



**Software Engineering Essentialized
Teaching material**

The *Essence* Language





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

www.semat.org

Essence Prime

- Essence provides a precise and actionable language to describe software engineering practices.
 - The constructs in the Essence language are in the form of shapes and icons.
 - The different shapes and icons each have different meaning.
- Essence categorizes the shapes and icons as:
 - Things to Work With
 - Things to Do
 - Competencies
- Essence provides explicit and actionable guidance.
 - This actionable guidance is delivered through associated checklists and guidelines.

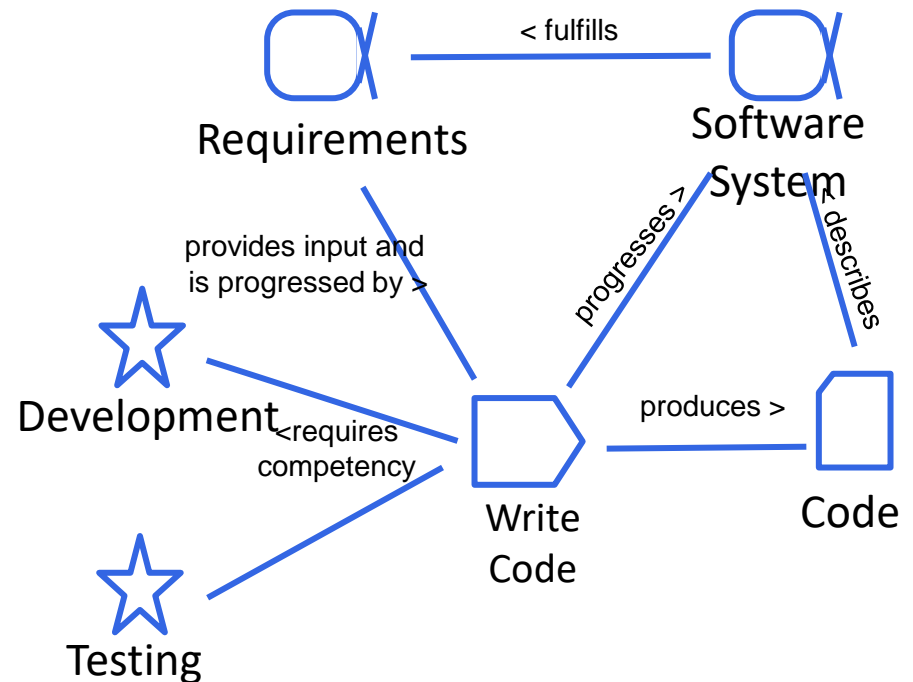
Essence Language Element Types

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.

- The Essence list is longer, but at this time we consider these elements as key and the first to learn

An Example: 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.
- Two persons (students) work in pairs to turn requirements into a software system by writing code together.
- Writing code is part of implementing the system.




Alphas

Def: ***Alphas*** are subjects in a software endeavour whose evolution we want to understand, monitor, direct, and control

- Alphas are the most important things you must attend to and progress in order to be successful in a software development endeavour
- For our programming practice example:
 - The Alphas are: Requirements and Software System
 - There will always be requirements, regardless of whether you document them or not, or how you document them, e.g. as requirement items, use cases, etc.
 - In some cases the requirements for a software system may just exist in the heads of people. However, an alpha may be made evidenced by providing one or more descriptions; that is, by attaching work products to the alpha.
- An Alpha is not tangible by itself, but it is understood or evidenced by the work product(s) that are associated with it and thus describe a particular aspect of the alpha

Alpha Card

The Alpha Card provide a short and crisp description of the Alpha and it's States

 **Requirements**

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

Conceived


Bounded

Coherent

Acceptable

Addressed

Fulfilled

 Generated by IJI Practice Workbench™ 1.1.1

Alpha Name

Very brief
alpha description

Alpha states
Each alpha state has
an alpha state card

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
 - So teams can interpret each checklist item as they deem it appropriate to their endeavour.
- 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 UI 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 UI 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 UI 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 UI 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 UI Practice Workbench™

1.1.2



Requirements

Fulfilled

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

6 / 6

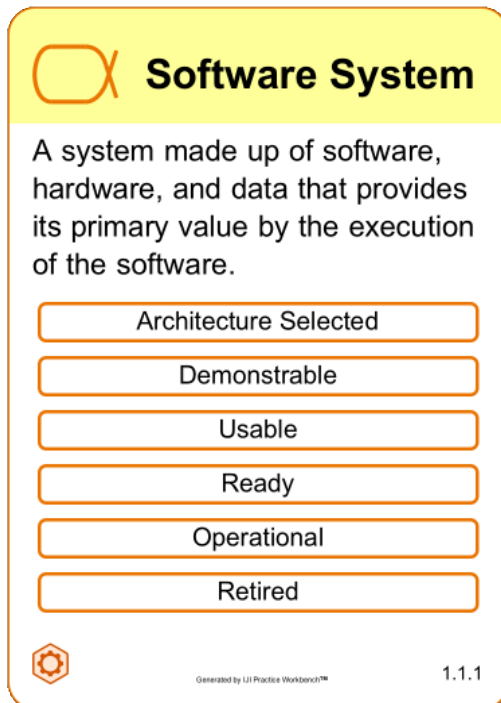


Generated by UI Practice Workbench™

1.1.1

Software System Alpha example

The Programming Practice in our example has also the Software System Alpha



The states are defined on the basis of an incremental risk driven approach to building the Software System:


- **Architecture Selected** – key decisions about the Software System have been made.
 - For instance, the most important system elements and their interfaces are agreed upon.
- **Demonstrated** – key use of the Software System has been demonstrated and agreed.
- **Useable** – the Software System is usable from the point of view of its users.
- **Ready** – the Software System has sufficient quality for deployment to production, and the production environment is ready.
- **Operational** – the Software System is operating well in the production environment.
- **Retired** – the Software System is retired and replaced by a new version of the Software System, or by a separate Software System.

Work Products

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

- Work products may provide evidence to verify the achievement of alpha states.
 - For example, when a complete and accepted requirements document has been developed that evidence can be used to confirm achieving certain checklists within a state of the Requirements alpha.
- The fact that you have a document is not necessarily a sufficient condition to prove evidence of state achievement.
 - Historically, documentation has not always provided an accurate measurement of progress.
 - It is the checklist for that state that has been achieved satisfactorily the condition to satisfy
- 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

Work Product Card


**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

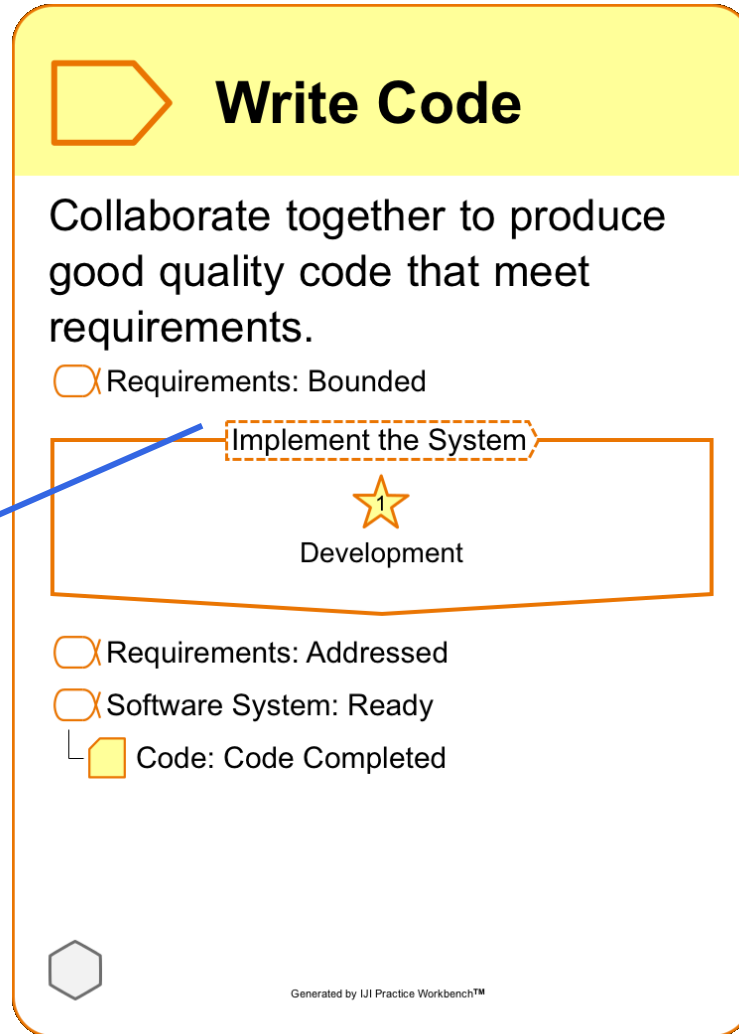
Relationship to
other elements

Activity

Def: ***Activities*** are things which practitioners do

- Activities examples are: holding a meeting, analysing a requirement, writing code, testing or peer review
- Practitioners often struggle to determine the appropriate degree of detail or formality with an activity, or exactly how to go about conducting the activity.
 - This is another motivation for explicit practices as they can provide guidance to practitioners in selecting appropriate activities as well as provide guidance in how to go about conducting each activity.
- A practice may include several activities that are specific to the practice being described.
 - Activities are specific and not standard – they are not a part of Essence.
- An activity is always bound to a specific practice, it cannot “float around” among the practices.
 - If you find an activity that needs to be reused by many practices, then you may want to create a separate practice including this activity.

Activity Card



Activity space
which this activity
belongs to


- } Activity name
- } Very brief
Activity description
- } Inputs for activity
- } Competency to
conduct activity
- } Outputs of activity

Competency






Def: ***Competencies*** are the abilities needed when applying a practice


- Often software teams struggle because they don't have all the abilities needed for the task they have been given.
 - In these situations, a coach can help by explaining different ways the practitioner can address the problem, such as learning something that is missing in their competencies.
 - A useful exercise that teams are encouraged to conduct is to do a self-assessment of their competencies and compare the results to the competencies they believe they need to accomplish their specific endeavour.

Competency Card

**Development**

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

Innovates	
Adapts	
Masters	
Applies	
Assists	



Generated by IJL Practice Workbench™

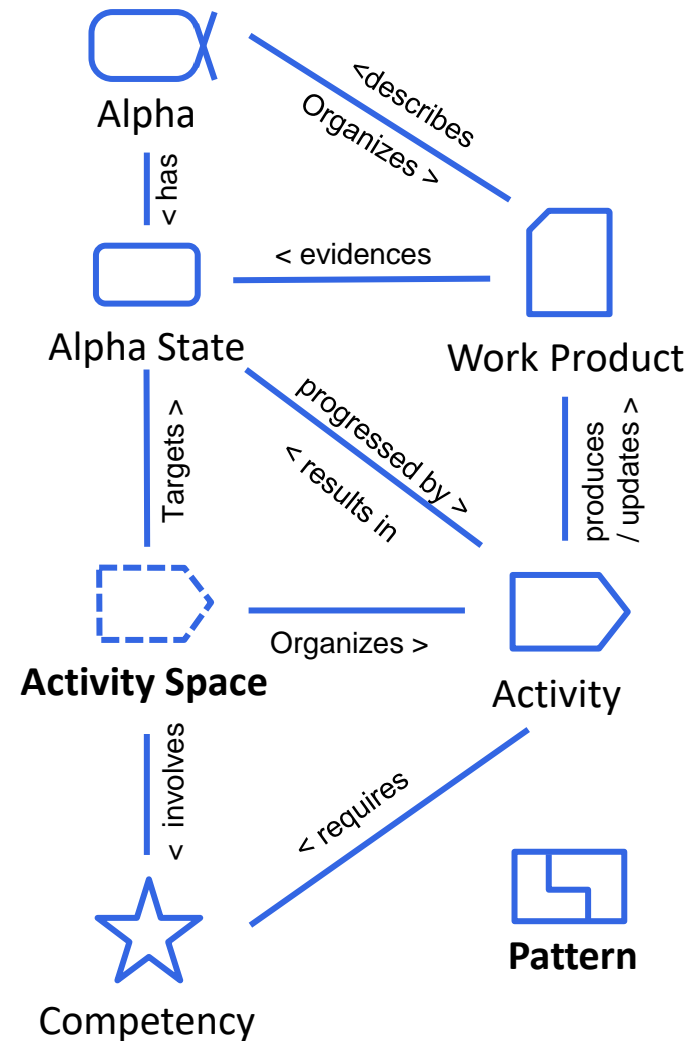
Competency name

Brief
Competency
description

Competency levels



Essence Language

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



Additional Elements in Essence Language

- The Essence Language Element list contains two more elements

Element Type	Syntax	Meaning of element type
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.

- These elements will be described deeper later on.

Essentializing a Practice

- The steps to Essentializing a practice are:
 - ***Identifying the elements*** – this is primarily identifying a list of elements that make up a practice.
 - The output is essentially a diagram like that one seen for the Programming Practice
 - **Drafting the relationships** between the elements and the outline of each element
 - At this point, the cards are created
 - **Providing further details** – Usually, the cards will be supplemented with additional guidelines, hints and tips, examples, and references to other resources, such as articles and books.

Please Note: Alphas vs Products difference

Essence distinguishes between elements of health and progress versus elements of documentation.

- Elements of health and progress: Alphas
 - Alphas are the important things we work with when conducting software engineering activities.
 - Alphas are not work products.
 - Alphas are things we want to track the progress of.
- Elements of documentation: products.
 - Work products are tangible things such as documents, which can have *different levels of detail*.

For Next Time

- Review Essentials Chapter 5
- Review this Lecture
- Read Essentials Chapter 6
- Come to Lecture