software System: Ready

Code: Code Completed

Generated by UI Practice Workbench™

Competency



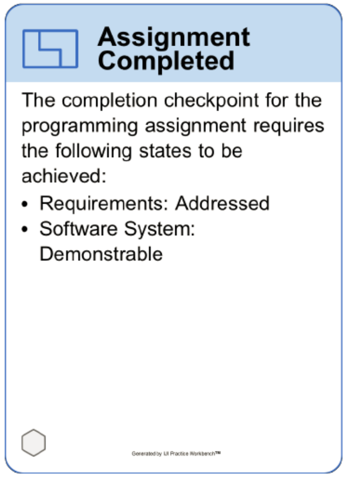
Development

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 UI Practice Workbench™

Pattern



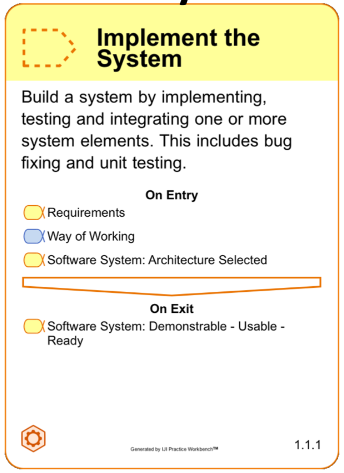
Assignment Completed

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

- Requirements: Addressed
- Software System: Demonstrable

Generated by UI 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

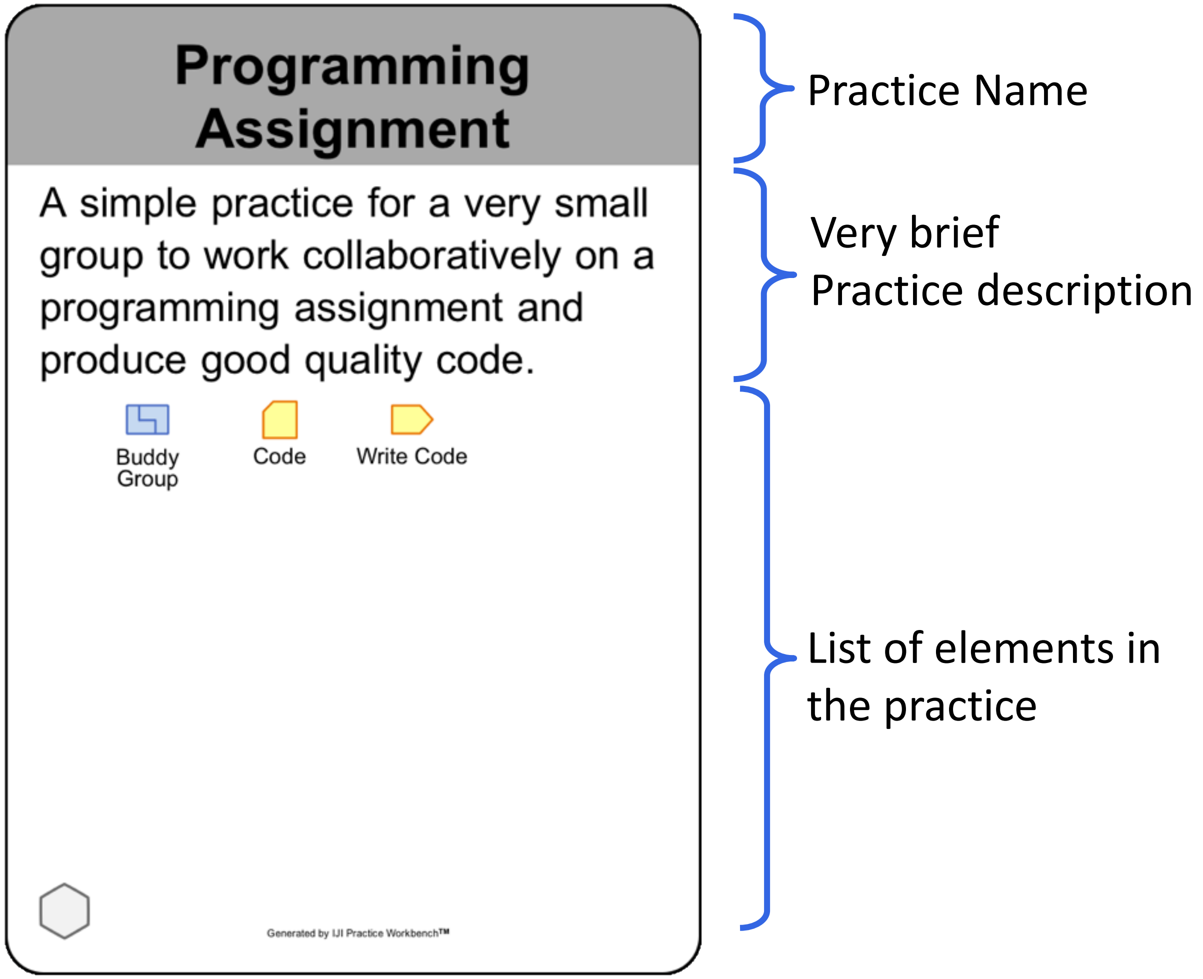
On Exit

Software System: Demonstrable - Usable - Ready

Generated by UI Practice Workbench™ 1.1.1

There is a snapshot that need to be replaced.

Practice



Programming Assignment

A simple practice for a very small group to work collaboratively on a programming assignment and produce good quality code.

Buddy Group
Code
Write Code
Generated by IJI Practice Workbench™
Practice Name
Very brief Practice description
List of elements in the practice

For Next Time

- Review Essentials Chapters 7
- Review this Lecture
- Read Essentials Chapters 8, 9 and 10
- Come to Lecture