


# Essence
# *Software Engineering Essentialized*
# Part 3 – Adopting Practices & Conclusions

*Giuseppe Calavaro, Ph.D.*

# Agenda of this Teaching Module

- Practice Composition
  - Composition Rules
  - Travel Essence Practice Composition
  - Travel Essence Alpha Composition
- Use of Essentialized Practices
- Leveraging Your Essence Skills
- Powering Practices through Essentialization
- Recommended Additional Reading

Example of using Microservices Lite Practice on our project are in green boxes

# Adopting Practices

- When adopting practices, teams do not need to adopt all practices at the same time. Instead they do so one at a time.
- In fact, this was how Smith's team learned and grew.
  - They started with the kernel. Then they added Scrum Lite to improve their collaboration.
  - They tried User Story Lite, but decided to go for Use Case Lite because it gave them a better requirements structure.
  - They then chose Microservice Lite to help make their software system rapidly deployable.
- Each time they added a practice, they received explicit know-how. Their knowledge about software engineering grew.

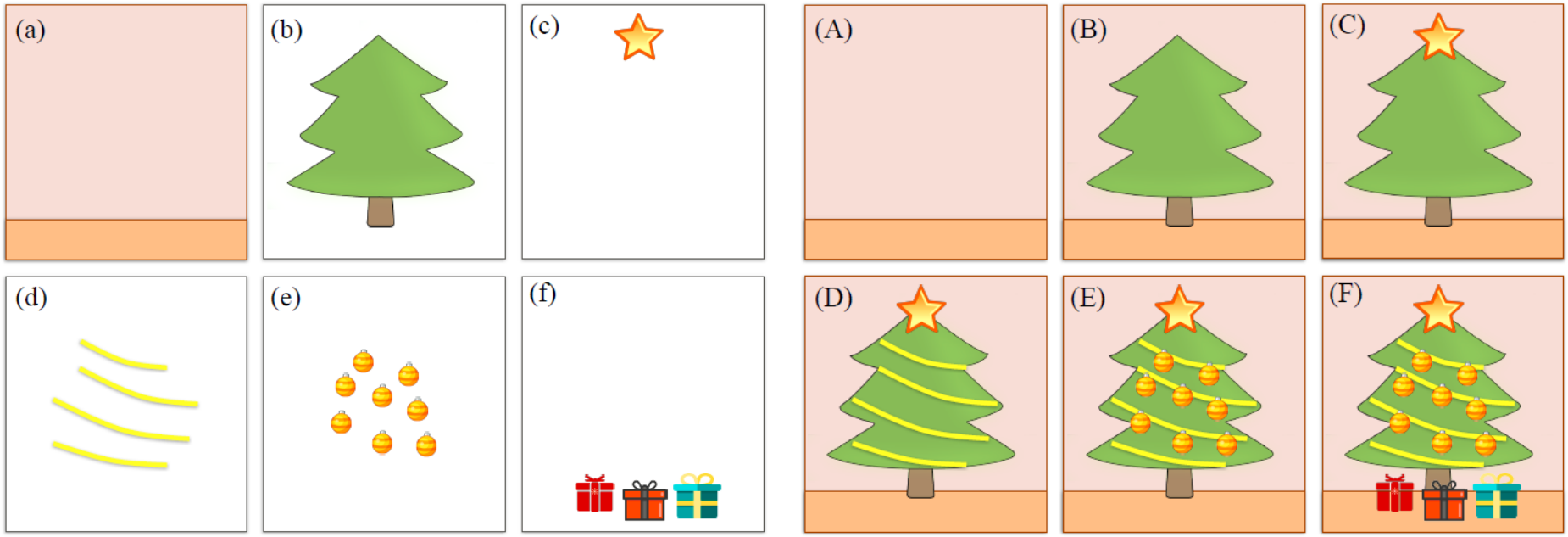
# Putting Together Practices: Composition

- Composition of practices is an operation merging two or more practices to form a method.
  - The composition operation has been defined mathematically in Essence to achieve precision.
  - The precision is achieved through the essentialization of practices
    - This means, describing practices using
      - alphas,
      - work products,
      - activity spaces,
      - activities,
      - competencies,
      - patterns.

# Composition

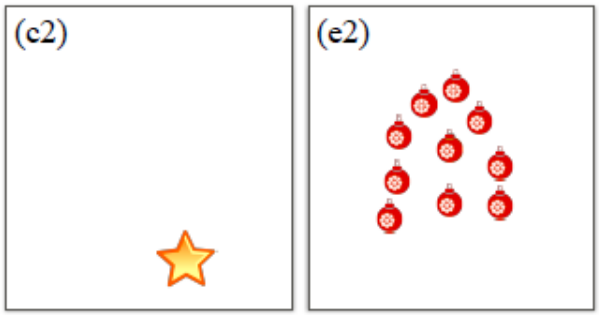
Composition is a merging operation that can be understood intuitively like overlaying overhead projector transparencies on top of one another to form a new image as shown

Single Transparencies → Composition → New Images

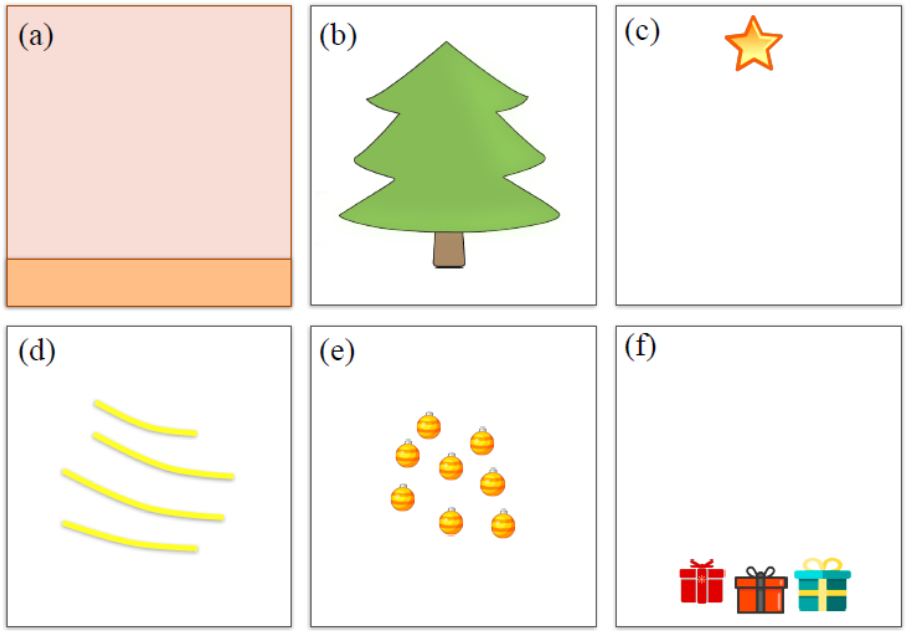


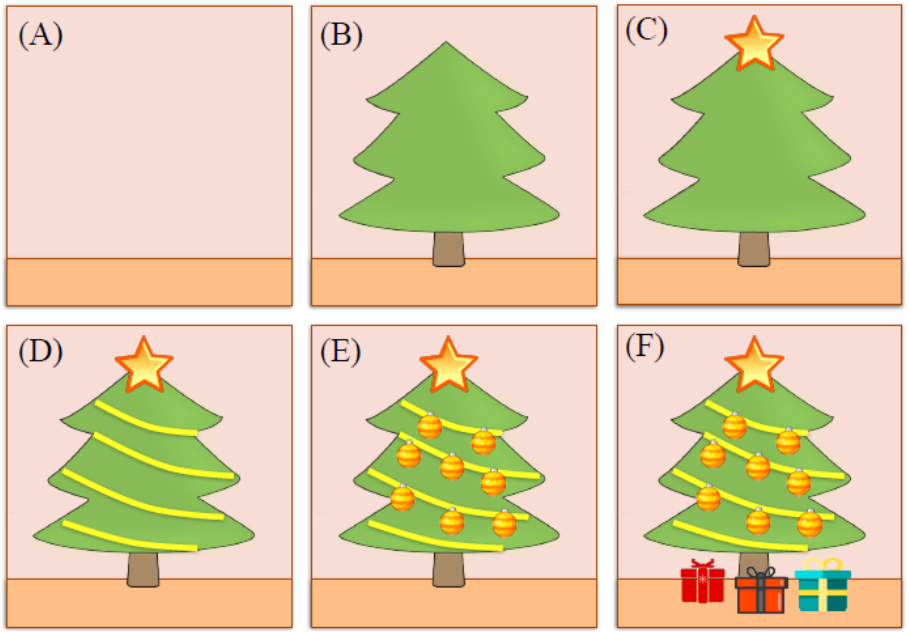
# Composition

What would happen if you replace transparency (c) and (e) with transparencies (c2) and (e2)?




Single Transparencies → Composition → New Images

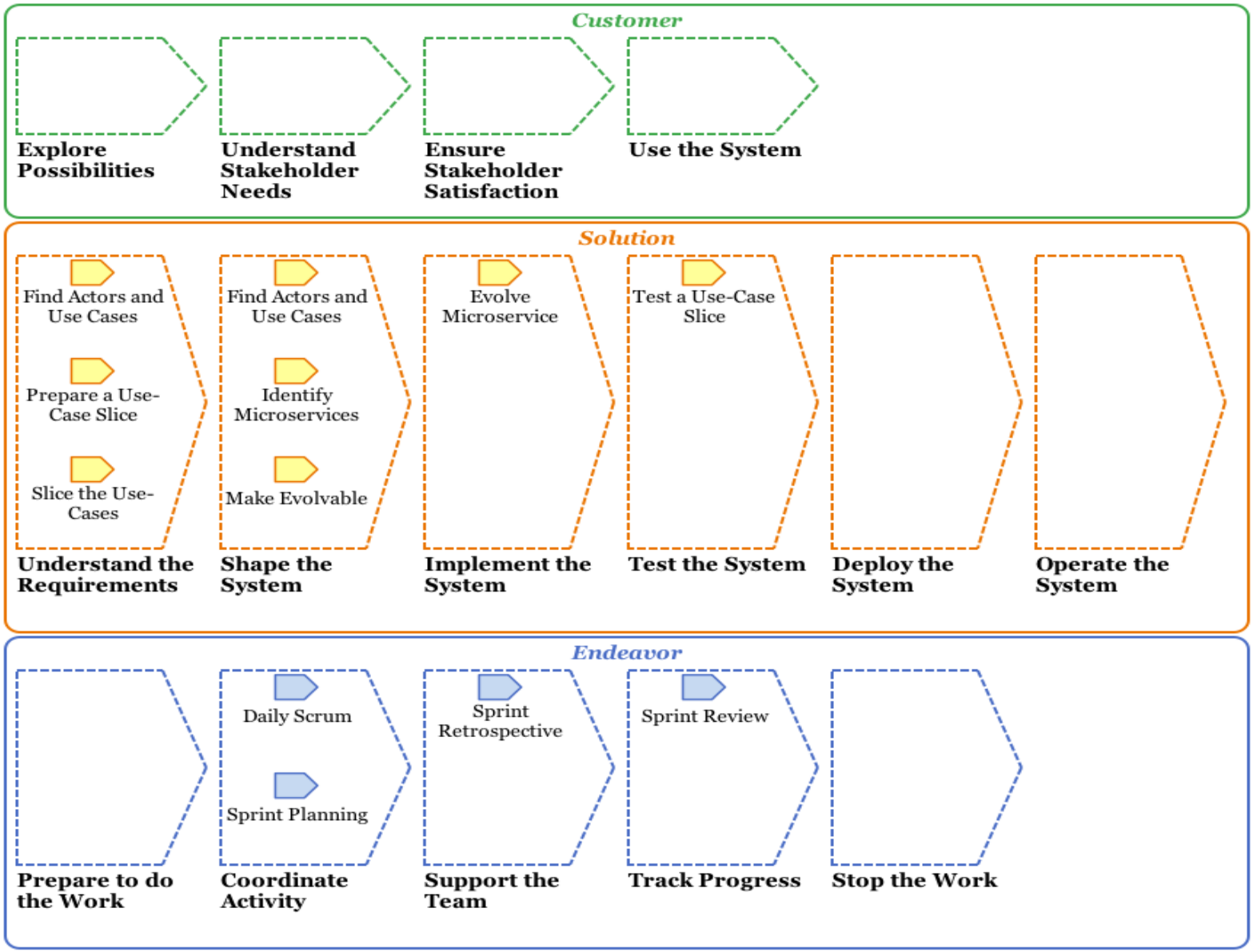


# Composition Rules

- The previous example shows an important concepts behind valid composition:
- Although the transparencies are physically separate, they are drawn such that the coordinates must match, otherwise you would have the situation where the Christmas star appears on the floor.
  - The Christmas tree anchors the position of all other images.
- The corollary is that **while practices are separate, they are not independent**.
  - **They are dependent on the namespace (e.g. kernel alphas and activity spaces) by which they are composed.**
- The **Essence kernel defines such a namespace**. It acts as the unifying anchor.
- As long as the namespace is adhered to, practices for the same purpose can substitute one for another.
  - In the example above, we can swap the transparency (c) with (c2) and get red balls instead of yellow balls as Christmas decorations.
- Theoretically, you can substitute one sub-alpha with another,
  - for example, substituting the User Story alpha with the Use Case alpha under the Requirements alpha.
- In this case, the Requirements alpha acts as an anchor.
  - However, **when you replace an alpha, you will likely need to replace activities as well**.

# Travel Essence Practice Composition

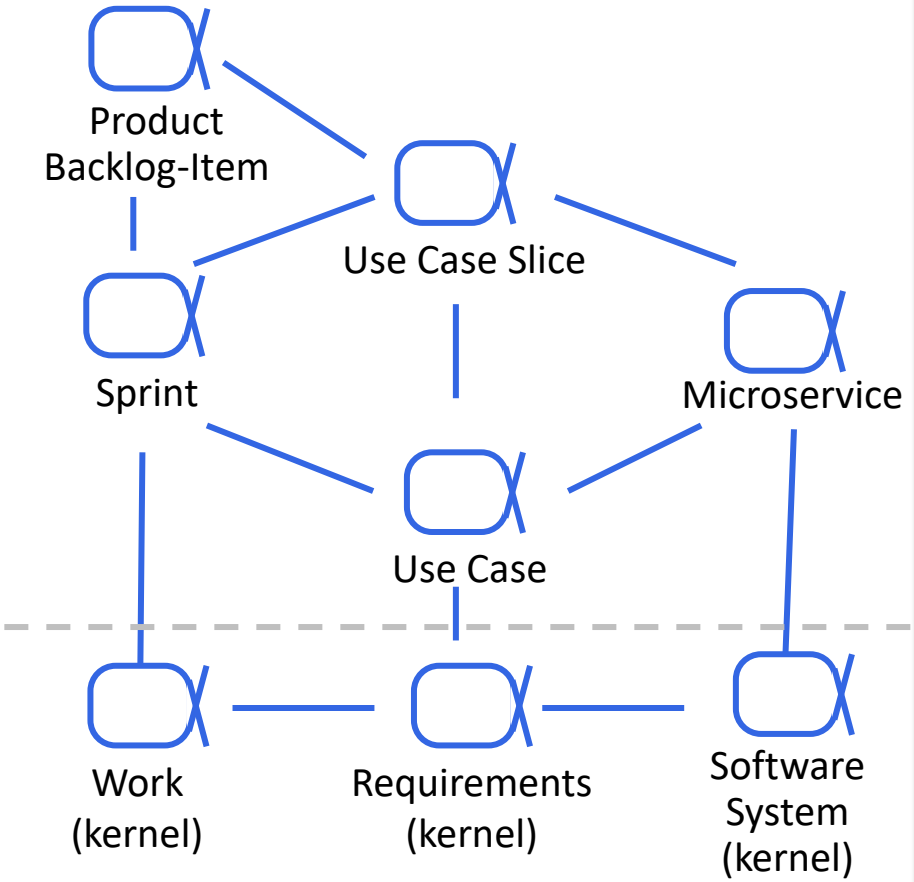


# Missing practices and project evolution

- It is quite clear that a number of activity spaces are not populated, in particular all activity spaces in the customer area of concerns.
  - Not all activity spaces in the solution and endeavour activity spaces are covered.
  - At this point in time, Smith's team was shielded from these challenges because Angela the Product Owner had been dealing with the opportunity and external stakeholder aspects, and Smith's team was focusing on internal stakeholders.
  - In subsequent releases, when Smith's team needs to deal with larger user groups, they would need practices to fill the gap
    - i.e. the gaps in the Deploy the System and Operate the System activity spaces

# Travel Essence Alpha Composition

- The composition of things to work with is a little more complicated.
  - The complication lies in the relationships between alphas.
- The kernel alphas **Work**, **Requirements** and **Software System** provides the top-level structure under which sub-alphas are group
  - The composition operation is like hanging decorations on a Christmas tree.
  - The kernel acts like the Christmas tree by which you hang the decorations, which in this case are the sub-alphas **Sprint**, **Use case** and **Microservice** respectively



# Use of Essentialized Practices

Essence helps teams apply practices effectively in multiple ways:

1. Practices described on top of Essence gives practical guidance visible to development teams.
2. Essentialization help teams translate agreed to principles into practices.
3. Essentialization supports regular team feedback and adaptation through its simple graphical language.
4. Practices described on top of Essence keeps the relationship to kernel alphas and states visible across all essential dimensions of software engineering including Opportunity, Stakeholders, Requirements, Software System, Team, Work, and Way of Working
5. The kernel serves as a reminder, and substitute for practices that may fall outside the current scope of interest.

**The Essence kernel serves as a practice agnostic lens that the team uses to continually be mindful of the progress and health of their overall endeavour**

# Leveraging Your Essence Skills

- Now that you know what a practice comprises, **you will know what to look for when being introduced to a practice new to you**.
- As an example, assume you are introduced to a new practice named Behavior Driven Development (BDD), which we will not explain.
  - By now, you know that a practice addresses a specific challenge in software engineering.
  - You also know that a practice comprises
    - alphas (and their states),
    - work products (and their level of details),
    - activities and
    - patterns.
  - Thus, when trying to learn about BDD, you will be thinking about these same factors. **Once you understand the elements of the practice, you can understand how it fits and contributes to your software engineering endeavour**.
- It is from this discovery that you will get a sound understanding of the practice and thereby achieve mastery.

# Powering Practices through Essentialization

The value of Essence to teams are twofold:

1. serving as a lens from which they can evaluate the progress and health of their software engineering endeavour, and
2. making practices explicit through essentialization.

Rather than a vague concept, Essence make practices concrete and actionable by calling out the alphas and work products

- **The explicit structure of each practice supported by the use of the Essence language provides the skeleton by which additional team members can contribute their experiences.**
  - For example, they can update the checklists in the cards, add other hints and tips to any practice elements.
- Very important to note that the Essence language has been **intentionally developed with the goal of being easily used by practitioners, and not just method development professionals.**

# Recommended Additional Reading

For a deep dive on the presented practices, we recommend that you refer to more authoritative sources.


[14] Sutherland, Jeff, Schwaber, Ken. "The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game." Scrum.org. 2016. This is the latest version of the Scrum Guide (at the time of the initial publication of this book) and should be read by anyone interested in learning more specific details on the "official" rules of Scrum.

[15] Derby, Ester, Larsen, Diana. "Agile Retrospectives: Making Good Teams Great," The Pragmatic Programmer, 2006

[16] Beck, Kent. "Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000. This book should be read by those interested in learning more about Extreme Programming.

[17] Cohn, Mike. "User stories applied: For agile software development." Addison-Wesley Professional. 2004. This book should be read by those interested in learning more about the User story practice.

[18] Bittner, Kurt, Spence, Ian. "Use Case Modeling". Addison-Wesley Professional, 2003. This book provides a comprehensive treatment of Use Case Modeling. Though the book was written before agile became the trend, it is still a valuable treatment of the subject.

[19] Booch, Grady, Rumbaugh, James, Jacobson, Ivar, "The Unified Modeling Language User Guide (Second Edition), Addison-Wesley, 2005. This book should be read by anyone interested in learning more about the Unified Modeling Language.

[20] Jacobson, Ivar, Spence, Ian, Bittner, Kurt, "Use Case 2.0: The Guide to Succeeding with Use Cases, December, 2011, https://www.ivarjacobson.com/publications/whitepapers/use-case-ebook This ebook provides detailed guidance in applying Use Case 2.0 and should be read by anyone interested in learning more about Use Case 2.0.

[21] Jacobson, Ivar, Spence, Ian, Kerr, Brian. "Use-Case 2.0." Communications of the ACM 59, no. 5, 61-69. 2016. This article is recommended for those interested in learning more about Use-case 2.0, and comparison of user story and use-case practices.

[22] Newman, Sam. "Building microservices". O'Reilly Media, Inc. 2015. This book should be read by anyone interested in gaining a more in depth understanding of the microservices practice.