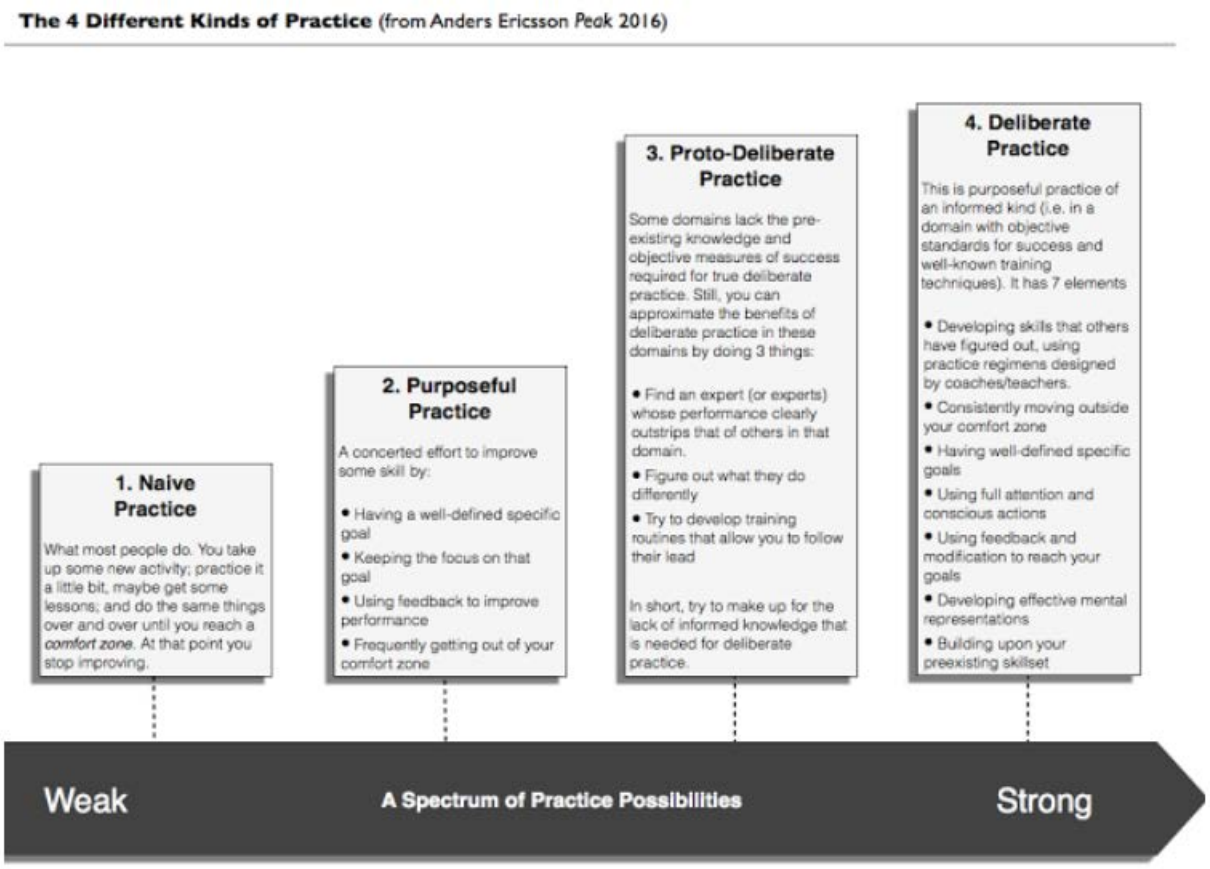
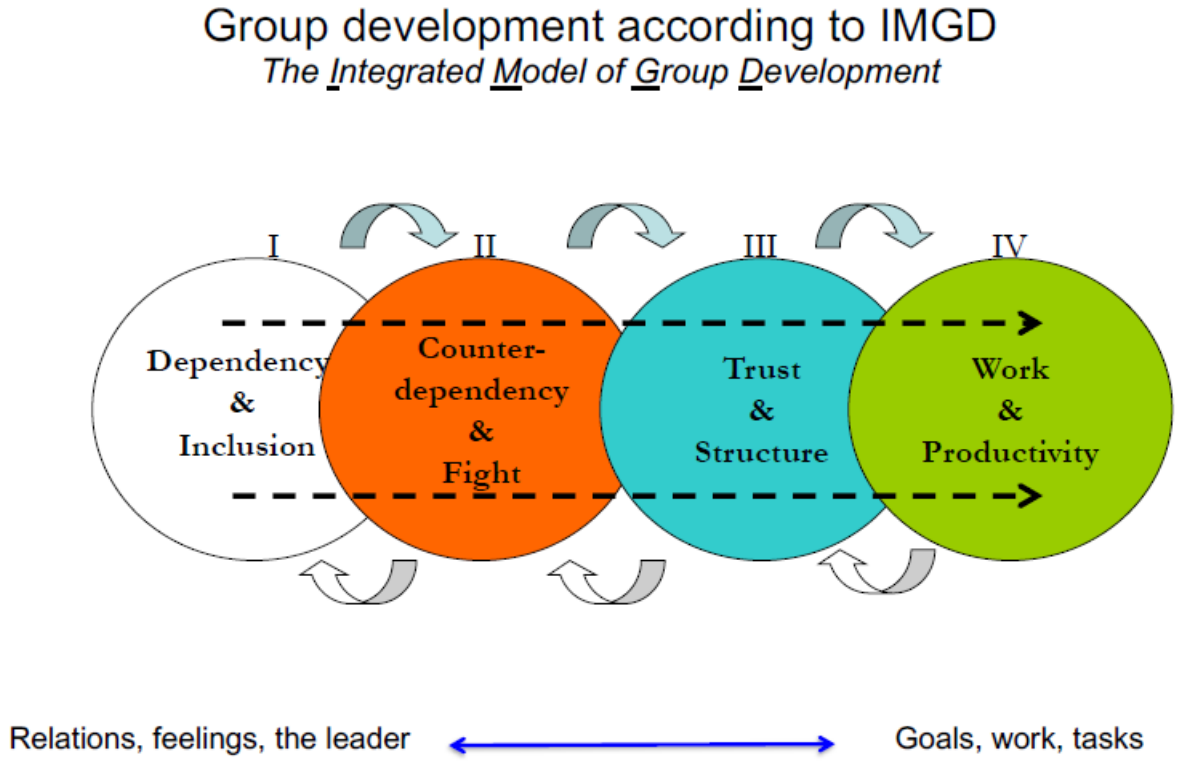
DD2480: Software Engineering Fundamentals

Lecture 3 Part 2: Introduction to Essence

Idea of deliberate practice



Team work


Group development according to IMGD
The Integrated Model of Group Development
I
II
III
IV
Dependency & Inclusion
Counter-dependency & Fight
Trust & Structure
Work & Productivity
Relations, feelings, the leader
Goals, work, tasks

How to give feedback

Constructive feedback…

1. do not intend evaluate the results already achieved, but rather pointing out the forward direction
2. aims to reinforce a behavior or to change a behavior
3. does not objectify the person by focusing on personal attributes like traits, character, attitudes, values or intentions
4. always focus on specific behaviors or actions and their consequences – the functional level
5. is facilitated by clear agreements on objectives, performance collaboration, norms and values
6. is based on benevolence, sincerity and a belief in change and development.

SEMAT

SEMAT: Software Engineering Method and Theory

SEMAT is the community working on Essence

www.semat.org

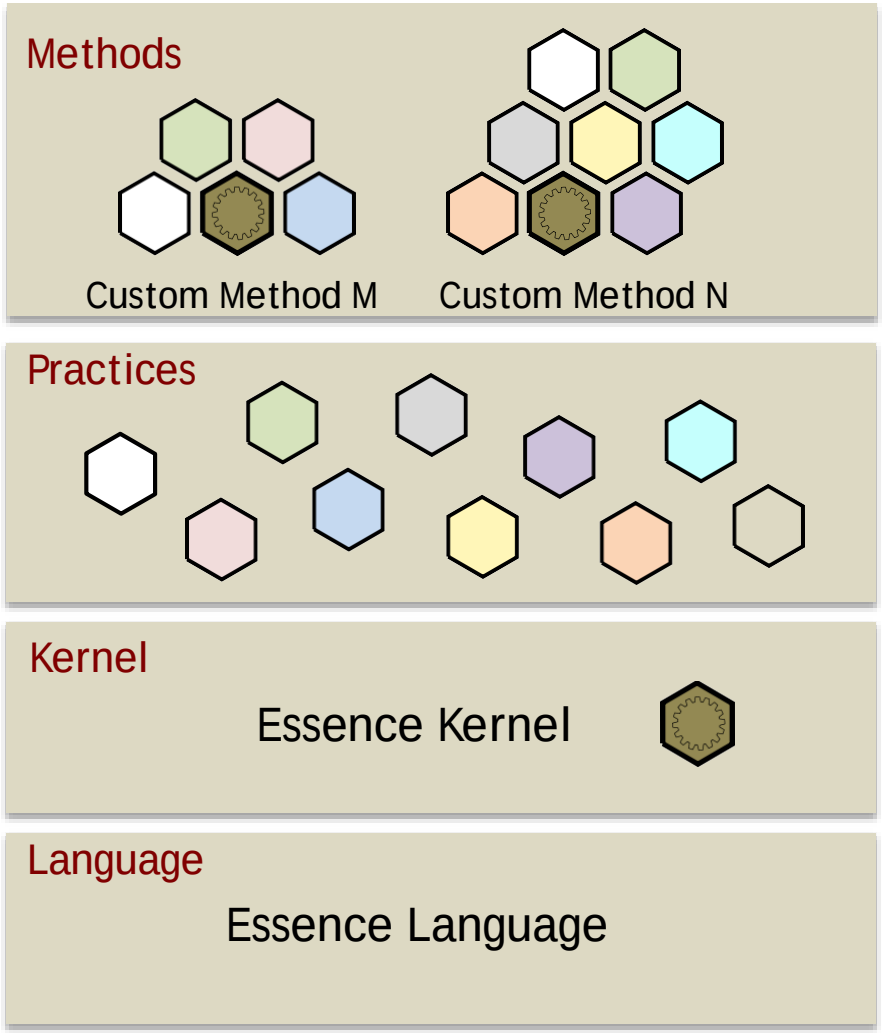
Created in : 2009

Founders: Ivar Jacobson, Bertrand Meyer, Richard Soley

Vision: Re-found software engineering as a rigorous discipline based on a general theory of software engineering and a unifying process framework

Essence

Essence is the work of SEMAT and includes a Language and Kernel

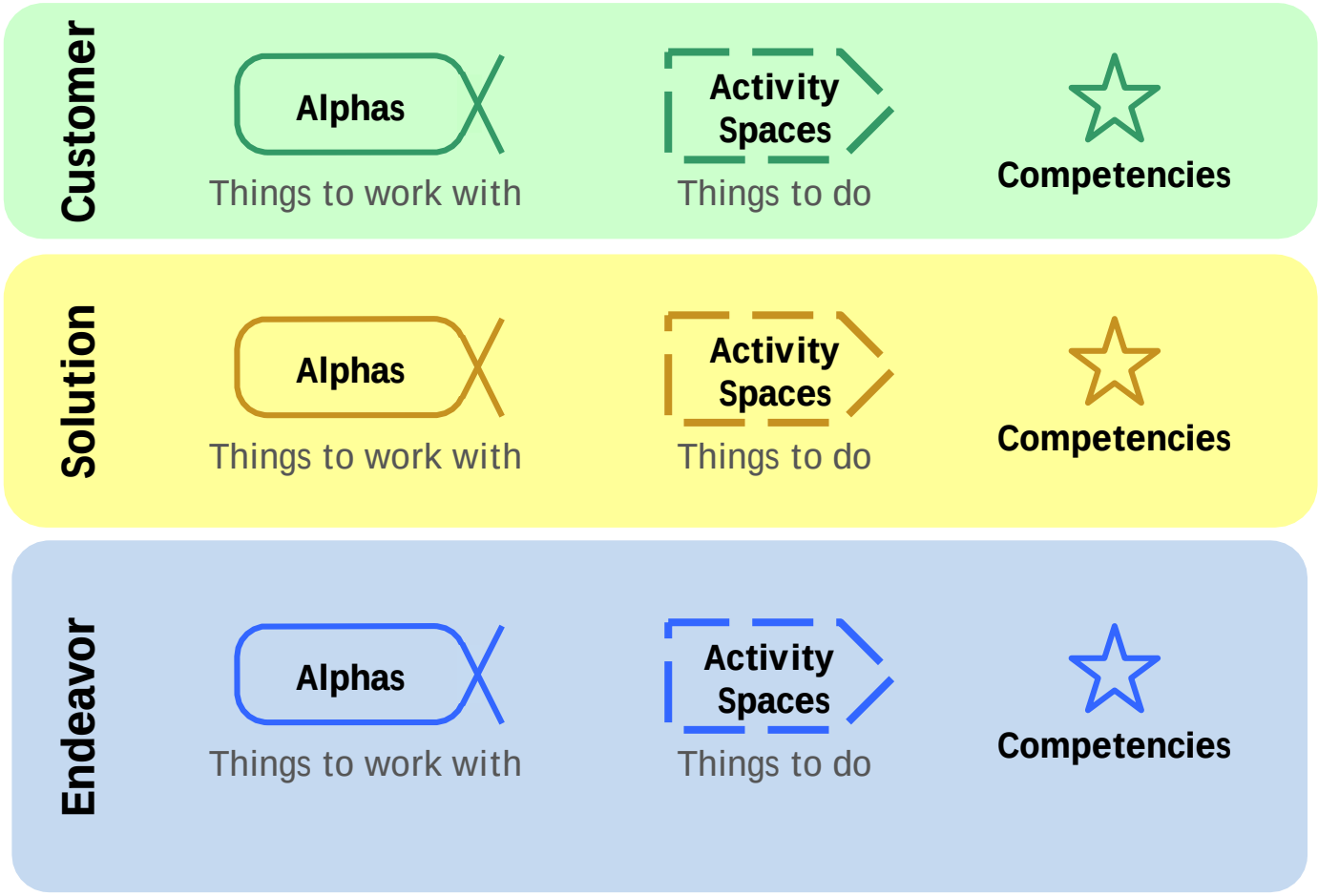


Structure of the Framework

Essence Language & Kernel became OMG beta standard in 2013

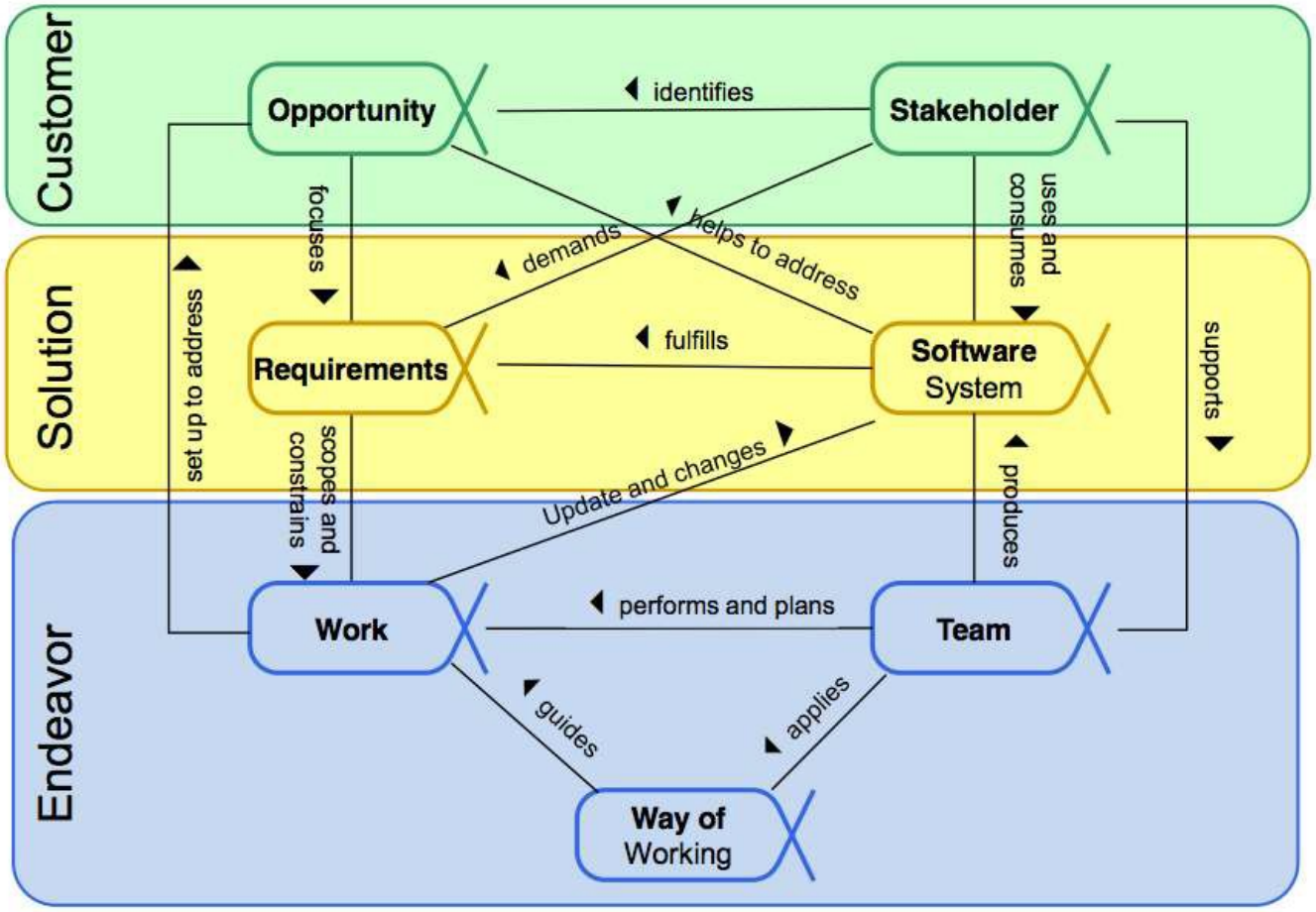
State-based Progress Monitoring & Goal-driven Project Steering

Essence Kernel



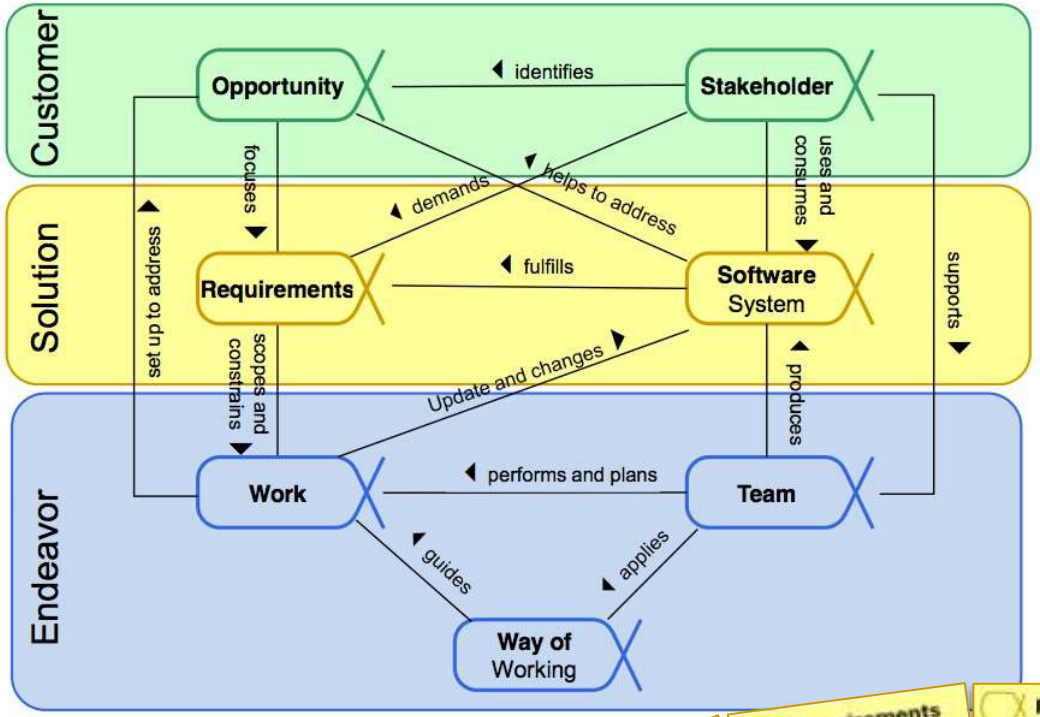
Essence Kernel Alphas

The Things to Work With

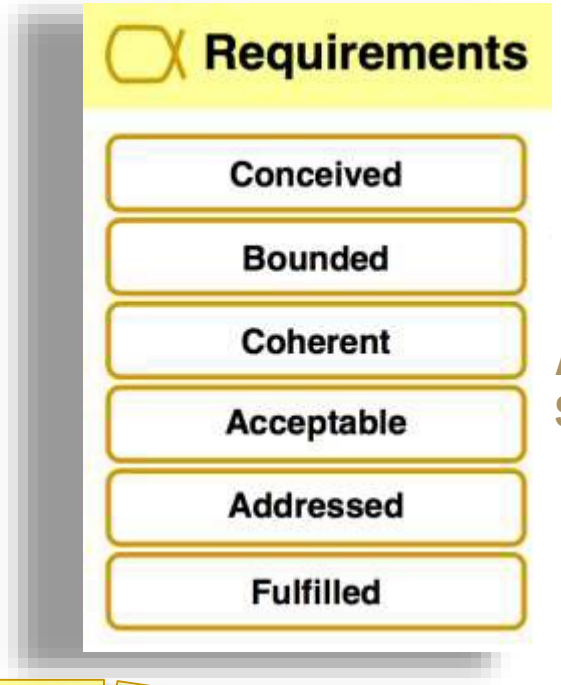


Alpha: **A**bstract-**L**evel **P**rogress **H**ealth **A**ttribute

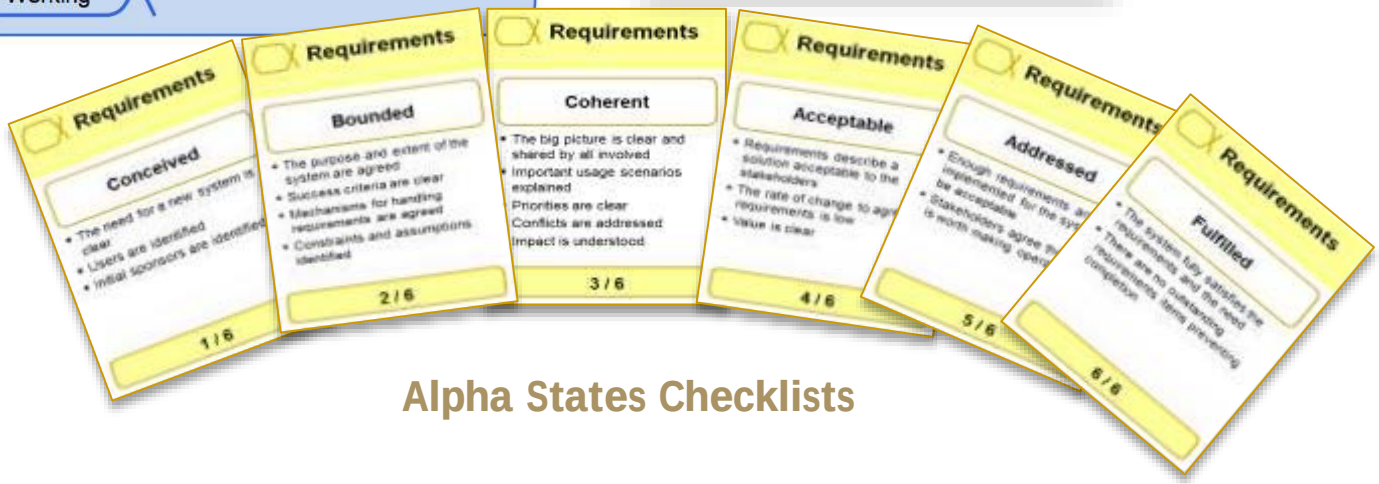
Essence Kernel Alphas



Kernel Alphas



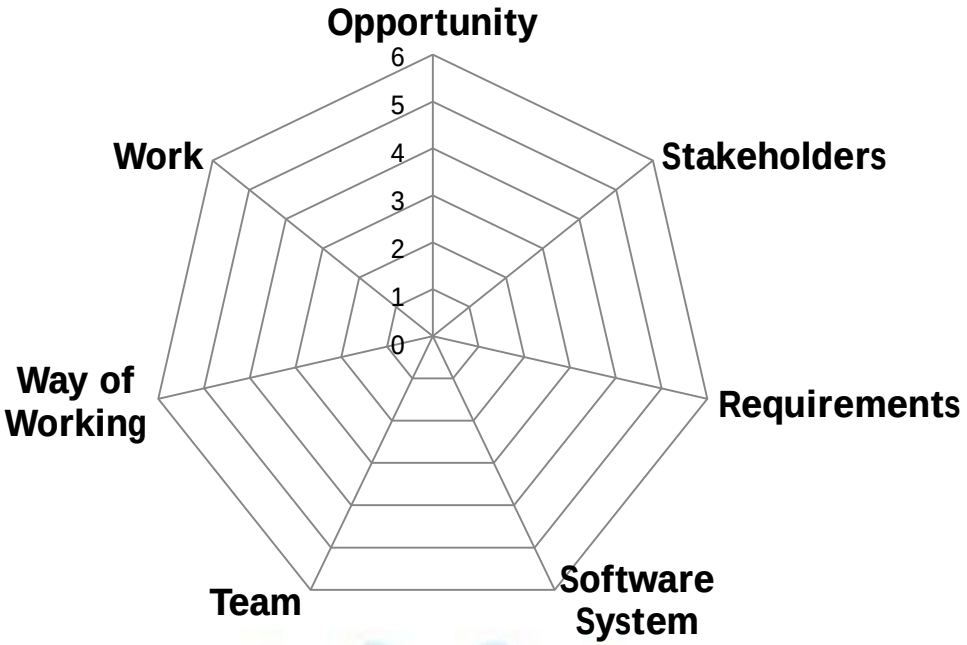
Alpha States



Alpha States Checklists

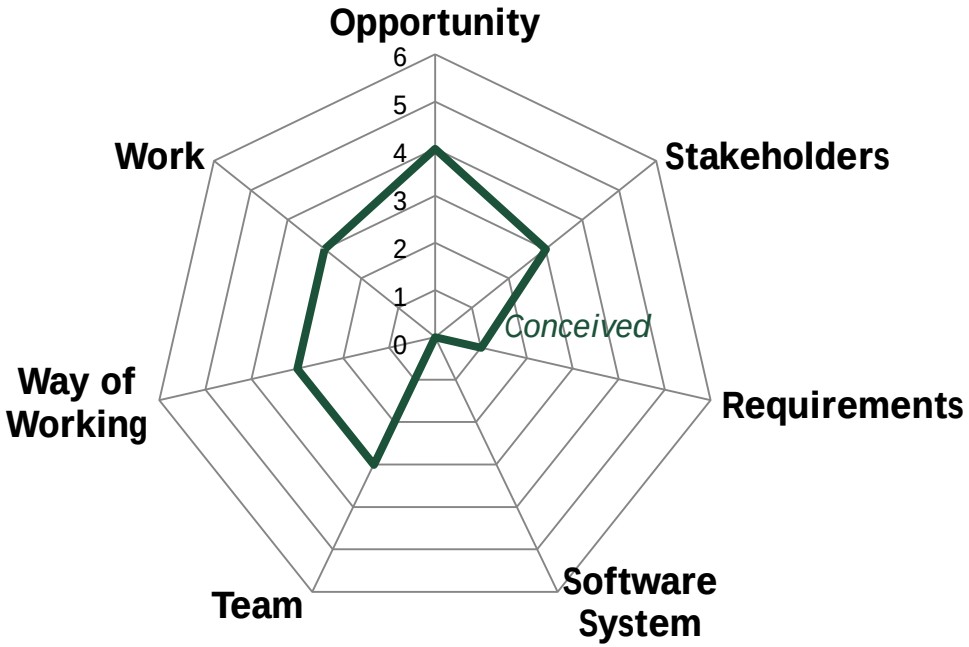
Essence Kernel in Practice

- Goals are set using Alphas
- Progress is monitored using Alphas States defined by the cards



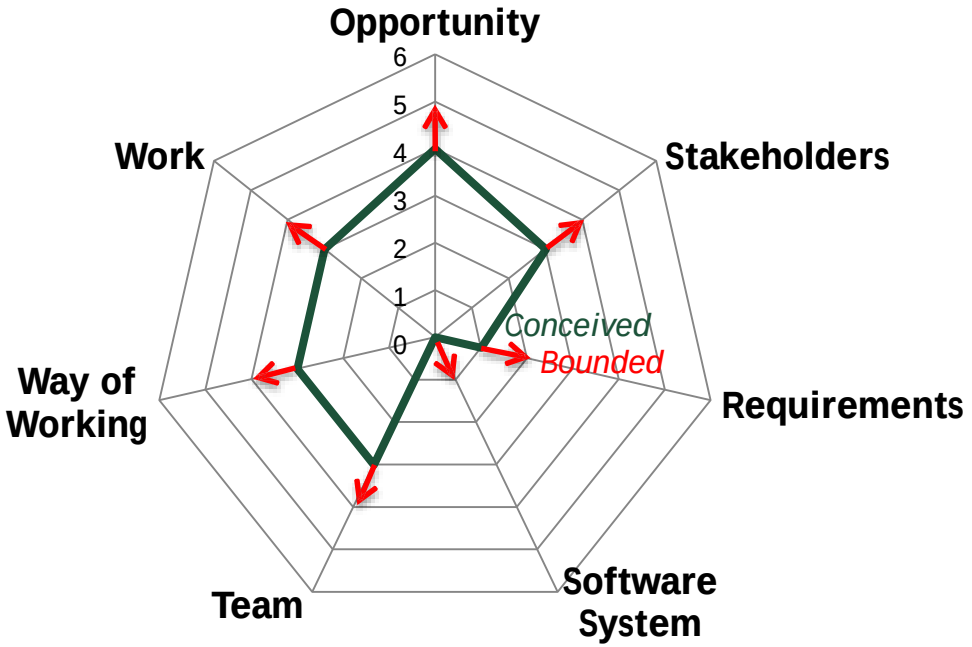
Essence Kernel in Practice

- Progress is quantified for each Alpha
- Easy to visualise and assess the current state
- Easy to identify problems



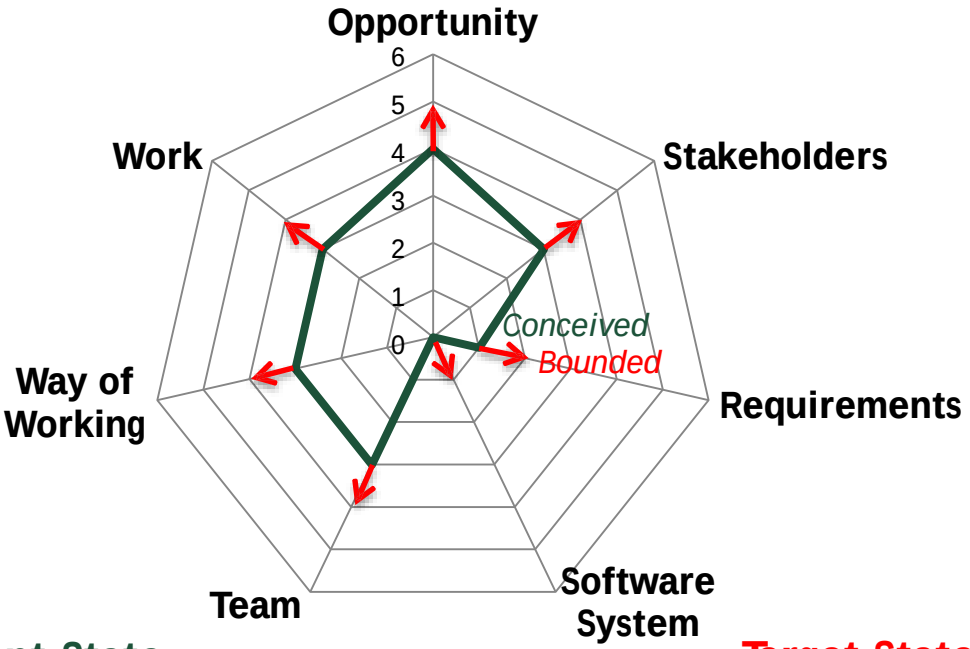
Essence Kernel in Practice

Set the goal: which Alpha to work on and which state to achieve



Essence Kernel in Practice

Identify actions required to achieve target state



Current State

Requirements

Conceived

- The need for a new system is clear
- Users are identified
- Initial sponsors are identified

1 / 6

Work Items:

- ❑ Define Project Scope
- ❑ Clarify Success Criteria

Target State

Requirements

Bounded

- The purpose and extent of the system are agreed
- Success criteria are clear

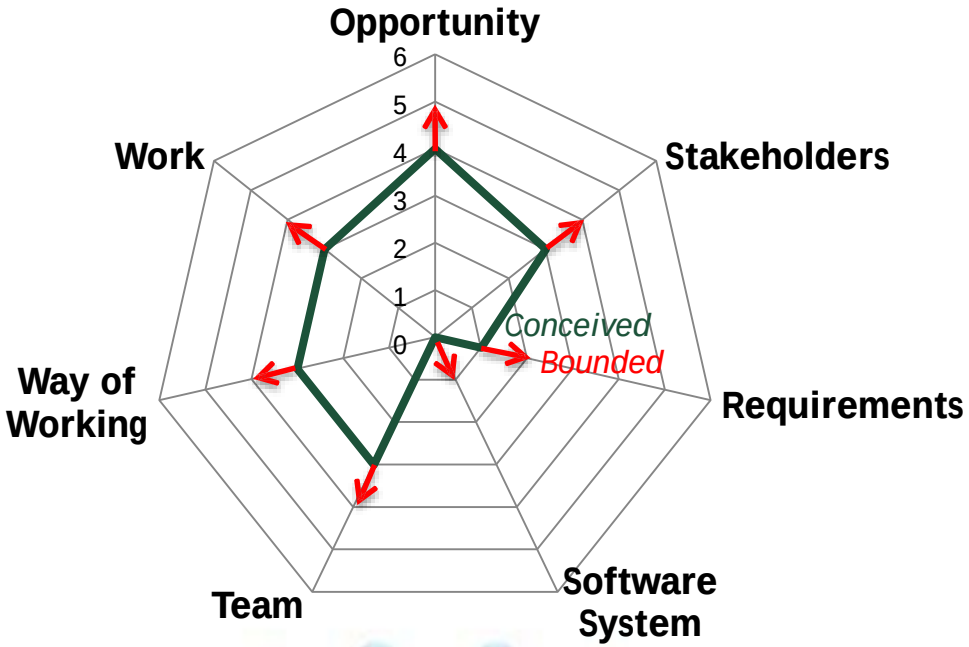
Goals

- Mechanisms for handling requirements are agreed
- Constraints and assumptions identified

2 / 6

Essence Kernel in Practice

As time progresses, the team achieves balanced progress in all Alphas



Essence Kernel Benefits

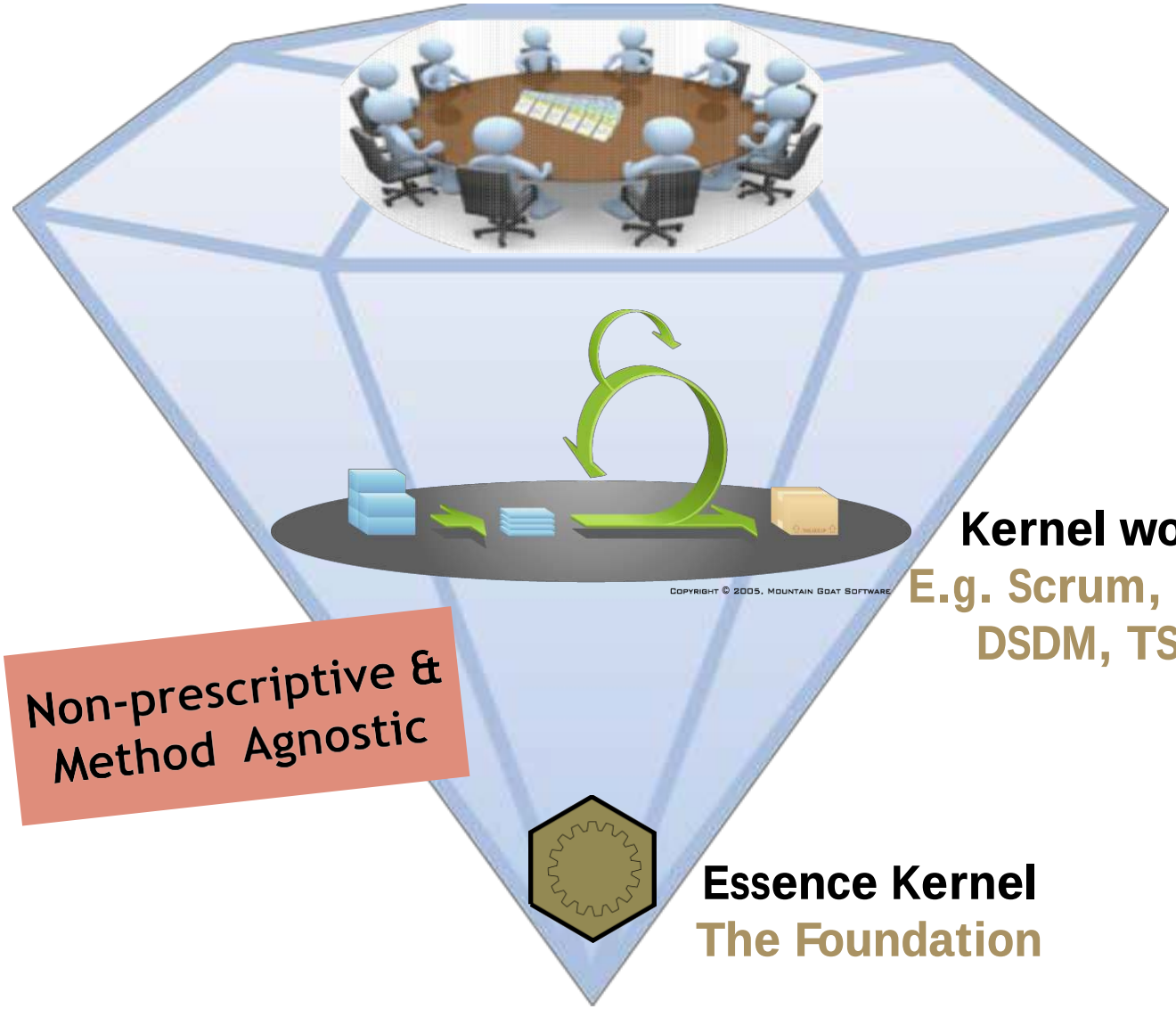
The Essence kernel provides a structure and mechanism for:

- **Progress monitoring**
- **Team reflection**
- **Risk management**
- **Project steering**

In a holistic, simple, lightweight, non-prescriptive and method-agnostic fashion

How does the Essence kernel complement other ways of working?

Essence Kernel and Development Methods



Non-prescriptive & Method Agnostic

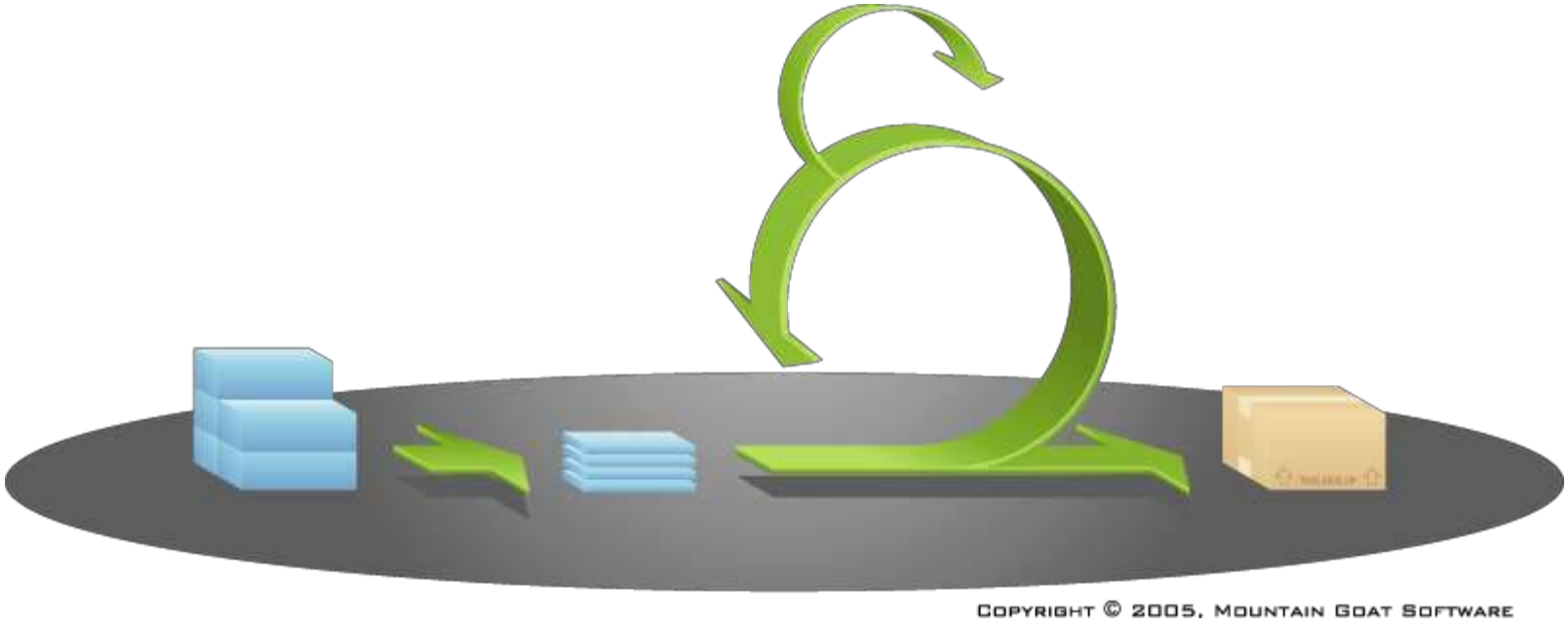
Kernel works with Any Method

E.g. Scrum, XP, Kanban, DAD, Safe, DSDM, TSP, RUP, Crystal, etc.

Essence Kernel

The Foundation

Defining Scrum Practice



About Scrum

- Scrum consists of Scrum Teams and their associated roles, events, artifacts, and rules.
- Scrum's roles, artifacts, events, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum.

- Source
  - K. Schwaber and J. Sutherland, "The Scrum Guide", Scrum.org, October 2011.
  - http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf

Scrum Concepts

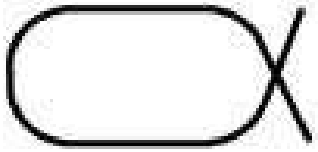
- Scrum team (roles)
  - Product Owner
  - Development Team (of developers)
  - Scrum Master
- Scrum artifacts
  - Product Backlog
  - Sprint Backlog
  - Increment
- Scrum events
  - The Sprint
  - Sprint Planning Meeting
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

Step 0: SEMAT Kernel & Essence Language Concepts

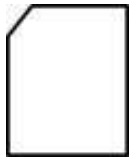
- A standard **Kernel** provides a baseline starting point – a "map" of the software development endeavour.
- **Practices** add details and provide specific guidance on particular aspects of the software development
- Key language concepts: **Alpha**, **Activity Space**, **Work Product** and **Activity**

Kernel

Practice



Alpha

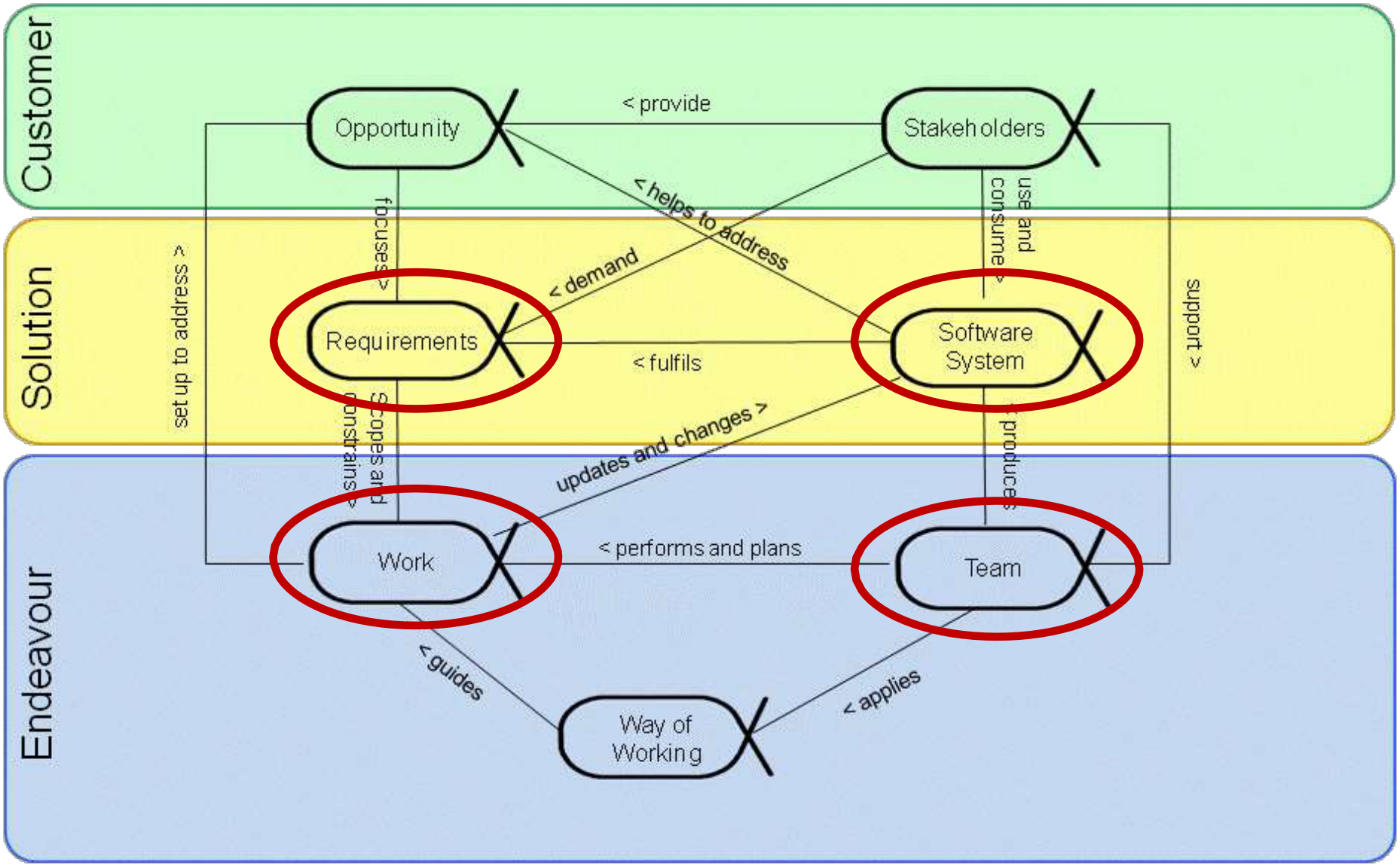


Work Product

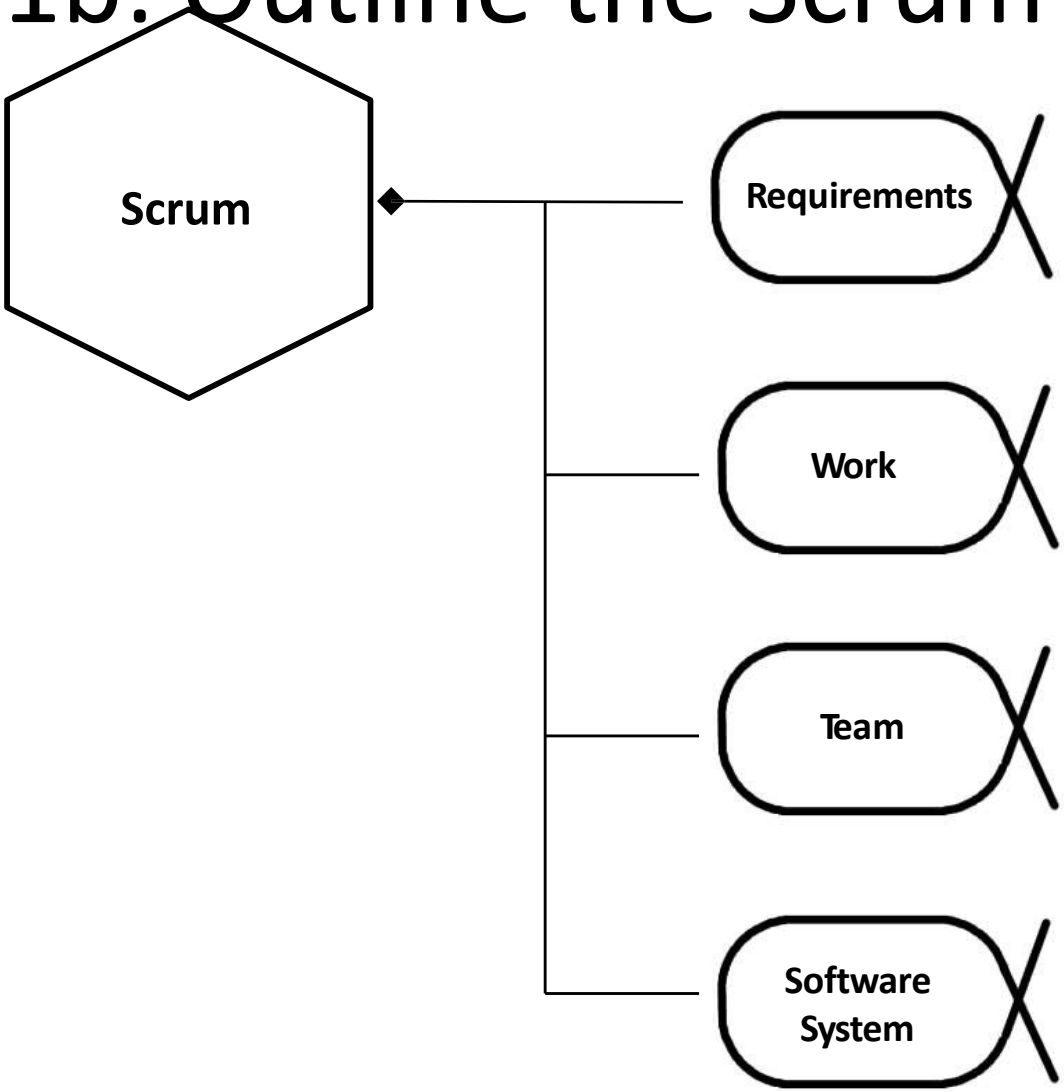
Activity Space

Activity

Step 1a: Identify relevant Kernel Alphas

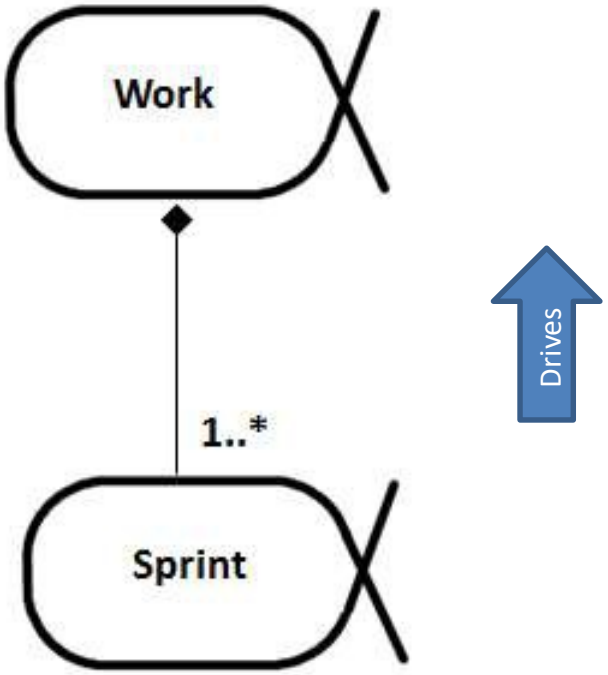


Step 1b: Outline the Scrum Practice



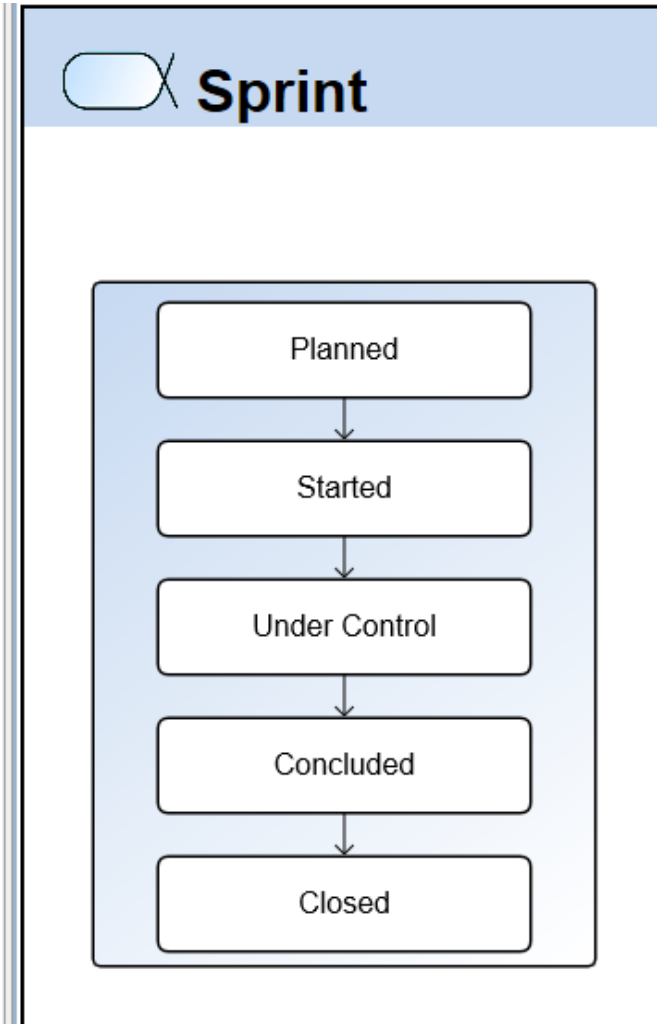
Step 2a: Add sub-alphas

- Extending the Work Alpha

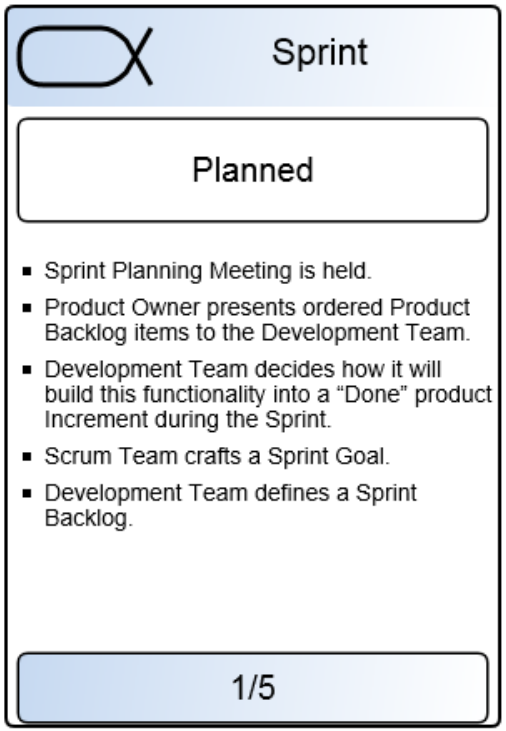


- The Work alpha is typically used for the duration of a development project that may cover a number of sprints.
- Thus we define a new sub-alpha called **Sprint**.
- Sub-alphas drive their parent alphas

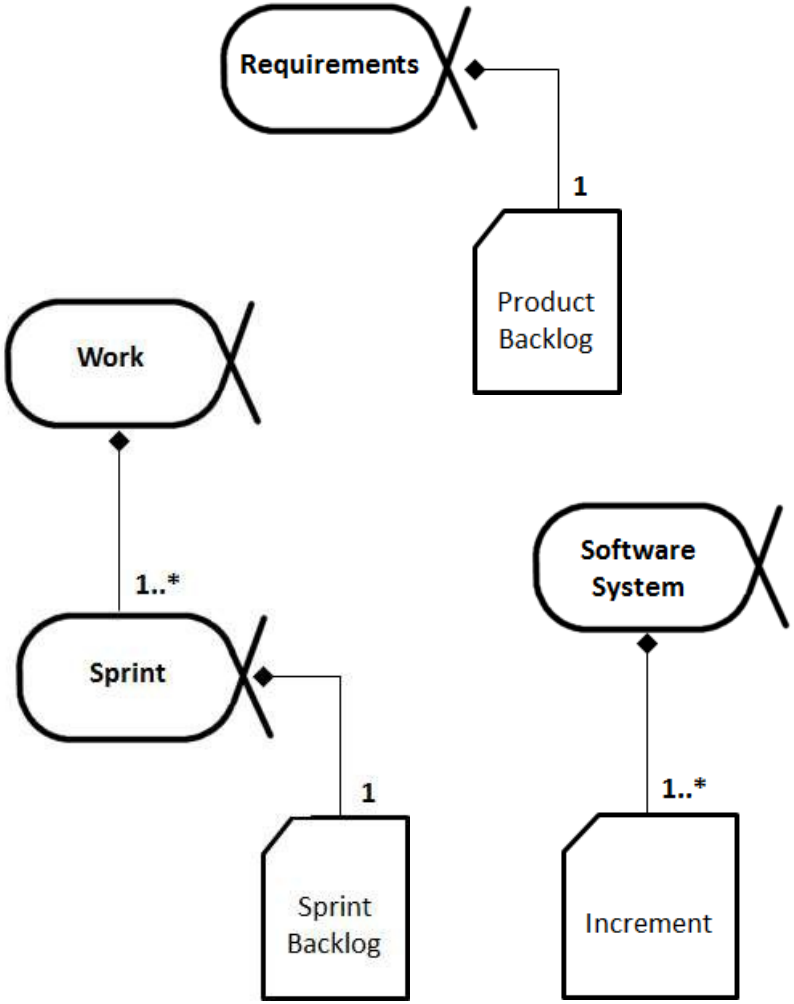
Step 2b: Define alpha states and checkpoints



- Specific Scrum rules are defined as part of the alpha state checkpoints.

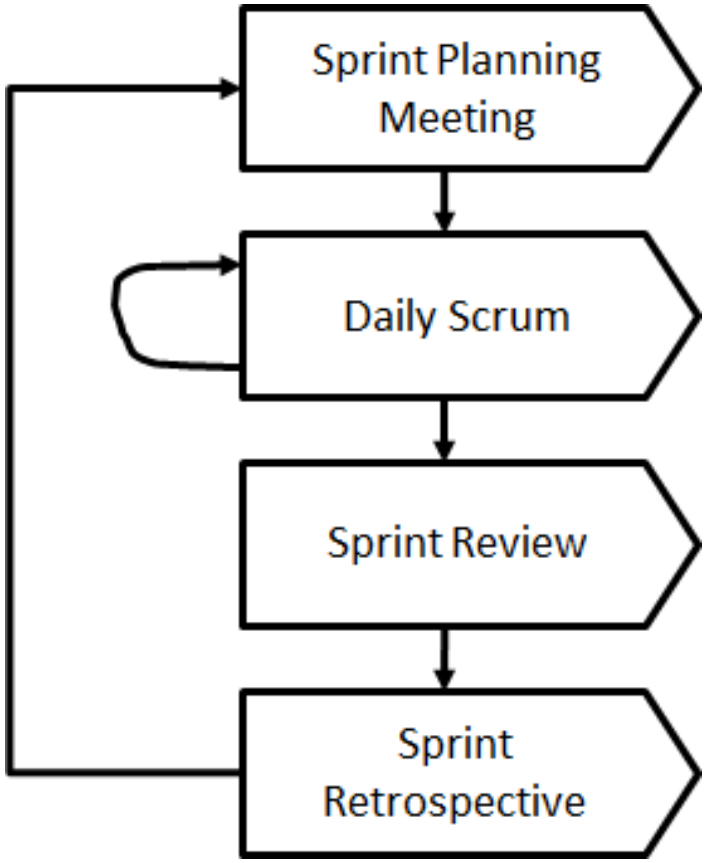


Step 3: Add Work Products



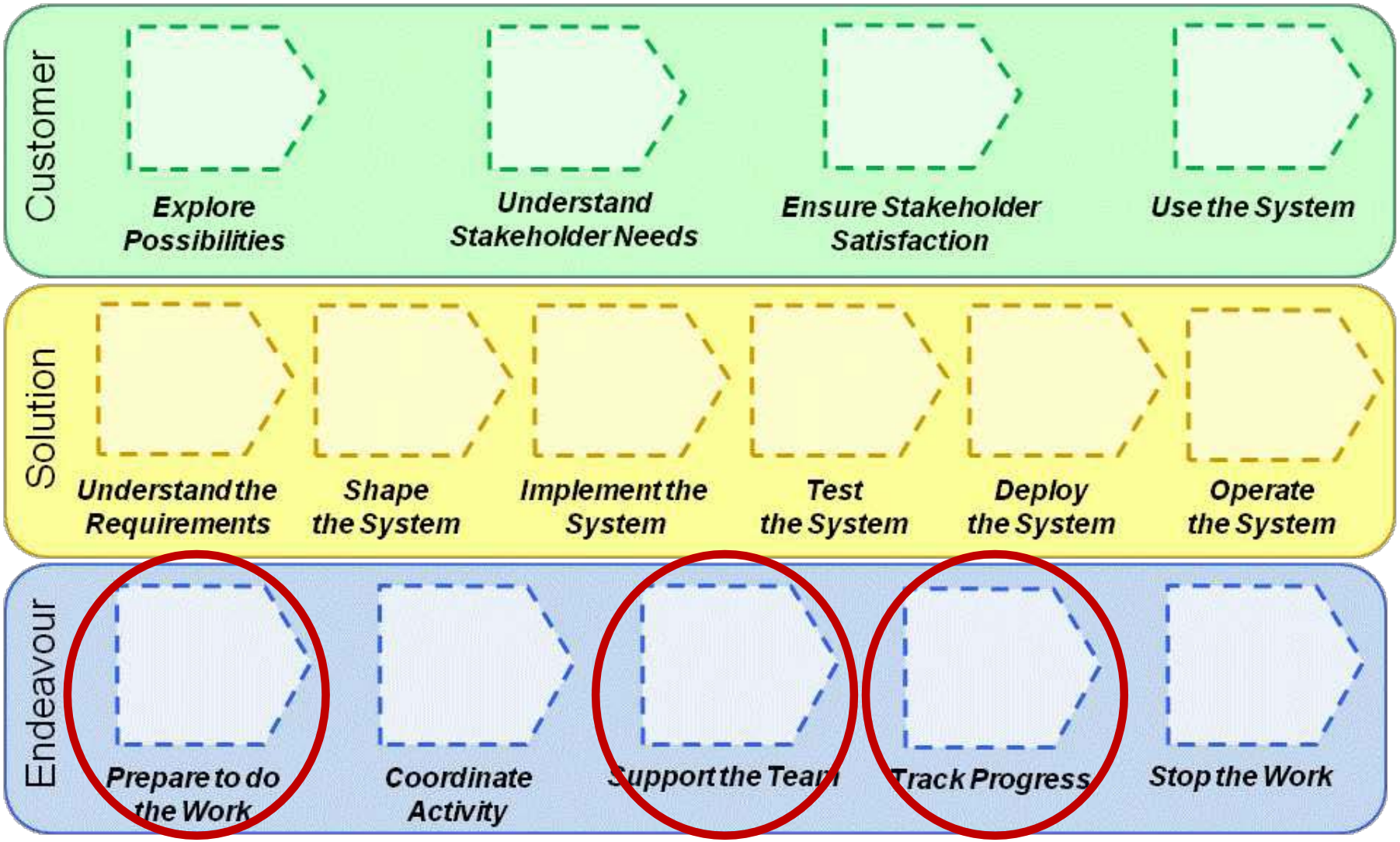
- "The **Product Backlog** is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product."
- "The **Sprint Backlog** is the set of Product Backlog items selected for the Sprint plus a plan for delivering the product Increment and realizing the Sprint Goal."
- "The **Increment** is the sum of all the Product Backlog items completed during a Sprint and all previous Sprints."

Step 4a: Define Activities

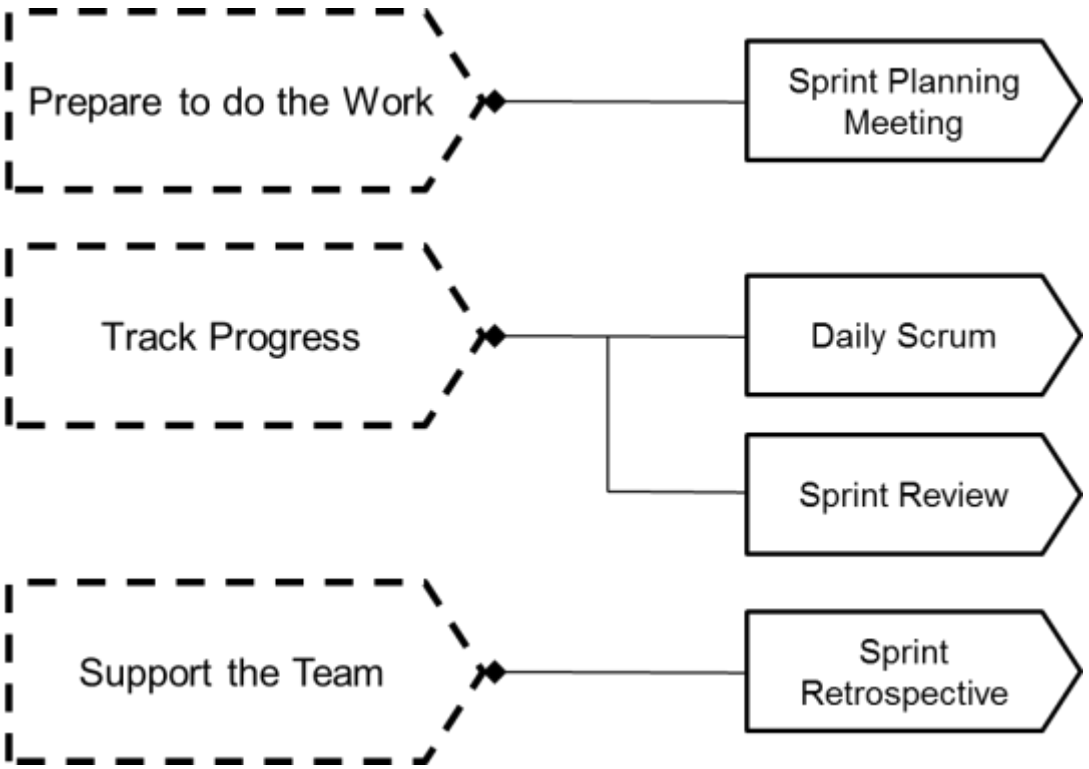


- "The work to be performed in the Sprint is planned at the Sprint Planning Meeting."
- "The Daily Scrum is a 15-minute time-boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours."
- "A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed."
- "The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning Meeting."

Step 4b: Identify relevant Kernel Activity Spaces



Step 4c: Relate activities to Kernel Activity Spaces



Brief note on Assignment #1. "DECIDE"

- Goal: write a function called DECIDE()
  - A part of a hypothetical anti-ballistic missile system
- Generates Boolean signal to launch or not an interceptor based on input radar tracking info
- Info is available at the instant the function is called
- 15 Launch Interceptor Conditions (LIC)
- Function determines which LICs are true for the input (up to 100 of planar data points representing radar echoes)
- Decision to launch made only if all relevant combinations of LICs met

Main concepts: LIC

- 15 LICs are defined in the requirements document.
- Each LIC represents a certain characteristic of radar tracking data

Examples:

LIC 0:

There exists at least one set of two *consecutive* data points that are a distance greater than the length, **LENGTH1**, apart where (0 ≤ **LENGTH1**)

LIC 1:

There exists at least one set of three *consecutive* data points that cannot all be contained within or on a circle of radius **RADIUS1**, where (0 ≤ **RADIUS1**)

…LIC 14 …

- Check the glossary of the terms

Main concepts: data points

- NUMPOINTS – number of planar data points
- POINTS – array containing the coordinates (X,Y) of data points
- Dynamic input parameters (available when DECIDE() is activated)
- LICs are evaluated over the data points (radar data)

Main concepts: CMV

- The *Conditions Met Vector* (CMV)
- Intermediate result
- Set according to the results of LIC calculations
- The global array element CMV[i] should be set to true if and only if the i_{th} LIC is met.

Main concepts: LCM

- Logical Connector Matrix (LCM) defines which individual LICs must be considered jointly (static input parameter)
- 15x15 symmetric matrix with elements ANDD, ORR, NOTUSED

LIC	0	1	2	3	4	...	14
0	ANDD	ANDD	ORR	ANDD	NOTUSED	...	NOTUSED
1	ANDD	ANDD	ORR	ORR	NOTUSED	...	NOTUSED
2	ORR	ORR	ANDD	ANDD	NOTUSED	...	NOTUSED
3	ANDD	ORR	ANDD	ANDD	NOTUSED	...	NOTUSED
4	NOTUSED	NOTUSED	NOTUSED	NOTUSED	NOTUSED	...	NOTUSED
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
14	NOTUSED	NOTUSED	NOTUSED	NOTUSED	NOTUSED	...	NOTUSED

Main concepts: FUV

- The *Final Unlocking Vector* (FUV)
- Intermediate result
- Generated from the *Preliminary Unlocking Matrix*.
- PUV indicates whether the corresponding LIC should be considered as a factor in signaling interceptor launch.
- FUV[i] should be set to true if PUV[i] (ith column of PUM) is false (indicating that the associated LIC should not hold back launch)
- or if all elements in PUM row i are true.

Main concepts: LAUNCH

- **LAUNCH** Final launch / no launch decision encoded as ”YES”, ”NO” on the standard output.
- The final result
  - is based on the FUV.
- “YES” if all elements in the FUV are true,

i.e., if and only if FUV[i] is true for all i, $0 \leq i \leq 14$.

Working on the assignment

- Goal is to focus on software engineering part of the assignment: implement the DECIDE program according to the modern development techniques.
- Grading focuses the process rather than on the program itself
- You must use a development platform (Github, Bitbucket or KTH Github).
- Only the content published on the development platform is used for grading.
  - 1) the code 2) the issues 3) the pull requests 4) the continuous integration data (eg Travis) if available.

Final notes

- LICs are good units to split the work
- Ensure even distribution of work (simple and complex LICs)
- Organise your groups as soon as possible

- Grading criteria for the assignment in Canvas

- Observe the code of conduct

- Good luck!