
**Software Engineering Essentialized**
**Teaching material**

# Introduction to *Essence*

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

www.semat.org

# The Essence of our goal

1. Methods are compositions of practices
   - A practice, that is like a mini-method, is a reusable approach to a well defined problem
   - Practice examples: Requirements Management, Agile Development, Use Case Modeling, etc.
2. There is a common ground – Kernel- shared by all methods and practices
   - A common vocabulary
     - It makes easier to teach, learn, use and modify practices
     - Is a necessity to create a library of reusable practices from which selecting
3. Focus on the essentials when providing guidelines for a method or practice
   - Developers rarely have the time to read detailed methods and practices
   - The essential are defined as the initial minimum of what expert knows, but enough to start practicing. 5% could be enough and provide the idea that is really the essence of the whole.

# Essence

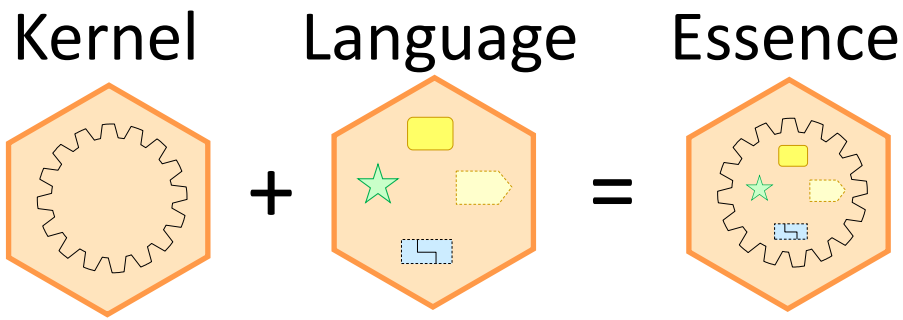
- The Essence initiative come to light very recently to address the situation described above

- **Essence is the kernel of a Software Engineering Theory as well as the language to describe such theory and the approach to describe methods and practices based on the theory**

- As described in the Appendix, this theory is not yet fully shaped and Essence, though the first and most promising kernel of Software Engineering Theory, will need to evolve to become a fully fledged theory.

# The Essence of "Essence"

- Essence is made of 2 parts:
  - Kernel
    - The kernel of Software Engineering
      - The set of elements that would always be found in all types of software system endeavours
  - Language
    - The Essence language is very simple, intuitive and practical
    - Utilized in describing the Essence kernel with the elements that constitute a common ground



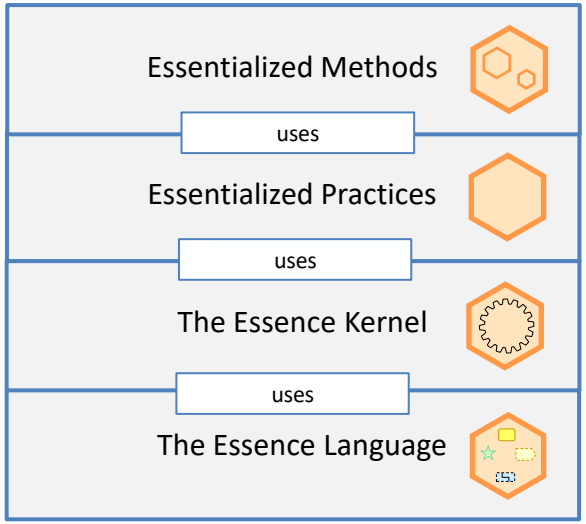
# Essentialize Practices and Methods

- Using Essence Kernel and Language you can Essentialize Practices and Methods
- Essentialize a practices means they are described using Essence
  - the Essence kernel
  - the Essence language
  - It focuses the description of the method/practice on what is essential.
- Consequently, the methods we describe are also essentialized.

# Essence Method Architecture

- **Essentialized Methods** are composition of **Essentialized practices**

- Practices can be compositions of smaller practices.
  - Scrum for instance can be seen as a composition of three smaller practices:
    - Daily Stand-up,
    - Backlog-Driven Development
    - Retrospective.



# Methods are composition of Practices

- Composition of practices is an operation merging two or more practices to form a method.
  - The operation has been defined mathematically in order to be precise.
  - The operation has to be specified by an expert with the objective to resolve potential overlaps and conflicts between the practices concerned, if there are any.
  - Usually most practices can be composed easily by setting them side by side because there are no overlaps and conflicts, but in some cases these have to be taken care of.

# Resolving overlaps and conflicts

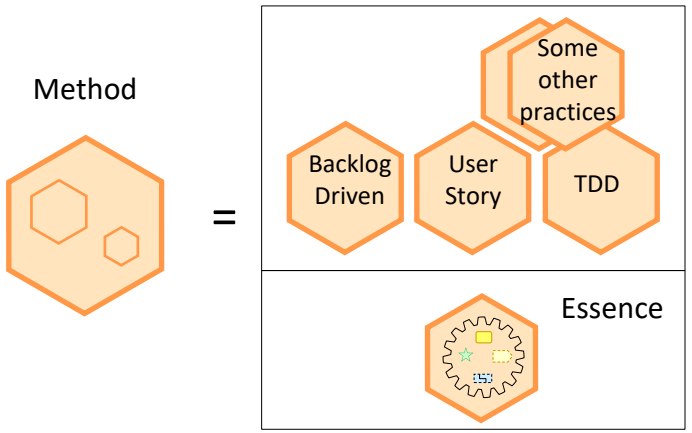
- While practices are separate, they are not independent
  - They are not like components which have interfaces over which communication will happen.
- Practices can share elements

  For example:
  - Guidelines for activities that a user (e.g. a developer) is supposed to perform
  - Guidelines for work products (e.g. components) that a user is expected to produce.
- If two practices share the same work product:
  - They contribute separate guidelines to this work product
  - Composing these two practices will require that you specify how the contributions must be combined in a meaningful and constructive way.

# How to learn a Method

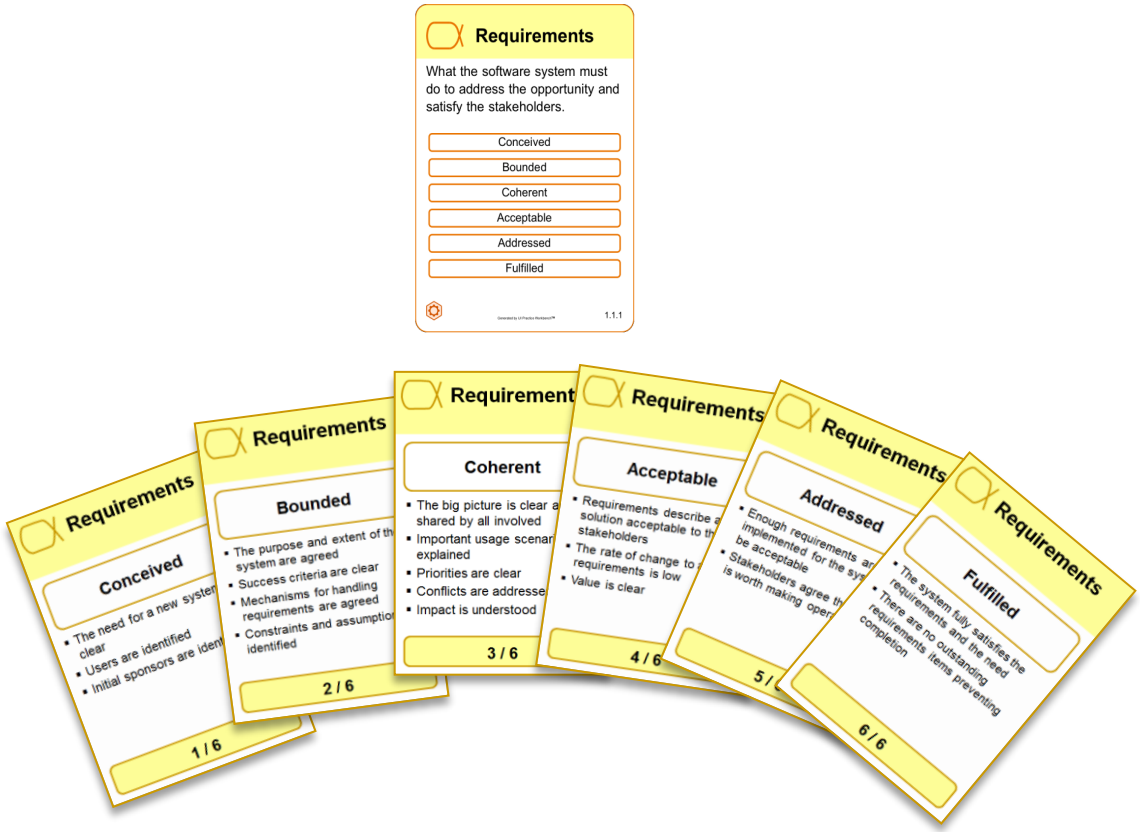
- Essence, beside text books, wants to provide team members with a new engaging and hands-on experience to learn the tailored method that each organization will define
  - A set of icons will help represent the elements
  - A set of cards will help describe and discuss the elements

- To build a method, a team start with the kernel and selects a number of practices and tools to make up its way-of-working



# 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




**Software Engineering Essentialized**
**Teaching material**

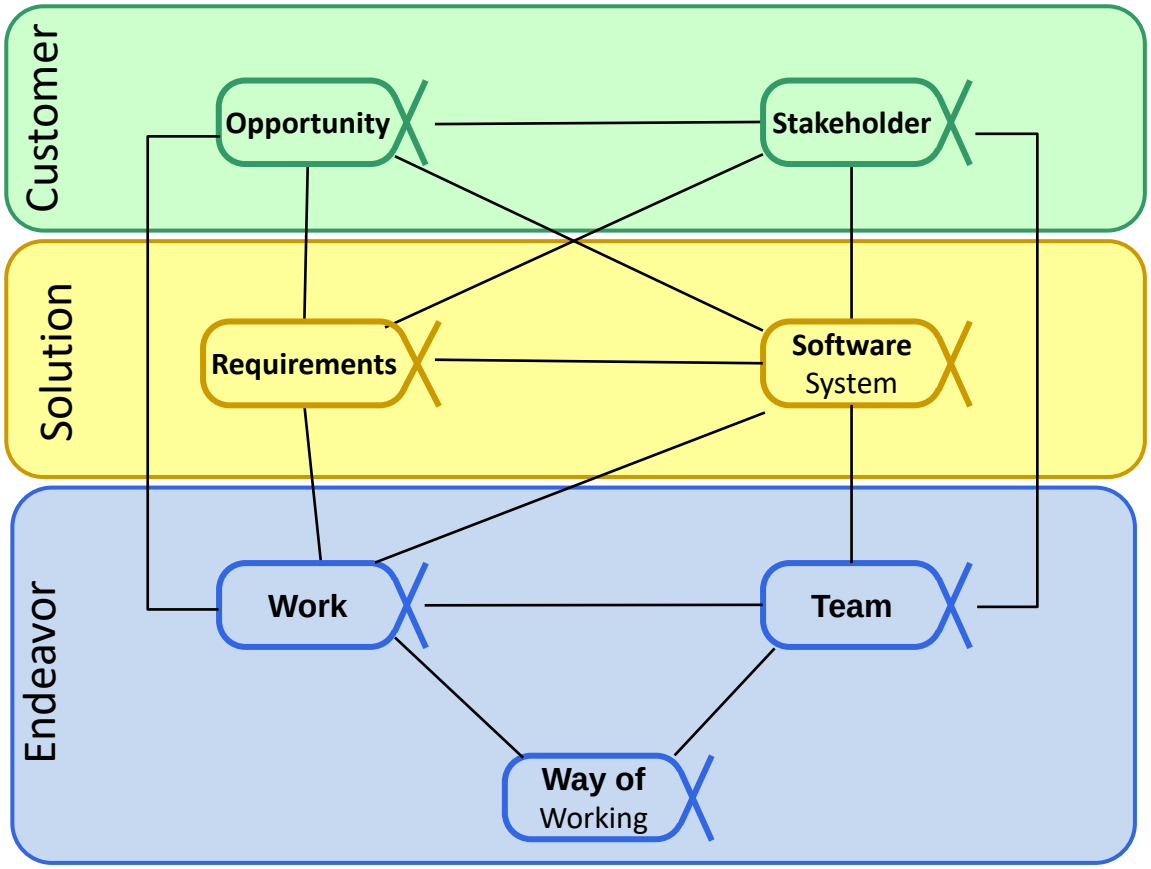
# The Basics of Software Engineering

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

www.semat.org

# Software Engineering Basics

- The commonly used terms in Software Engineering we must know before drilling down on building methods are organized around three areas
- Customer
  - Opportunity
  - Stakeholder
- Solution
  - Requirements
  - Software System
- Endeavour
  - Work
  - Team
  - Way of Working



# Customer

Customers: Users of our system or people that are purchasing this system for the users

**Software Engineering is about providing value to customers**

- Opportunity
  - An opportunity is a chance to do  something to provide value to customers, including fixing an existing problem via this software system
- Stakeholders
  - Stakeholders are individuals, organizations or groups that have some interest or concern either in the system to be developed or in its development

# Solution

**The solution is the outcome of this endeavor**

- Requirements
  - Requirements provide the stakeholder view of what they expect the software system to provide
    - They indicate what the software system must do, but do not explicitly express how it must do it
    - Among the biggest challenges software teams faces are changing requirements
- Software System
  - The primary outcome of a software endeavour is of course the software system itself.
  - 3 important characteristics of software systems
    - Functionality – Must serve some function
    - Quality – Reliability, Performance, Rich user experience, etc.
    - Extensibility – From version to version and platform to platform

# Endeavors

**An endeavor is any action that we take to achieve an objective**

- Team
  - Team must have enough people (and not too much), with right skill mix, work collaboratively, and adapting to changing environments
  - Good team working is essential
- Work
  - The work of bringing the opportunity to reality
  - Effort and Time are the most important measures of the work
    - Effort and Time are limited
    - The idea is to get things done fast but with high quality
- Way of Working
  - Team members must agree on their way of working
    - The practice and tools that will be used
    - Used by all team members
    - Improved by the team when needed
  - One of the things we hope to achieve with Essence is simplifying the process of reaching a common agreement, that is always a major challenge

# For Next Time

- Review Essentials Chapters 3 and 4
- Review this Lecture
- Read Essentials Chapter 5
- Come to Lecture