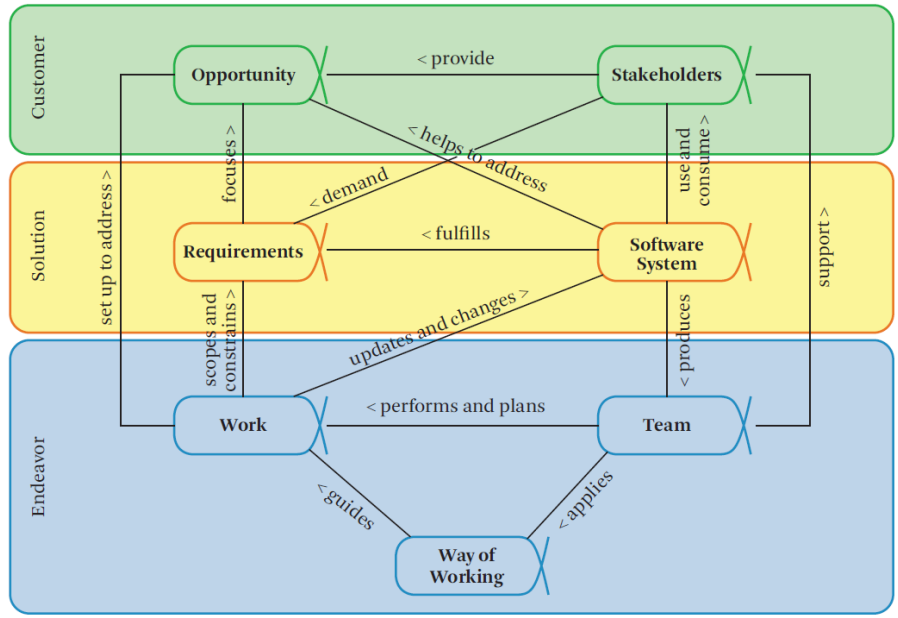
Software Engineering – Lecture 13
The Essence of Software Engineering

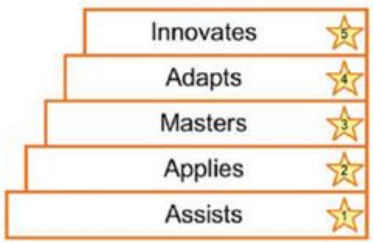
Adapted after
I. Jacobson, H. Lawson, Pan-Wei Ng, P.E. McMahon, M. Goedicke
The Essentials of Modern Software Engineering, 2019, ACM Books

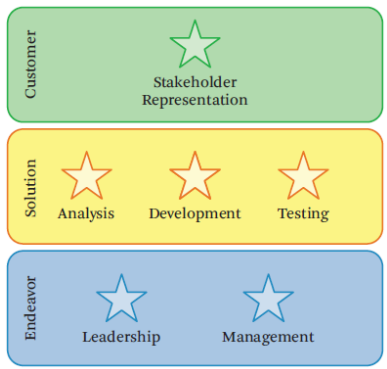
Rezumat

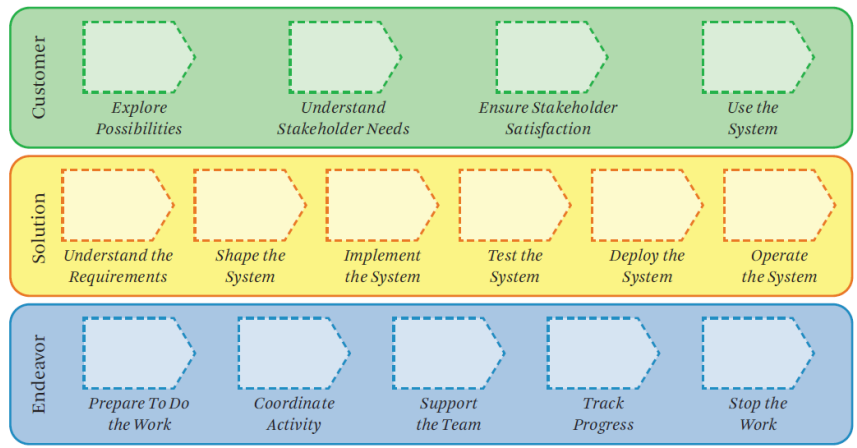
Essence : limbaj + nucleu

Alpha		Element esențial al efortului de dezvoltare relevant pentru *evaluarea progresului și sănătății* efortului.
Work Product		Lucru tangibil (artefact) realizat de practicieni pe parcursul activităților de inginerie software.
Activity		O activitate realizată de practicieni.
Competency		Abilitate, capabilitate, aptitudine, cunoștință, sau calificare necesară unui anume mod de lucru.
Activity Space		Container pentru ceea ce trebuie făcut în efortul de dezvoltare. Poate conține zero sau mai multe activități.
Pattern		O soluție la o problemă tipică, printr-o organizare particulară a unor elemente de limbaj.









Subiecte tratate

Essence games

Using Essence kernel

Using practices

Essence cards

Essence card - tangible thing that contains a concise description of the most important information about an Essence element.

- act as reminders to practitioners
- additional details are available in complementary guidelines.

Motivation

Team performance depends on effective communication, common understanding, trust $\Rightarrow$ collaboration.

Cards utility

Used to play ***collaborative games*** as facilitating tools in a variety of settings and purposes (ex. obtain a consensus about the work)

- used to introduce the kernel and practice elements
- to understand endeavor purpose, benefits and problems
- resolve conflicts in limited time

Essence games

Serious games (beyond entertainment)

Simulate lifelike events aiming to achieve specific goals

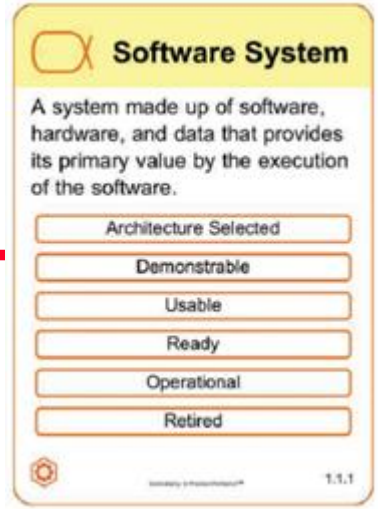
- solve a particular real-world problem
- learn something new.
- develop skills (basic mental abilities such as perception, attention, and decision making).

Essence games *:*

- Cooperative, consensus-based (not competitive)
- Highly reusable aids when carrying out multiple practices.
- Players express their thoughts clearly, listen to one another, share information and resources, learn from one another, identify solutions, negotiate, and make common decisions.
- Stimulate a team to :
  - discuss the issues related to the health and progress of their own endeavors.
  - look ahead at states and checklists not yet achieved clarifying what is important to do next.

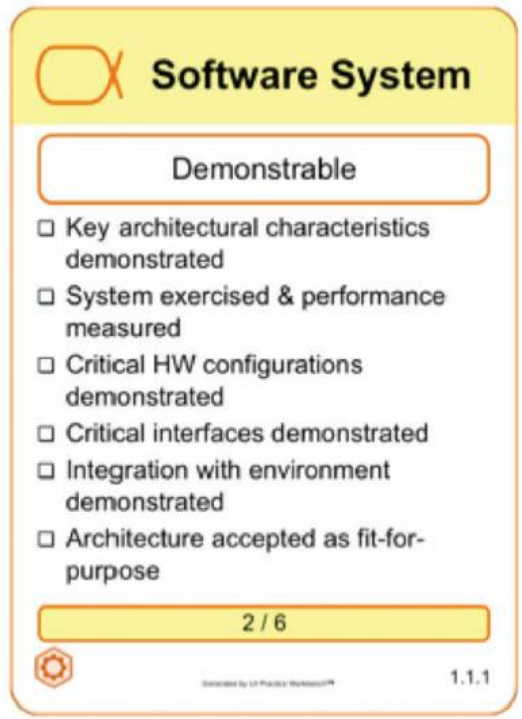
Essence games

Progress Pocker



Track progress based on the state transitions of alphas.

Items in a checklist provide a hint of what needs to be done.



Problem :

Are subject to interpretation by the team members, with different opinions on the their meaning.

⇓

need for agreement

Solution : Progress Poker

- facilitate discussion
- achieve understanding about of the current state of a particular alpha.

Fig. 8.1 Software System: Demonstrable alpha state card

Essence games
Progress Poker

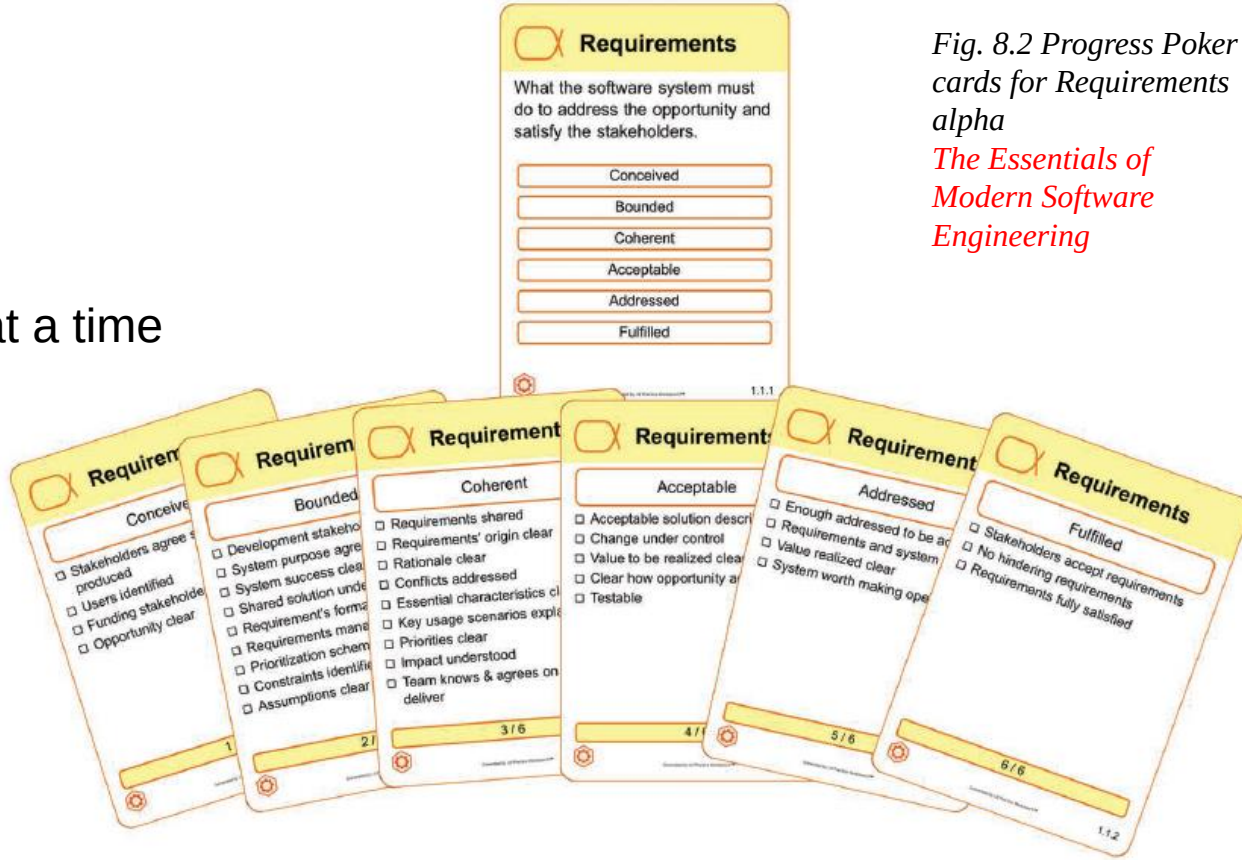


Fig. 8.2 Progress Poker cards for Requirements alpha
The Essentials of Modern Software Engineering

- Played with one alpha at a time
- Are used :
  - alpha overview card
  - alpha state cards
- Optimal team dimension

3-9 players

Rules :

- alpha card on the table
- each player places face down the card with his opinion
- compare the results
- discuss different choices (explain and motivate, starting with extreme ones)
- new round

The game ends when a *consensus* has been reached on the *current state* that has been achieved for a particular alpha.

Fig. 8.4 Initial position for Chasing the state game
The Essentials of Modern Software Engineering

Essence games
Chasing the state

- Played with all the alphas
- Are used :
  - alpha overview card
  - alpha state cards
- Optimal team dimension : 3-9 players

Essence game
Chasing the state

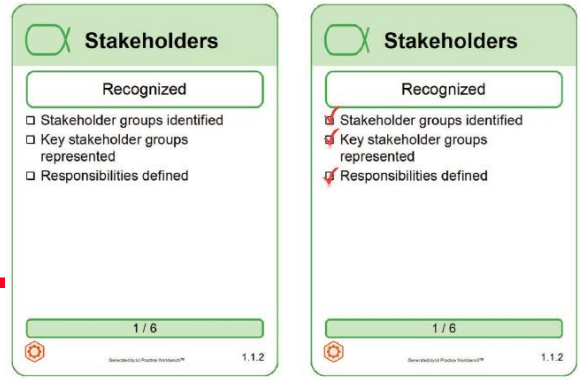


Fig. 8.5 Stakeholders alpha before and after discussion
The Essentials of Modern Software Engineering

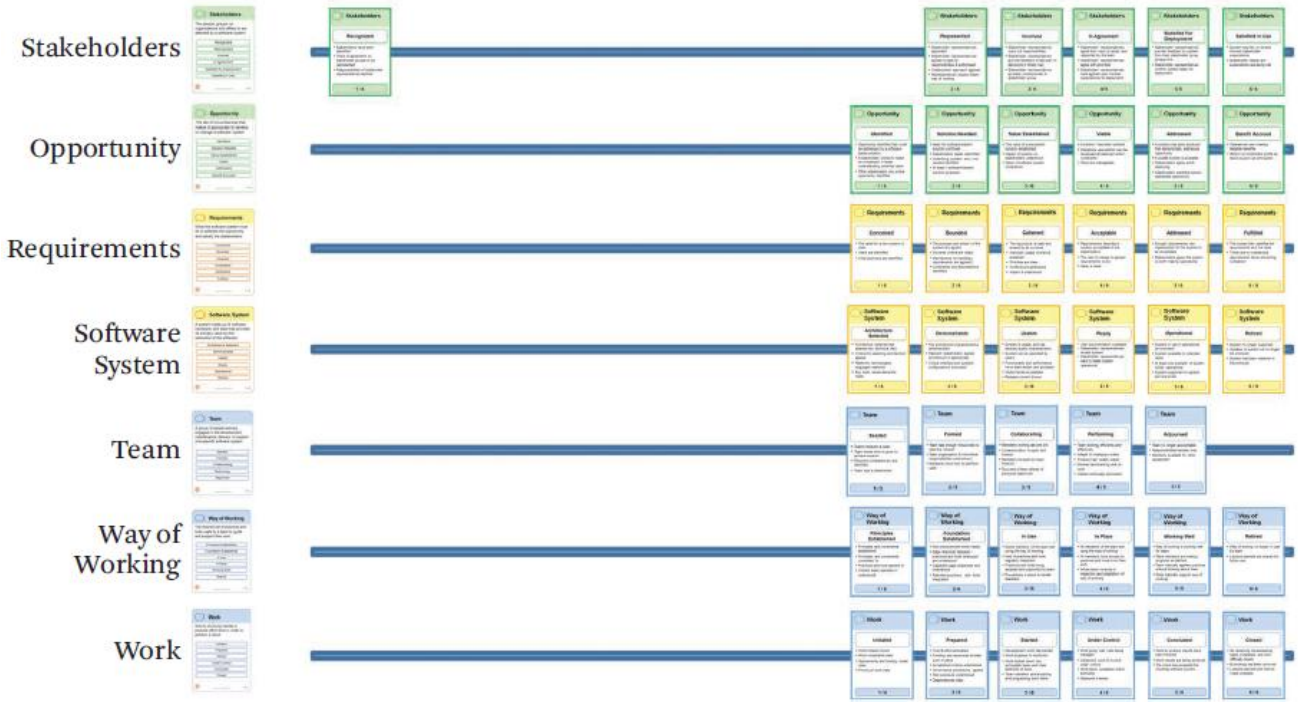


Fig. 8.6 Stakeholders alpha has reached first state
The Essentials of Modern Software Engineering

Rules:

For each *alpha*

- establish the state
- if consensus is not easily obtained then play Progress Poker

Essence games

Chasing the state result

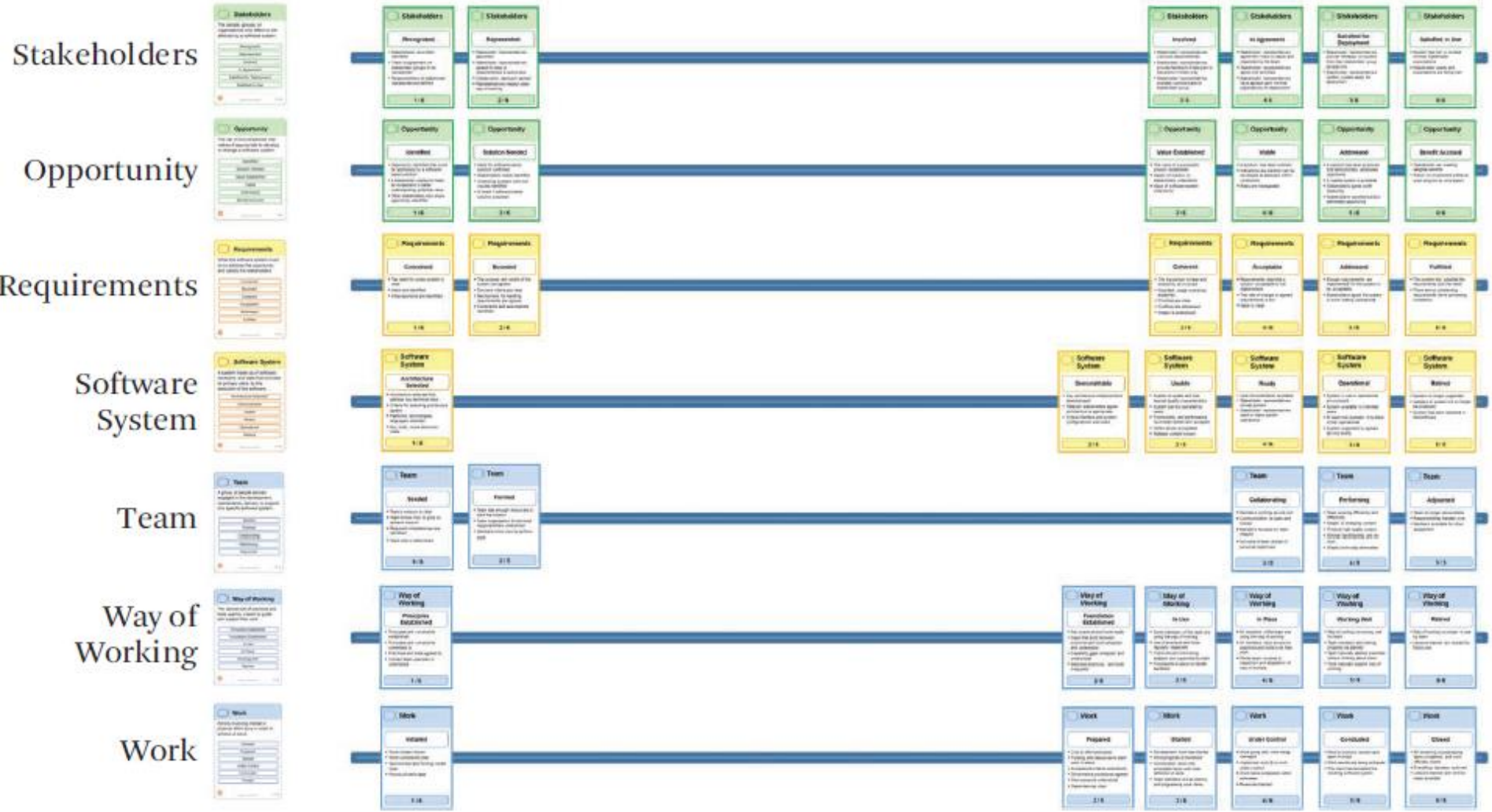


Fig. 8.8 The current states for all alphas have been identified

The Essentials of Modern Software Engineering

Essence games

Objective Go

Aim - agree upon the next steps

Played after the Chasing the State game

The objective of the next stage, which can be moving some or all *alphas* to a next state, is established.

Alphas usually progress in waves, depending on each other progress.

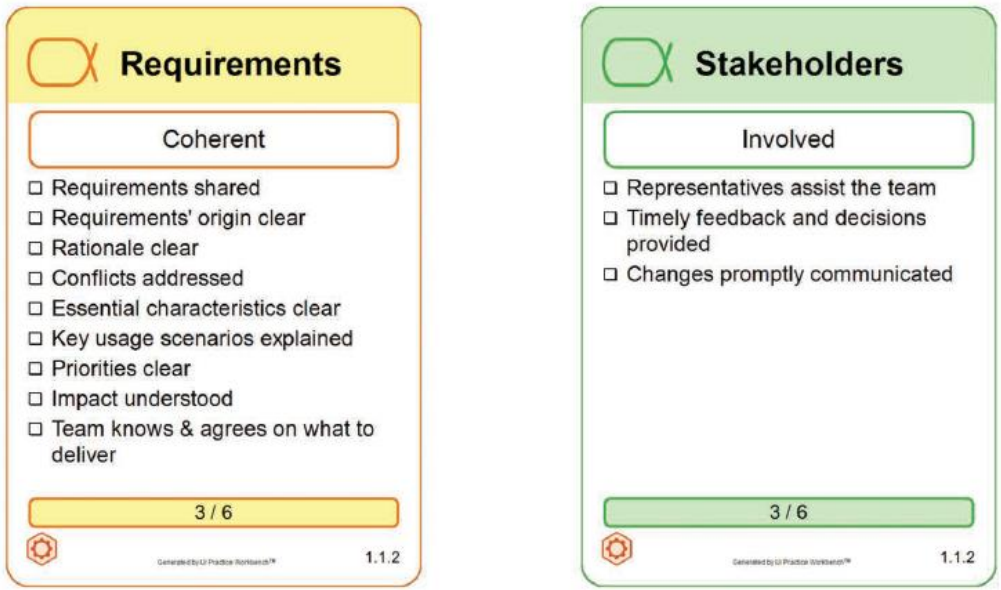


Fig. 8.9 Requirements and Stakeholders Alpha Wave
The Essentials of Modern Software Engineering

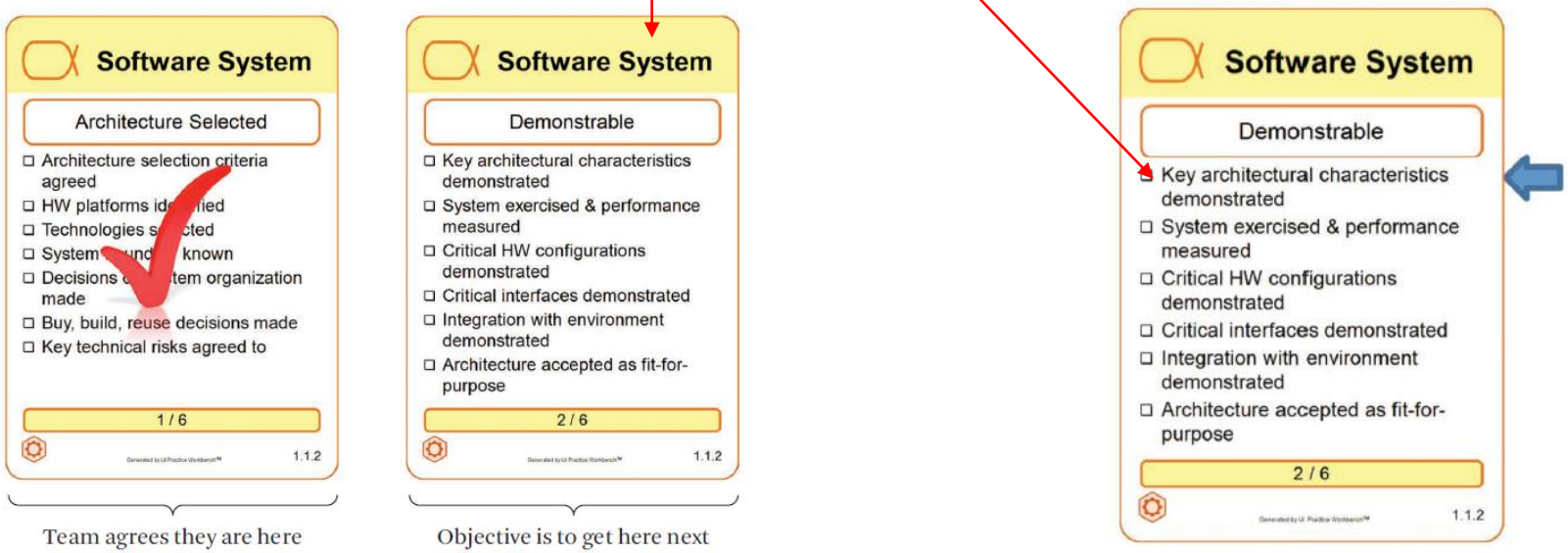
Essence games
Objective Go example

For each alpha discuss

- next state that should be achieved
- which checklist items for that state are not yet achieved
- the tasks needed to be done

Example :

Objective : Software System: Demonstrable;



Figures 8.10 and 8.11
The Essentials of Modern Software Engineering

Essence games

Objective Go example

Objective：

- Stakeholders: Involved,
- Software System: Demonstrable;
- Way of Working: Foundation Established;
- Work: Prepared.

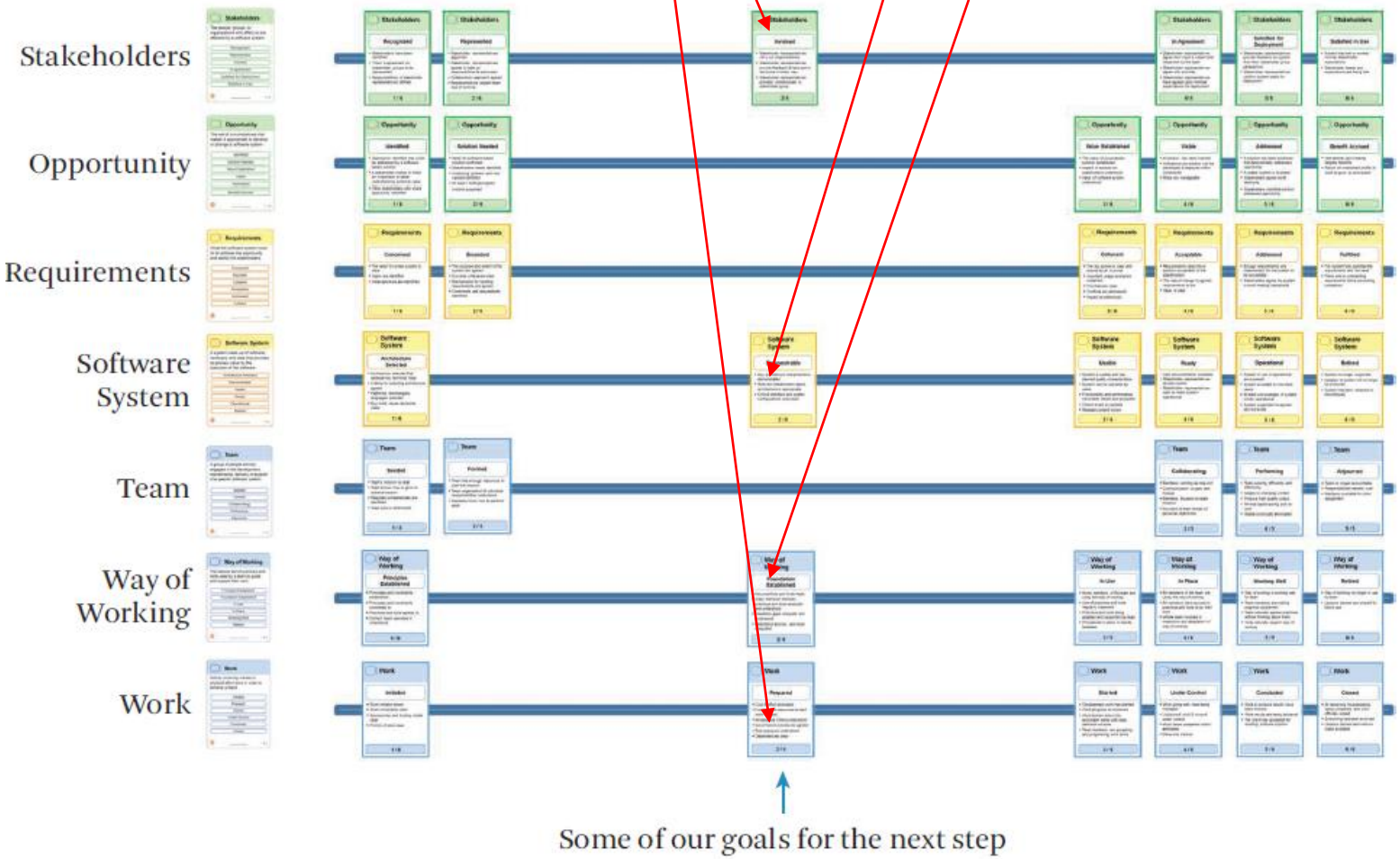


Fig. 8.12 The next step is represented by cards in the middle of the table

The Essentials of Modern Software Engineering

Essence games
Checkpoint construction

Checkpoint = set of criteria to be achieved at a specific point in time in a development endeavor; key point in the lifecycle of a software endeavor where an important decision must be made.

The set of criteria is defined using *alpha states.*

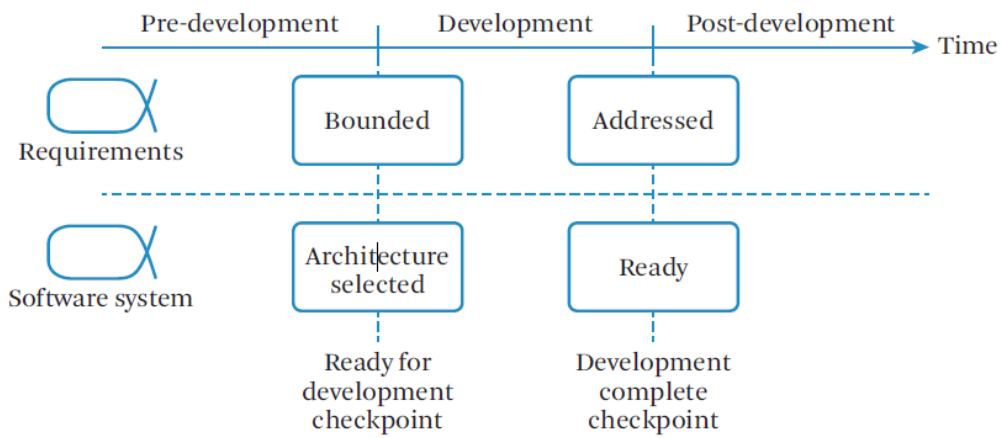
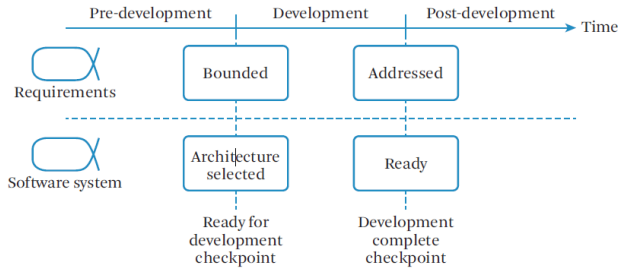


Fig. 8.9 Requirements and Stakeholders Alpha Wave
The Essentials of Modern Software Engineering

Essence games
Checkpoint construction



Checkpoint construction is used to synchronize teams working in parallel.

⇓

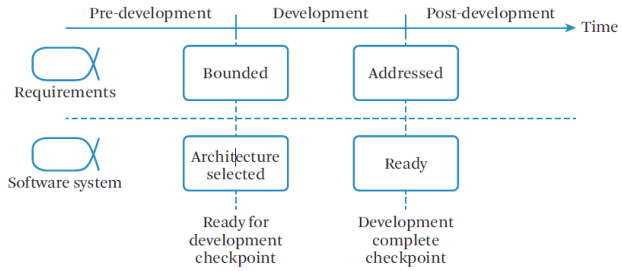
Must be pecified by the stakeholders of the whole endeavor and not by every team participating in the endeavor.

⇓

The game is played by the *stakeholder team.*

Stakeholder team = a few key stakeholder members that can represent the views of the stakeholders.

Essence games
Checkpoint construction



Checkpoint construction is played for *one checkpoint* and in *two rounds*.

Rules:

Round 1

- Facilitator lays out the seven Alpha Overview cards on the table and describes the checkpoint being considered.(e.g. Ready for Development)
- Each team member decides which alphas should be considered as part of the *checkpoint*
- Team agrees on which alphas should be considered for the *checkpoint*.

Round 2

- for each selected alpha
  - each team member identifies the state he believes the alpha needs to be in to pass the checkpoint
  - different choices are discussed (explain and motivate, starting with extreme ones)

The game is played until consensus is obtained.

Essence games

Checkpoint construction

Facilitator leads the group through a discussion of potential additional checklist items to be added for the checkpoint.

$\Downarrow$

The generic checklist items on the cards can be tailored to the context of the specific endeavor.

By applying the Checkpoint Construction game several times, a whole lifecycle can be defined.

Subiecte tratate

Essence games

Utilizarea nucleului Essence

Utilizare practici

Understanding the Context

Software engineering is a result of recognizing a *problem* or an opportunity.

Sources of problems (examples) :

- related to an existing software system - understanding its original requirements
- related to the stakeholders – they do not have time to involve as needed
- related to team communication – a less experienced developer may not be guided enough by a busy experienced developer.

Example : TravelEssence

TravelEssence – fictitious company, leading travel service provider.provides online hotel booking services for travelers. In addition, TravelEssence provides Software as a Service (SaaS) for the operation of hotels. SaaS means that the owner of the software, in this case TravelEssence, provides software as a service over the internet and the clients pay a monthly fee. Hotels can sign up and use the TravelEssence service to check-in and check-out their customers, print bills, compute taxes, etc.

The opportunity :

If TravelEssence would provide recommendations online through a software solution, it can provide better service to customers, thereby shortening the time customers need to make a purchase decision.

The current system provides different usage scenarios for different kinds of customers (e.g., new travelers, frequent travelers, corporate travelers, agents, etc.). The software system involves a mix of mobile applications and a cloud-based backend.

Smith and his team had been assigned to work on providing a new functionality for TravelEssence, specifically a recommendation engine for travelers.

Context representation with alphas

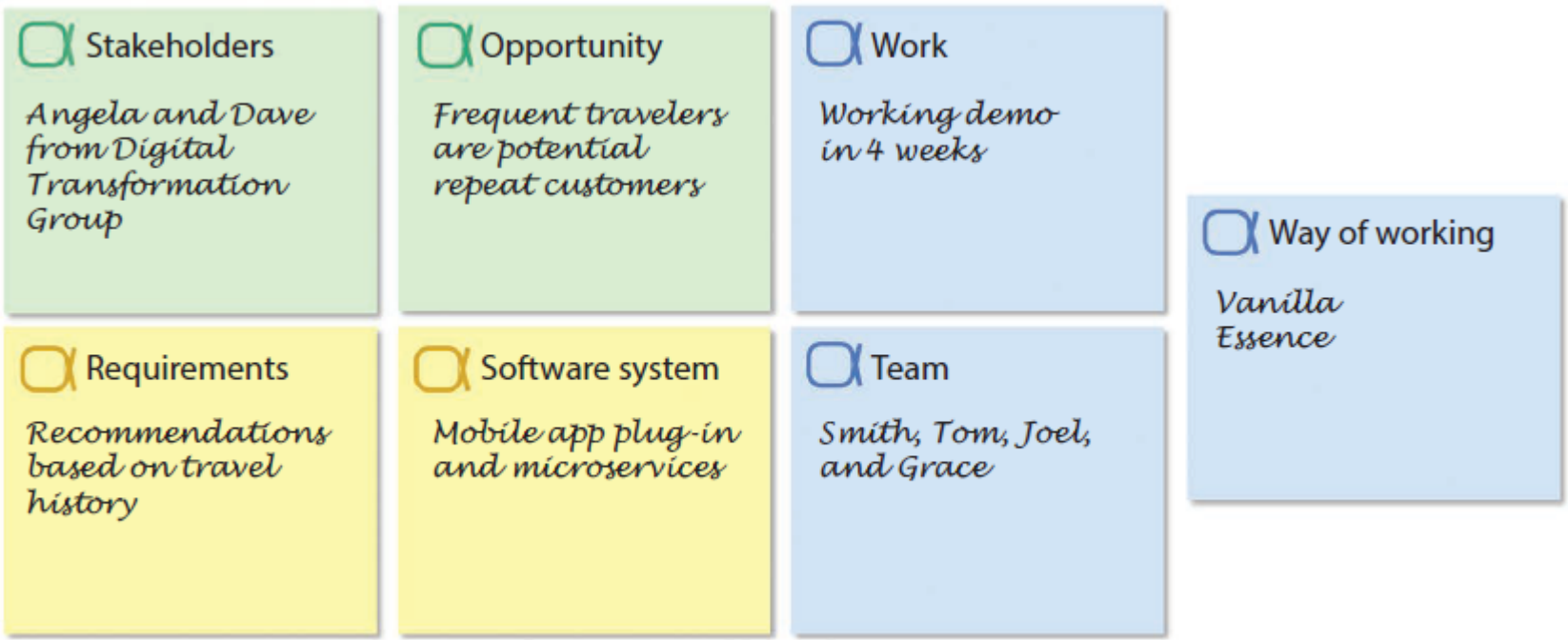


Fig. 9.1 Understand context with Essencee.
The Essentials of Modern Software Engineering

Context representation

Perspective : Customer

Stakeholders:

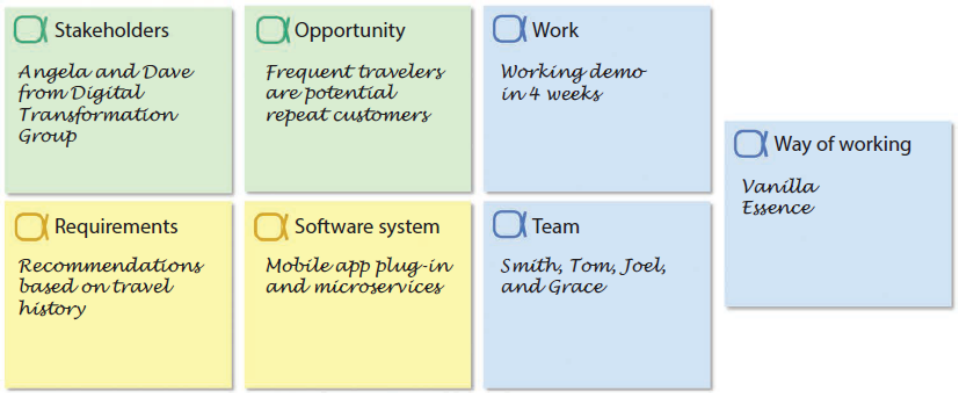
Digital Transformation Group

(Digital Transformation = use of technology to radically improve a company's performance)

- Dave – CDO (Cief Digital Officer)
- Angela – collaborates with Smith and his team

Opportunity :

- exist data about travelers who logged their experience and shared it using social media sites
- TravelEssence can use data from repeat customers to attract new customers.



Context representation

Perspective : Solution

Requirements:

- analysis of traveler data ⇒ identify trends and relationship ⇒ recommend new travel options

Software System :

- a mobile, cloud-based application already exist
- to be developed – a simple plug-in to allow customers to view recommendations

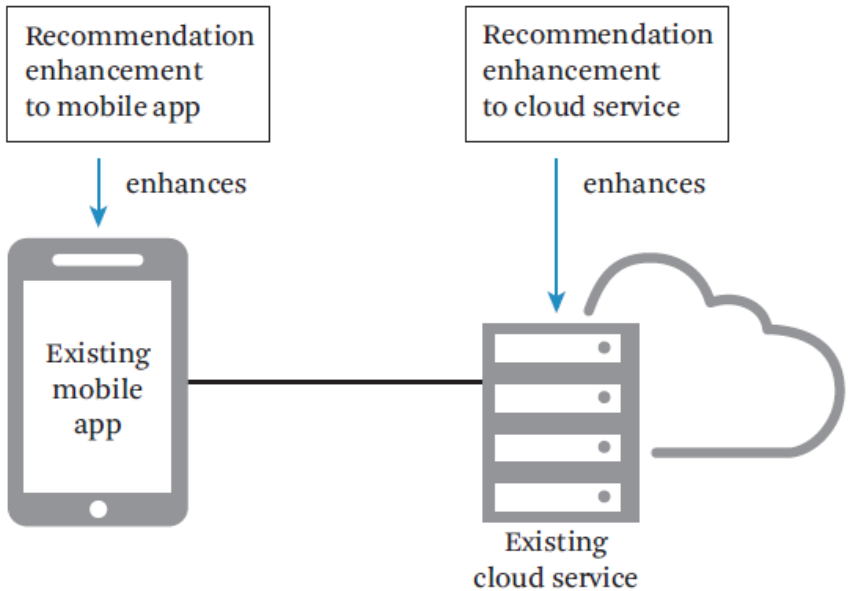


Fig. 9.2 Enhancement to the Software System to achieve recommendations.
The Essentials of Modern Software Engineering

Context representation

Perspective : Endeavor

Work:

- deliver a working demo in one month

Team :

- Smith (team leader),
- Tom, Joel (experienced with mobile app and microservices),
- Grace (experienced only with mobile app)

Way of Working :

- use the facilities of Essence kernel to evaluate progress and health,
- use only alphas, states, checklists $\Rightarrow$ ‘vanilla’ Essence, with no extensions to the kernel.

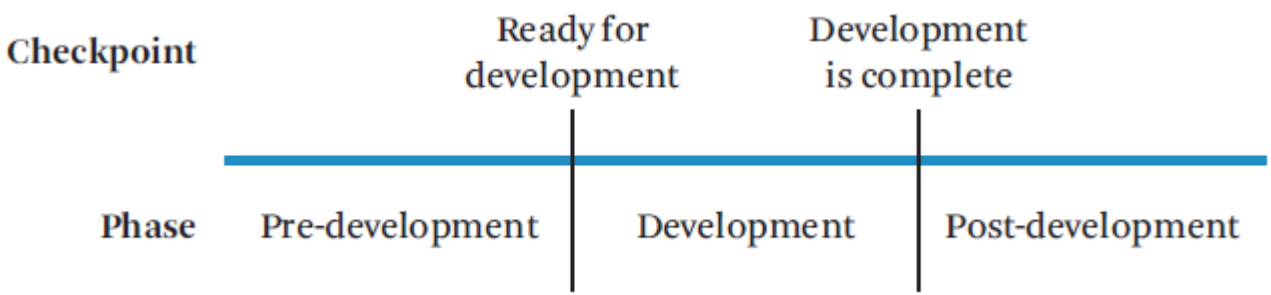
Development Scope and Checkpoints (milestones)

Alphas and their state's checklists – used to gain agreement about

- preconditions for starting development
- criteria for completing development

Checkpoints to be defined

Fig. 9.3 Checkpoints and phases for enhancement of TravelEssence
The Essentials of Modern Software Engineering

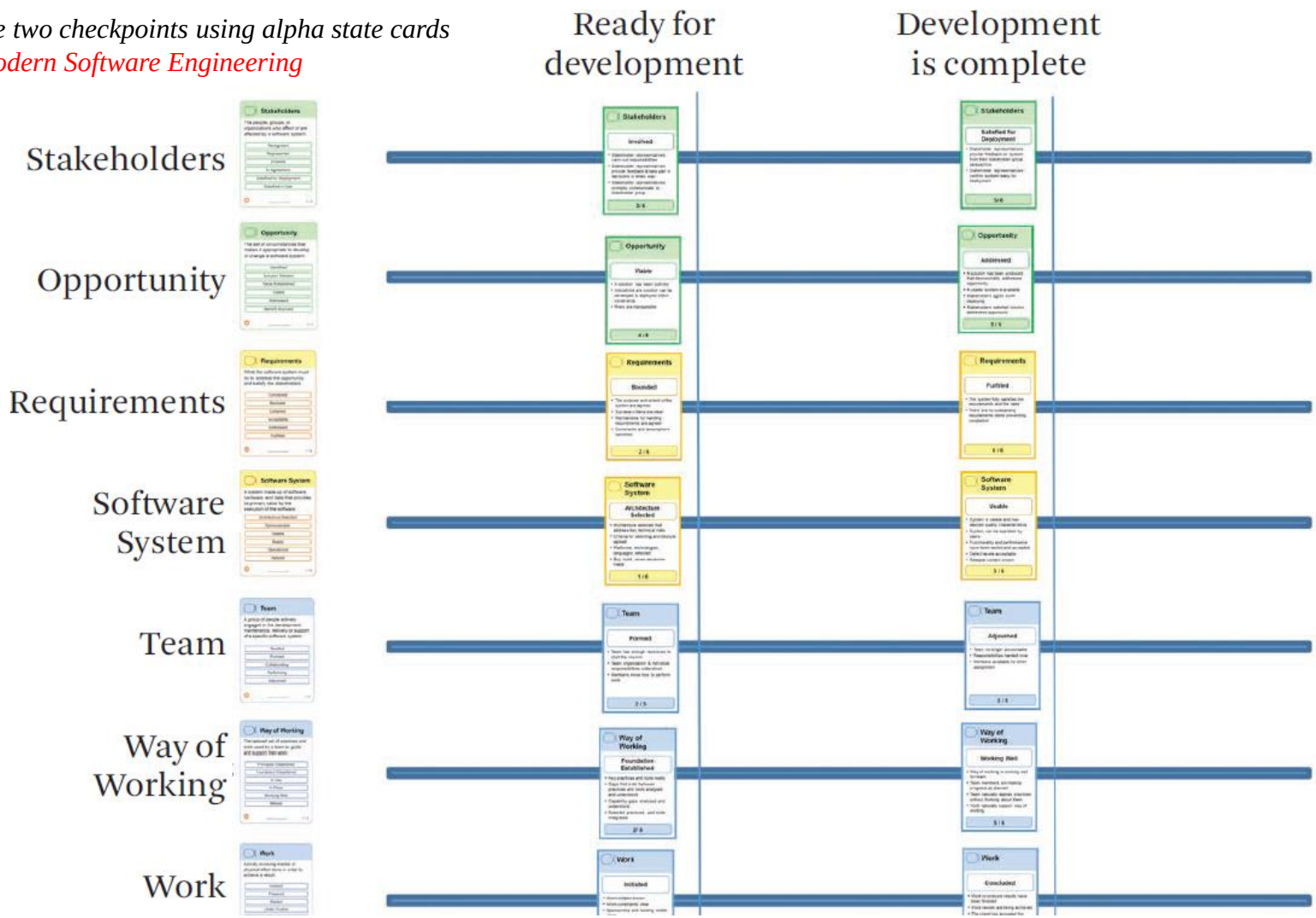


Ckeckpoint Construction game is played – 2 rounds

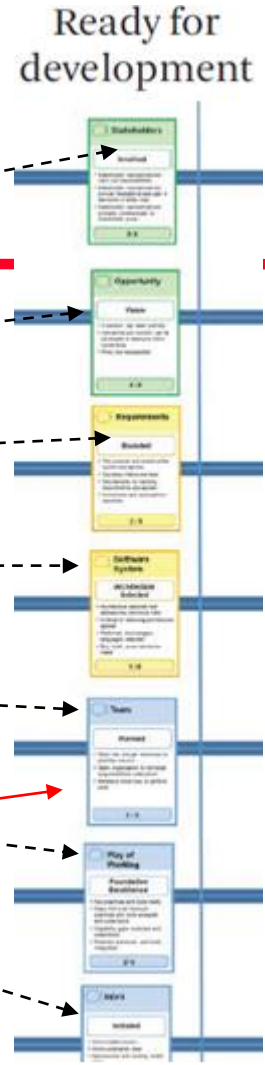
1. selecting the relevant alphas
2. agree on the alpha states in the definition of the checkpoint

Checkpoint Construction game result

Fig. 9.4 Defining the two checkpoints using alpha state cards
The Essentials of Modern Software Engineering



Checkpoint : Ready for development



Stakeholders : *Involved*

Opportunity : *Value Established*

Requirements : *Bounded*

Software System *: Architecture Selected*

Work : *Initiated*

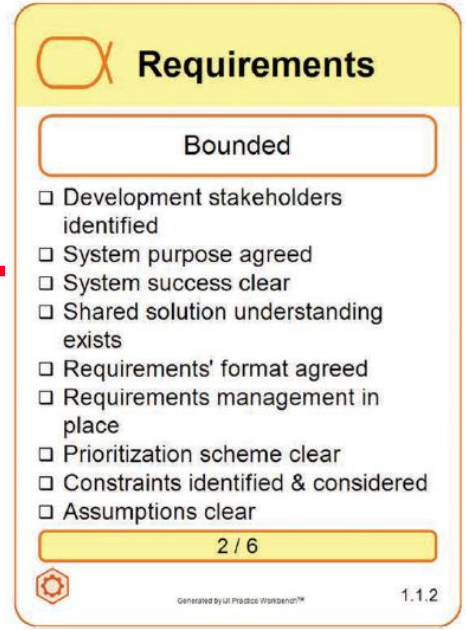
Team : *Formed*

Way of Working : *Foundation Established*

State cards for each alpha contain *checklists*.

Discussion of the checklists on each state card to reach agreement on any additional checklist items.

Agreeing on the most important things to watch



Requirements alpha is not enough for measuring day-to-day progress.

Team agreed to track progress for requirements items, defects, and issues during development, using a spreadsheet.

Obs. The individual requirement item (*Requirement item)* may be defined as a *sub-alpha* of *Requirements alpha*.

Agreeing on the most important things to watch

Chase the State game – determine the current stage of the development

Way of Working : Foundation Established

- iterative development

Requirements : Conceived

- a requirement items list is developed and agreed

⇒ *Requirements : Bounded*

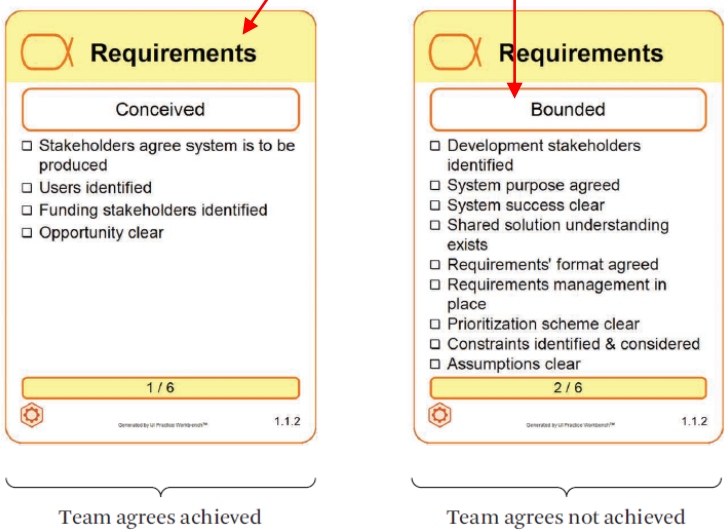


Fig. 9.5 Requirements: Conceived and Bounded state cards
The Essentials of Modern Software Engineering

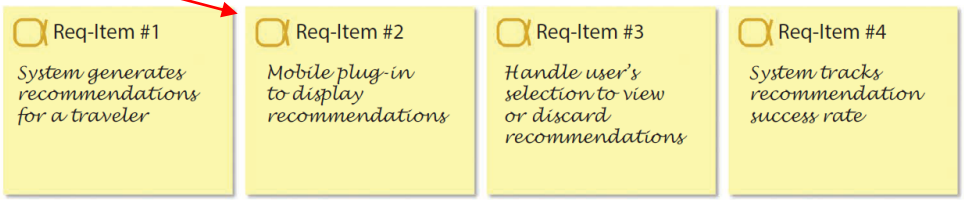


Fig. 9.6 Requirement item list
The Essentials of Modern Software Engineering

Plan-Do-Check-Adapt Cycle

Way of Working : *Foundation Established*

- agile, iterative development

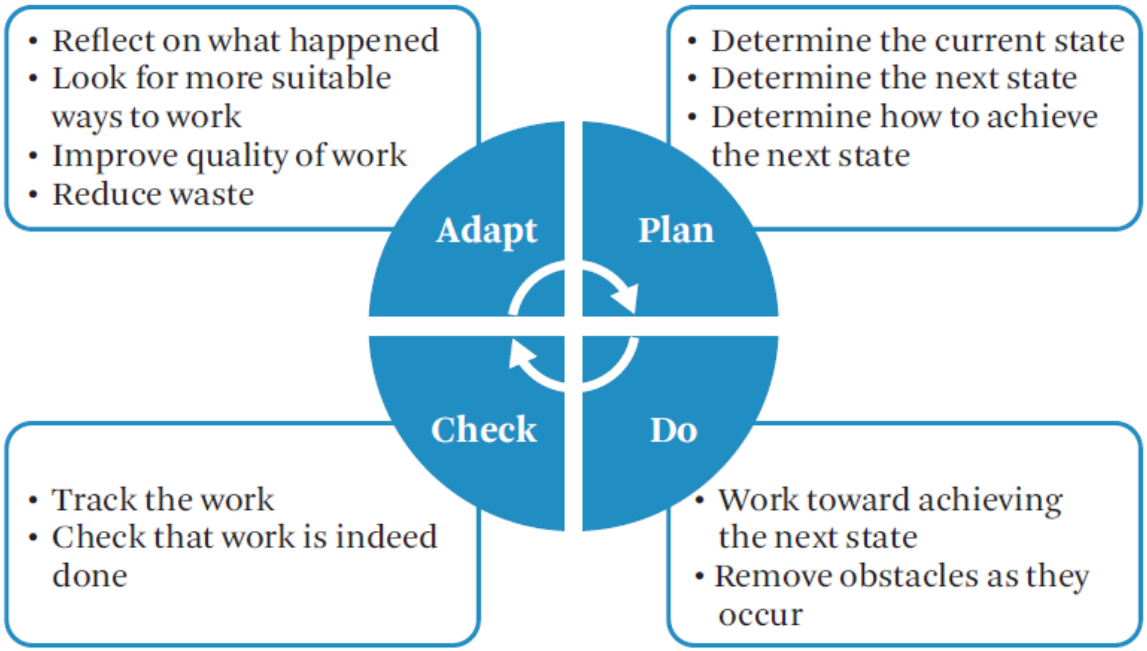
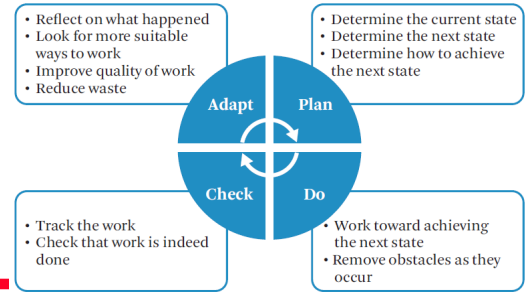


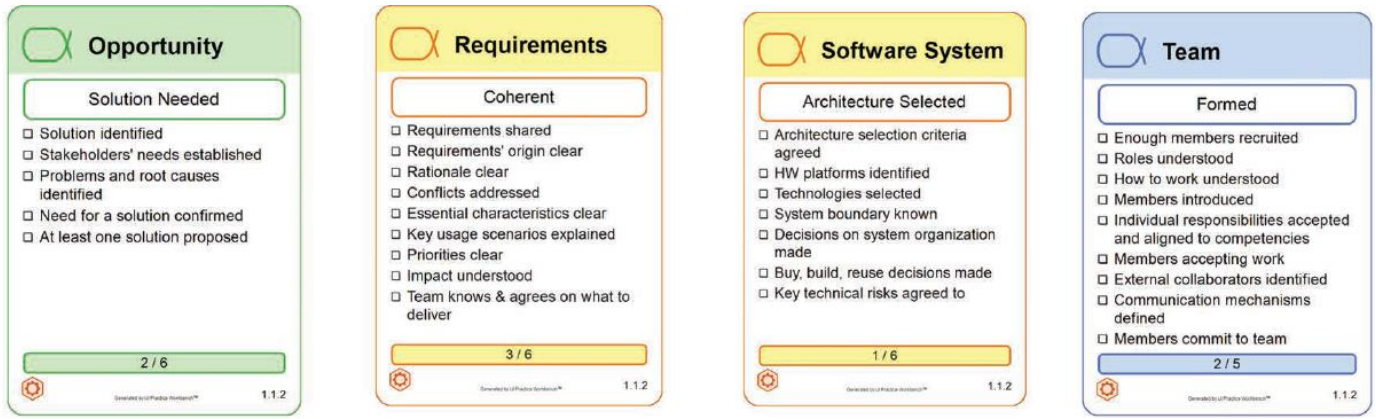
Fig. 10.1 Plan-Do-Check-Adapt cycle
The Essentials of Modern Software Engineering

Plan-Do-Check-Adapt Cycle



1. Determining the current stage :

- Progress Poker for Requirements and Software System alphas
- Chase the State for the rest of alphas



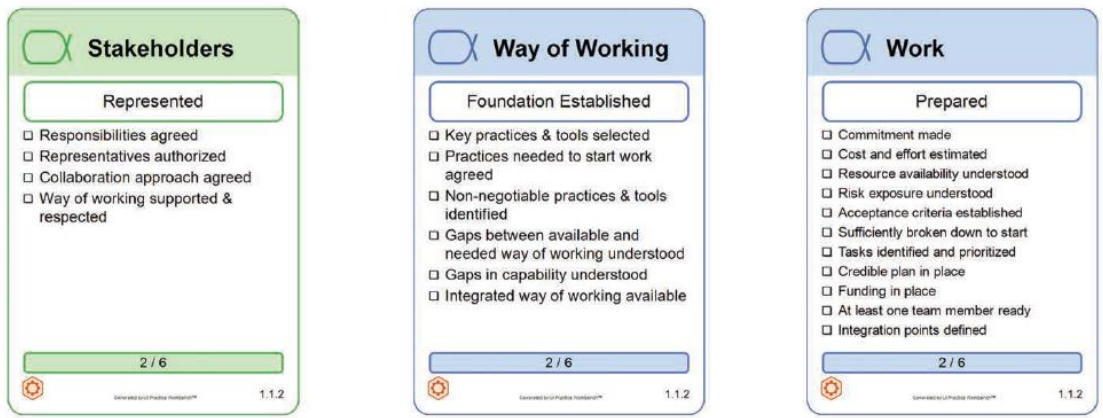
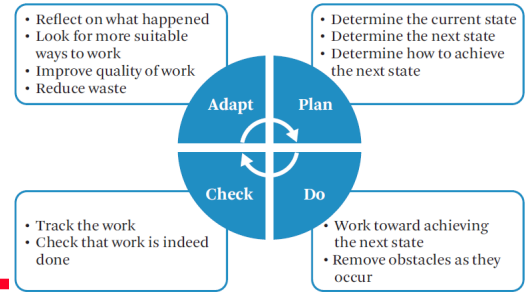


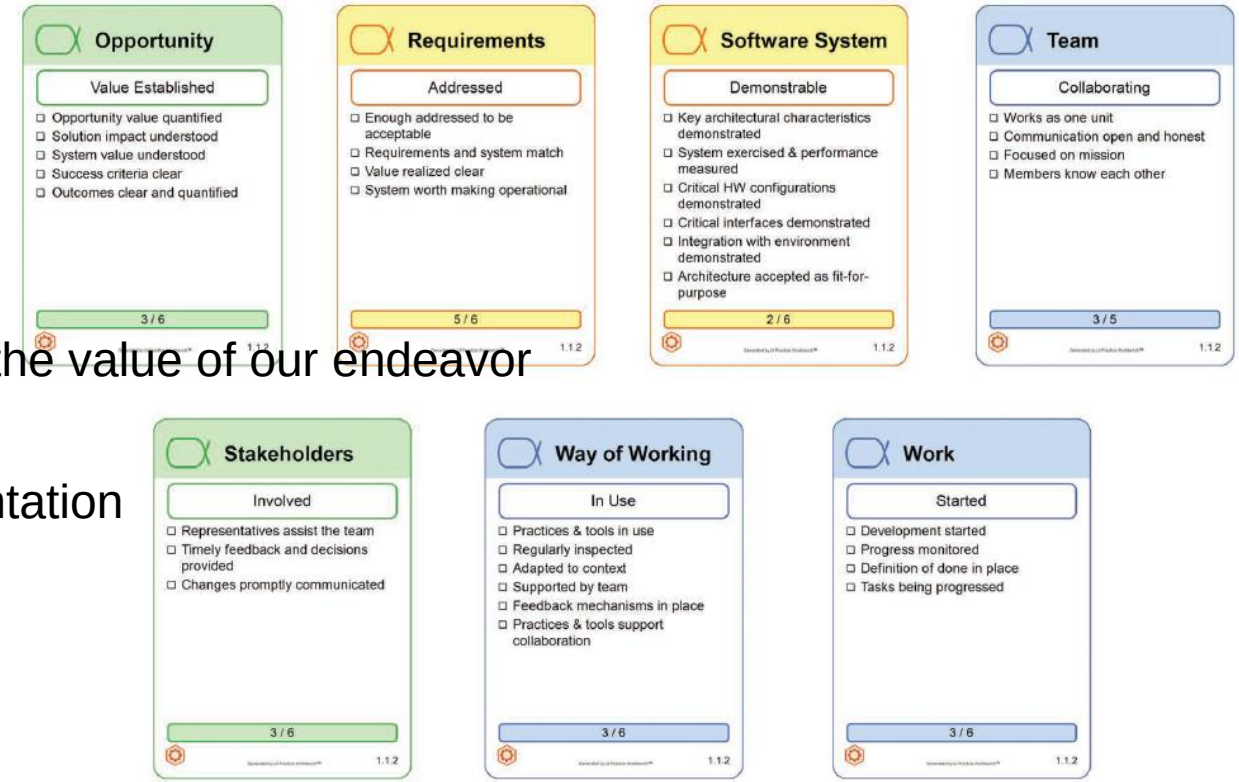
Fig. 10.2 The alpha states agreed
The Essentials of Modern Software Engineering

Plan-Do-Check-Adapt Cycle



2. Determining the objective of the current iteration :

- Objective Go

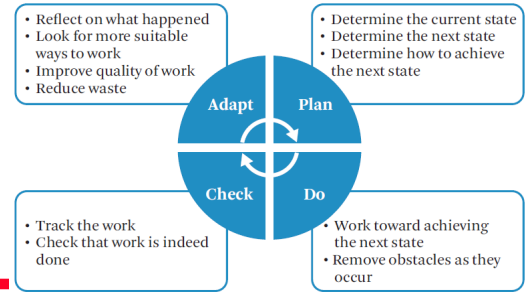


3. Identified needs :

- convince management of the value of our endeavor
- stakeholders involved
- demonstrate the implementation

Fig. 10.3 The results after applying the Objective Go game

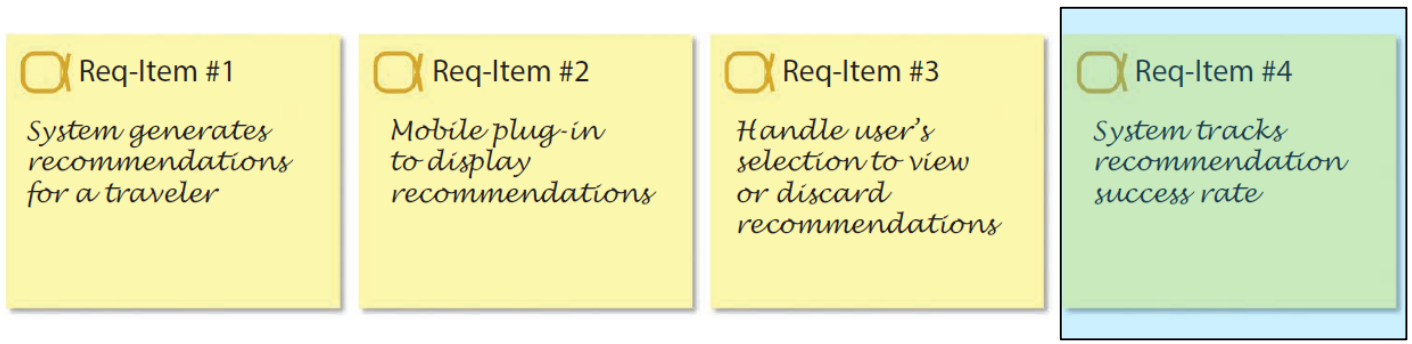
Plan-Do-Check-Adapt Cycle



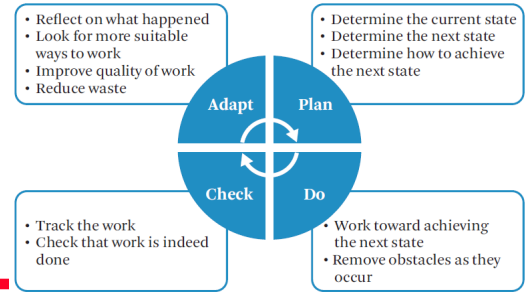
4. Identifying tasks to achieve the states.

Examples :

- set up a meeting with Dave and Angela (stakeholders)
- set up a test environment
- break tasks down further and agree on the pieces to complete in the current iteration (ex. not all requirements but only the first 3)



Plan-**Do**-Check-Adapt Cycle



Do

Working on the identified tasks

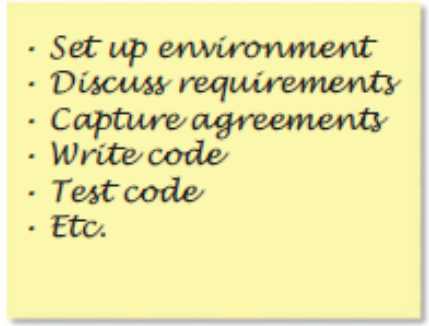


Fig. 10.4 Task list
The Essentials of Modern Software Engineering

Check

Green/red stickers near healthy/unhealthy states

Unhealthy : the checklist is not yet being met or had previously been met, but is no longer met due to some changes

Adapt

Review way of working

Identify obstacles

Find better or more suitable ways to do things

$\Downarrow$

Changes in plans and ways of working

Planning
with Essence
Customer perspective

The curent state and the target state have been identified playing Chase the State and Objective Go.

Identification of the tasks needed to achieve the target state.

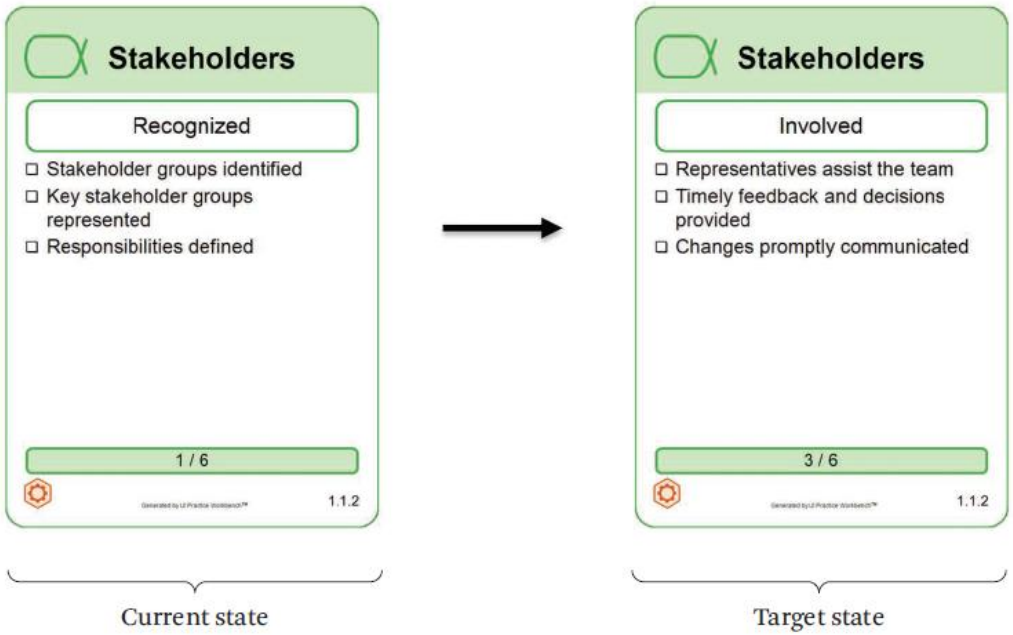


Fig. 10.5 Stakeholders current and target states
The Essentials of Modern Software Engineering

Task : Stakeholder involvement meeting

Planning
with Essence
Customer perspective

Need to convince senior management at TravelEssence to move forward and fund the effort.

To convince about the solution valability the team will set up a test environment where they could quickly experiment with different ideas for using the travelers' existing data.

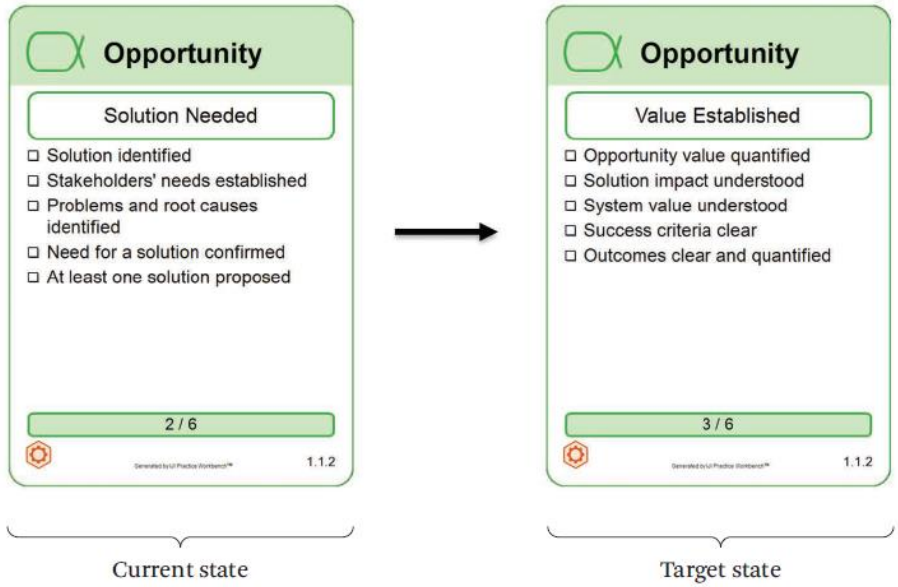


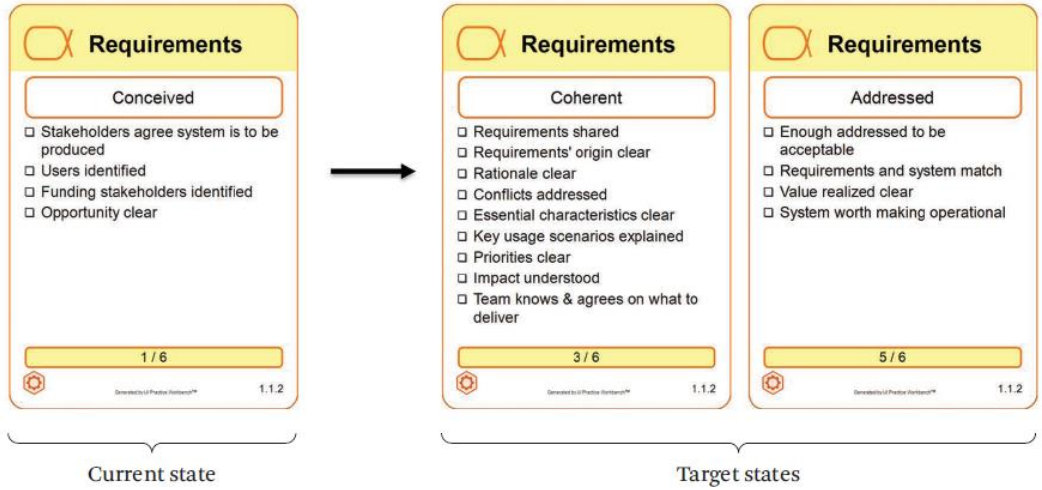
Fig. 10.6 Opportunity current and target states
The Essentials of Modern Software Engineering

Task: Experiment with different ideas to increase business.

Planning
with Essence
Solution perspective

Fig. 10.7 Requirements current and target states
The Essentials of Modern Software Engineering

Obs. It is possible to have more than a target state for each alpha.



Requirements selected to be addressed:

Req-Item #1. System generates recommendations for a traveler

Req-Item #2. Mobile plug-in displays recommendations

Req-Item #3. System handles user's selection to view or discard recommendations

Task: Smith to work with Angela to reach agreement on recommendation algorithm, and which set of travelers they would use as their test data set.

Planning
with Essence
Solution perspective

Code, test and integrate critical parts of the system.

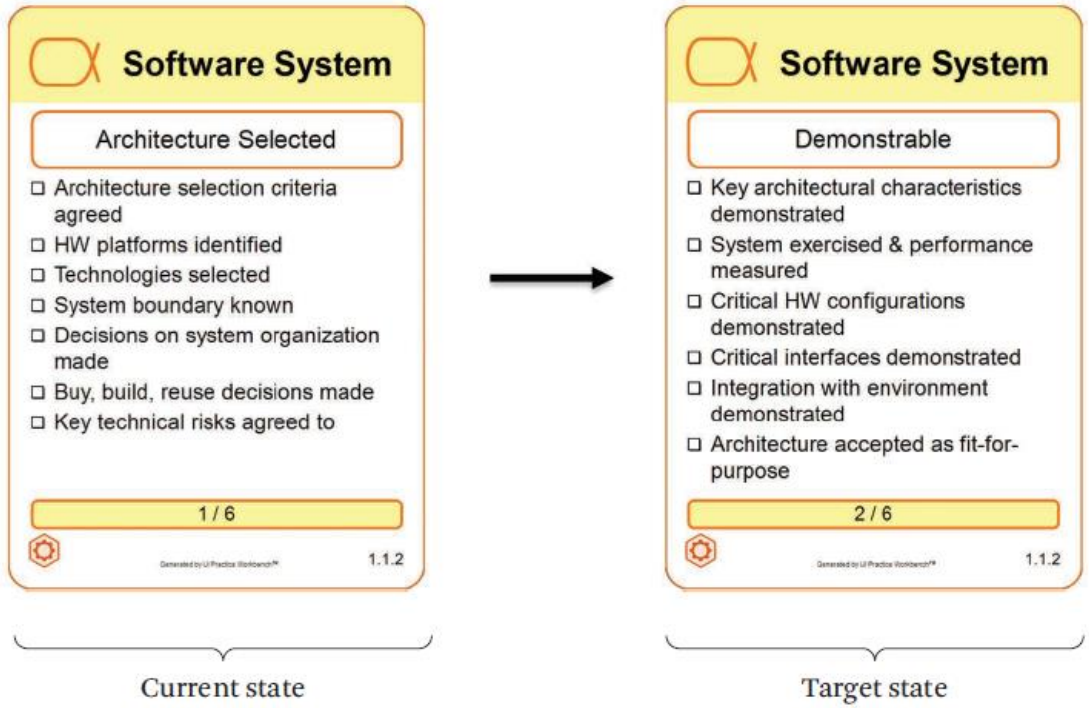


Fig. 10.8 Software System current and target states
The Essentials of Modern Software Engineering

Task: Team members work on implementing their respective requirement items.

Planning
with Essence
Endeavor perspective

As part of **Req-Item #1**, the team had discussed providing recommendations for both hotels and restaurants, but Smith decided this was too much for the first iteration and suggested the team limit the work for now to just providing hotel recommendations.

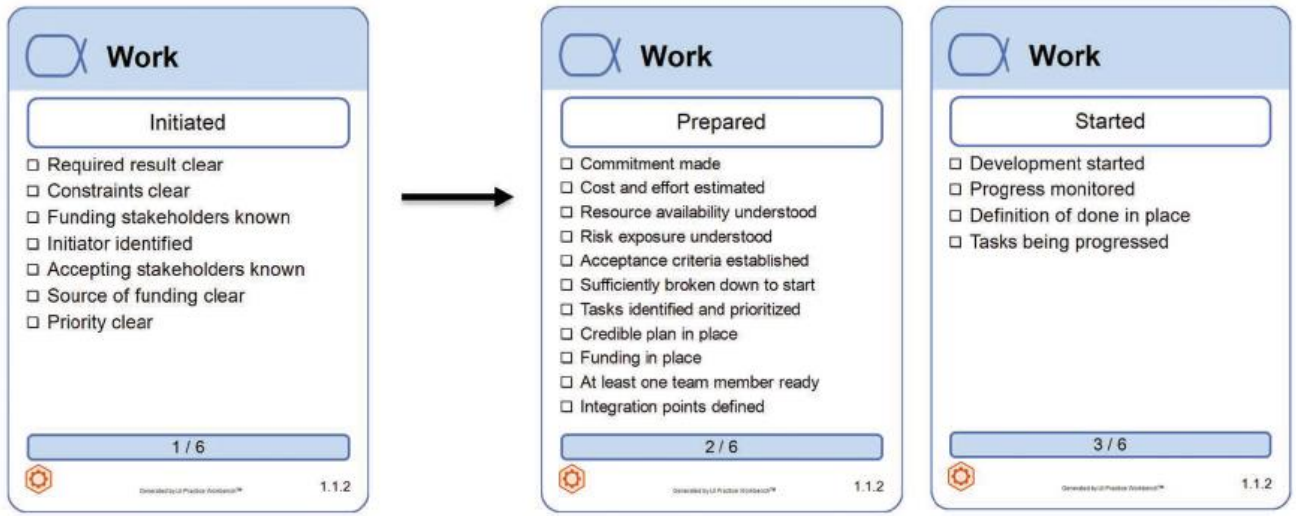


Fig. 10.9 Work current and target states
The Essentials of Modern Software Engineering

Task: Team breaks work down to fit in the planned iteration.

Planning
with Essence
Endeavor perspective

Team members had successfully worked together before. They each knew their responsibilities and how they would work together, but the team had not yet showed that it was working as one cohesive unit.

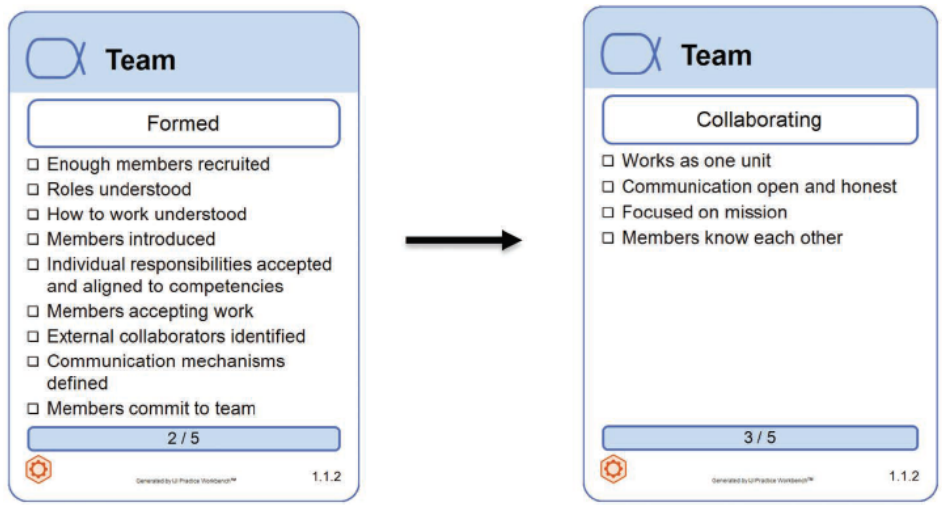


Fig. 10.10 Team current and target states
The Essentials of Modern Software Engineering

Task: Integrate work by Wednesday.

Planning
with Essence
Endeavor perspective

- establish repository version control tool and the test environment.
- prepare the test environment and supporting scripts

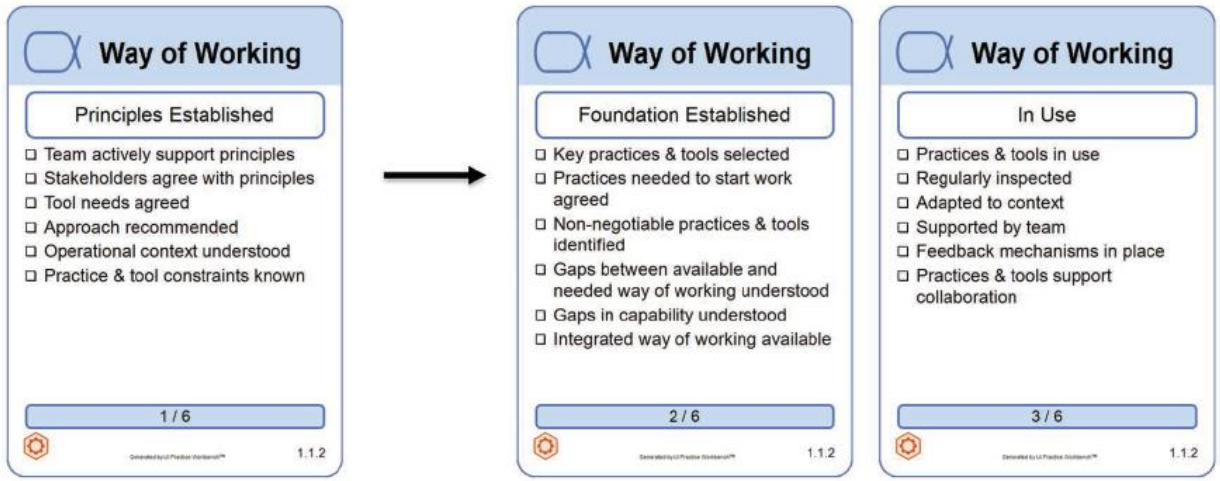


Fig. 10.11 Way of Working current and target states
The Essentials of Modern Software Engineering

Task: Establish development and test environment.

Doing and Checking
with Essence

- Work go smoothly because members were familiar with each other and they can use communication technology if they re not collocated (ex. Angela is in another room)
- Work finished on Friday afternoon
- The implementation has been demonstrated to Angela
- Health and progress reviewed by playing Chase the State

Customer perspective

- *Stakeholders : Involved*

In a meeting, after Friday demo, Angela and Dave agreed to their involvement in future demonstrations.

- *Opportunity : Value Established*

Friday demo was successful and convinced about its business value. Dave accepted to fund the effort.

Doing and Checking
with Essence

Solution perspective

- *Requirements : Coherent, Addressed*

Open issues related to the agreed requirements have been clarified, and then addressed in the Friday demo.

- *Software System : Demonstrated*

Successful demonstration of the critical parts of the system agreed for the Friday demo.

Endeavor perspective

- *Work : Prepared, started*

Agreed tasks have been broken, risks understood, code developed, tested and integrared, prepared for demo.

- *Team : Collaborating*

Successful integration on Wednesday for the Friday demo $\Rightarrow$ team was working as a consistent unit.

- *Way of Working : Foundation Established, In Use*

Environment has been set up and used during the first iteration.

Adapting Way of Working
with Essence

Using *Essence kernel* helped to :

- involve key stakeholders
- think about the opportunity
- break the work down to fit in the agreed way of working
- think about risks
- clarify requirements
- integrate each team member's work with teammates' work
- focus on the most important things first

But *improvements can exist.*

Adapting Way of Working
with Essence

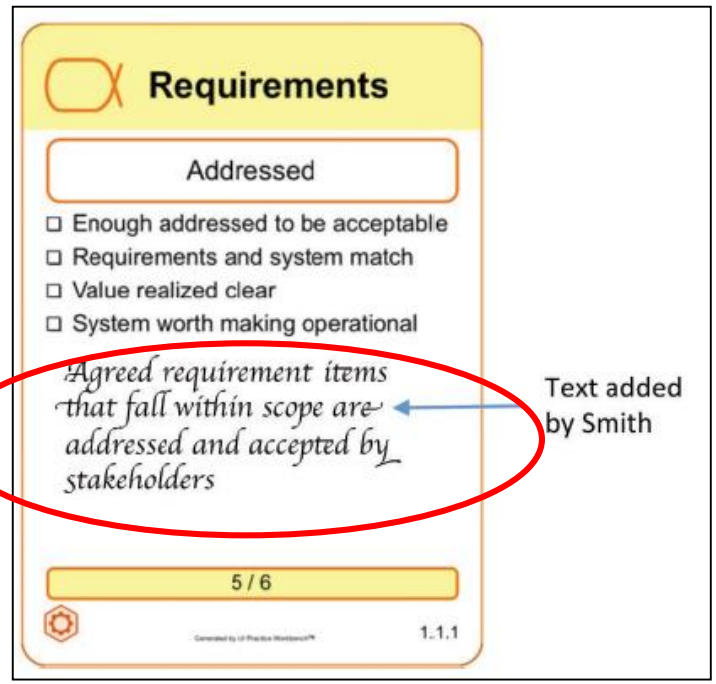
Discussion on the target of the just finished iteration

- What went well with our planning, doing, and checking related to the above alpha states?
- What did not go well with our planning, doing, and checking related to these alpha states?
- What can we do better with our planning, doing, and checking related to the alpha states?

Example :

Tom said, “The way to achieve the *Requirements: Addressed* state was not clear to me at the start of the iteration. I learned that I had to talk to Angela and get her to agree to the requirement items to be implemented. I didn’t understand this just by looking at the state checklist.”

Improvement : additional guidance with new item added to the check list of *Requirements Addressed* state



Adapting Way of Working
with Essence

Essence kernel helps adapt way of working :

- Target states become visible and the team reason about their way of working in terms of a process they have followed
- Kernel is extensible (eg. add items to alpha state cards, define new alphas, add checklists) allowing to make changes to improve way of working.

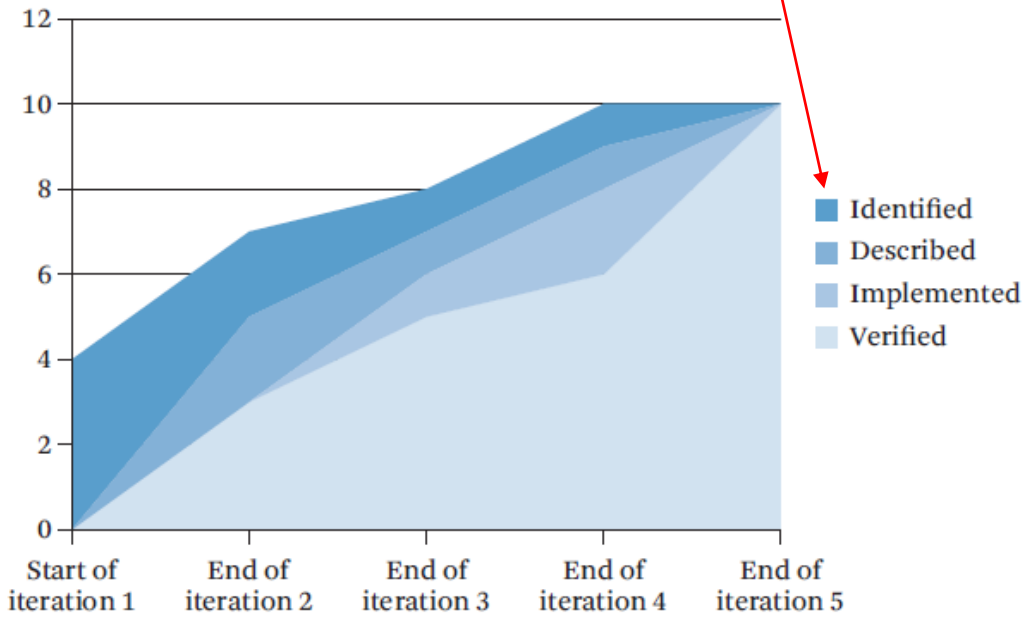
Software engineering is essentialized by representing the way a team is working using the Essence language and the Essence kernel common ground

Analysis

Visual display of how the requirements items progress.

States of the Requirements Item sub-alpha

- The list of the requirements items is not static.
- Items were added at the end of each iteration.
- Some items were verified in the first iteration.



By introducing ***Requirement-item*** *sub-alphas* the team can track more *accurately* the progress of each requirement item and thus of the requirements as a whole.

Fig. 11.1 Cumulative flow diagram
The Essentials of Modern Software Engineering

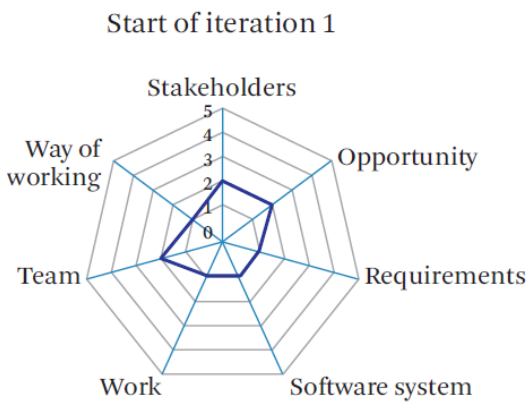
Analysis

	Start of Iteration 1	End of Iteration 1	End of Iteration 2	End of Iteration 3	End of Iteration 4	Target
Stakeholders	1	3	4	4	5	5
Opportunity	2	3	3	3	3	3
Requirements	1	2	5	5	6	6
Software System	1	2	2	4	4	4
Work	1	3	4	4	5	5
Team	2	3	4	4	4	4
Way of Working	1	3	3	4	5	5

Note: The numbers in the table indicate the achieved state.

Table. 11.1 Kernel state evolution
The Essentials of Modern Software Engineering

- States of the alphas do not always move forward linearly.
- Alphas are interdependent so they progress in waves.
- Alphas *need to progress in balance*.
- If a particular axis on the radar diagram is progressing slower, this needs to consider which is the problem and to find solutions.
- Parallel progressions in waves are critical for the success of the endeavor.



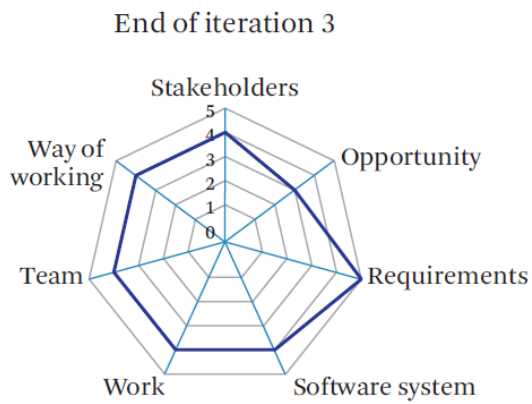


Table. 11.2 Radar diagrams
The Essentials of Modern Software Engineering

Dealing with anomalies

Problems may appear and cause endeavor to fall back compared to a previously reached alpha state.

Examples of possible problems :

- stakeholders may stop participating
- seasoned team members may leave
- new members, with less experience, may come

Solution :

Team should periodically use the Essence kernel *alphas* to provide health check.

Subiecte tratate

Essence games

Utilizarea nucleului Essence

Utilizare practici

The new current development context

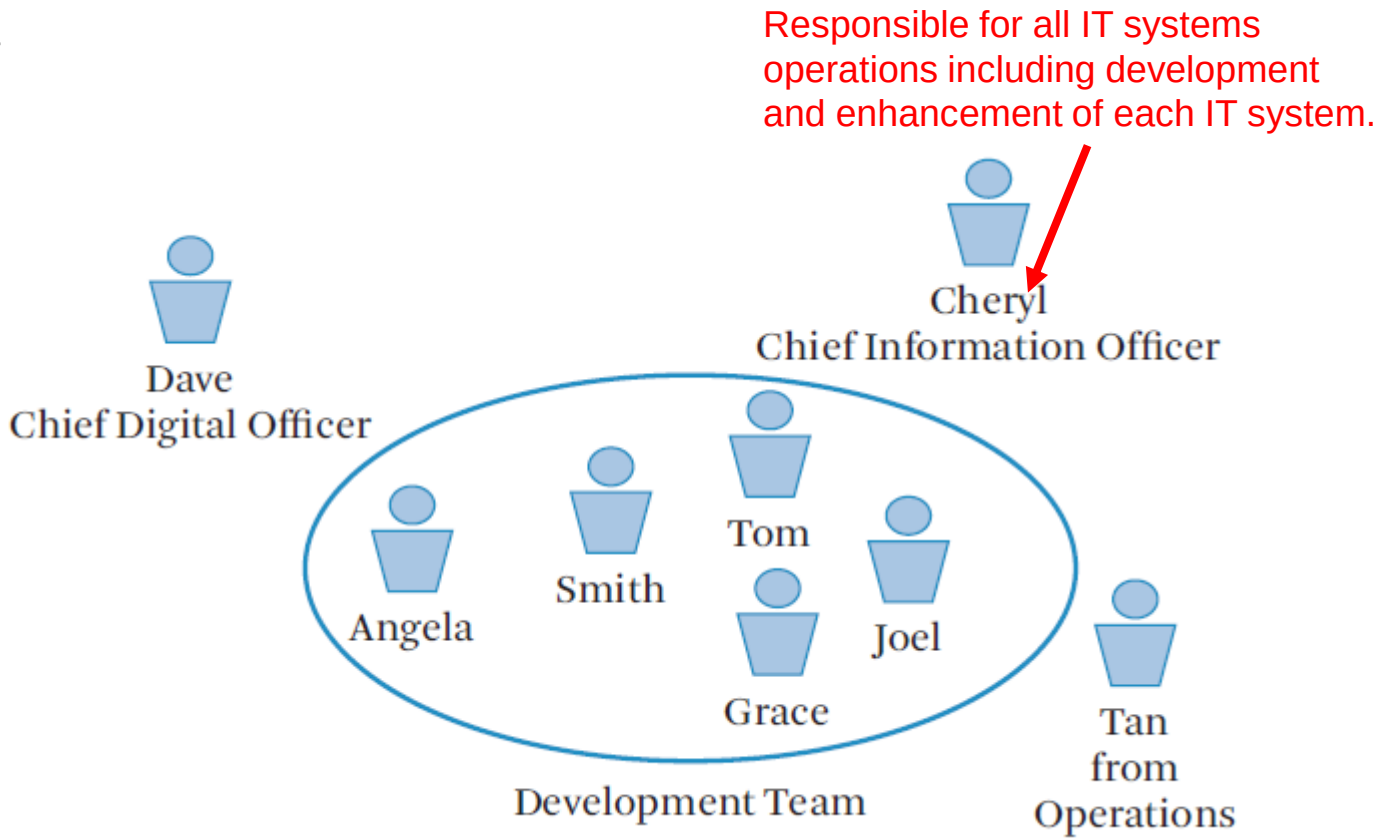
After the team's successful demonstration.

⇓

Decision to expand the scope and vision of the endeavor.

⇓

People involved.

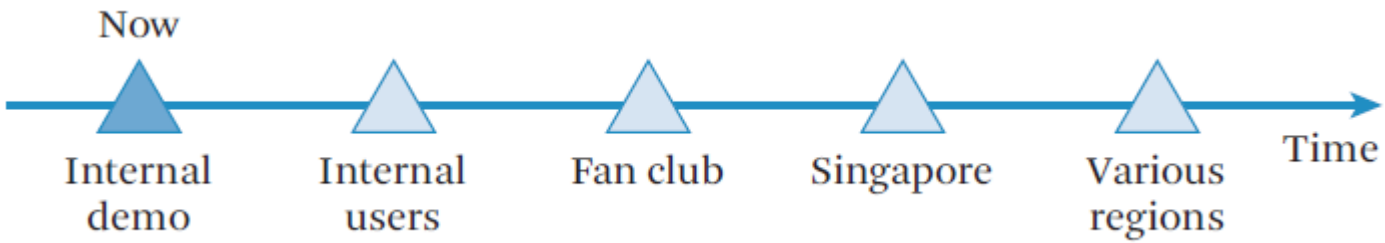


Current development context

Alpha	State achieved	Rationale for achieving the state
Stakeholders	Involved	Cheryl, Dave, and Angela are key stakeholders in the endeavor. The state is achieved because they were actively involved in helping the team achieve a successful demo.
Opportunity	Value Established	The team had a successful demo supporting the objectives of the Digital Transformation Group.
Requirements	Bounded	The team had successfully gotten the key stakeholders involved and those key stakeholders had reached a shared understanding of the extent of the proposed solution.
Software System	Architecture Selected	They had made their decision to use the existing proven mobile app, and to use an architecture approach referred to as microservices to host their recommendation engine.
Work	Initiated	All the team members had agreed that the source of their funding, and the stakeholders who would fund the work, were clear.
Way of Working	Working Well	Initially tacit agreed practices worked well for the team, but as we shall see the team eventually evolved to the more explicit practices of Scrum, User Story, Use Case and Microservice due to changes in their endeavor as it progressed.

Development Scope and Checkpoints

Agreement on incremental development ⇒ release roadmap.



The next increment : to finish with the release to internal users.

Games to play Ș

- Chase the State
- Objective Go **?**

Playing games results

Example of results after playing Chase the State and Objective Go:

Stakeholders : Involved $\rightarrow$ Recognized $\rightarrow$ Involved (New stakeholders appeared).

Opportunity : Value Established $\rightarrow$ Value Established

Requirements : Bounded $\rightarrow$ Conceived $\rightarrow$ Coherent, Addressed (new requirements, beyond that for the demo, added for the next iteration).

Software System : Demonstrable $\rightarrow$ Usable

Work : Started $\rightarrow$ Initiated $\rightarrow$ Prepared (for the new increment; have been defined scope, plan and schedule for the initial release).

Team : Collaborating $\rightarrow$ Performing (areas for which they could improve both their practices and tools; possible new members)

Way of Working : In Use $\rightarrow$ Principles Established $\rightarrow$ Foundations Established, In Use (miscommunications in the process of conducting certain activities have been detected $\Rightarrow$ agreed that they would need more explicit practices to make sure everyone understood and agreed to how the team conducted these activities).

Agree upon practices to apply

Essence kernel contains the ***universal alphas***

Practice – has ***specific alphas***,

There must be correctly identified :

- practice-specific alphas
- practice-specific alpha states
- correspondent checklists

Agree upon practices to apply

Select explicitly a practice to apply.

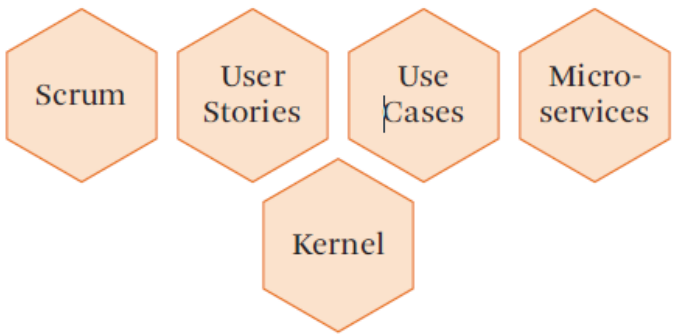
We assume that :

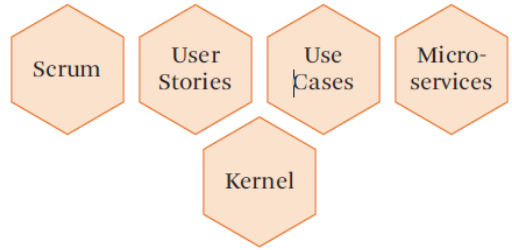
- a library of practices exists
- the team is able to select the appropriate practices
- some practices have been tested and accepted by the organization as working well.

The team will use a mixture of collaboration, engineering, and technical practices.

Example :

Practices selected by the team to be used in addition to the Essence kernel.





Agree upon practices to apply

- Scrum is about team collaborations.

Scrum is a practice for iterative development, each iterationbeing a sprint. The sprint is an alpha, something we need to watch. Scrum guides teams to complete work items in a backlog. These work items, known as Product Backlog Items (PBIs) using Scrum terminology, can also be treated as alphas.

- User stories and use cases are requirements engineering practices

User Stories is a practice about succinctly expressing requirement items, focusing on values. Specific user stories can also be viewed as sub-alphas, similar to treating Requirement Items as sub-alphas of Requirements.

Use Cases is a practice that helps teams identify and organize requirements in the form of use cases and use case slices. A use case slice is a part of a use case that is broken down to an appropriately sized piece of work for the development team to tackle. Specific use cases can be viewed as sub-alphas of Requirements and specific use case slices can be viewed as sub-alphas of a specific use case.

- Microservices is a highly technical practice

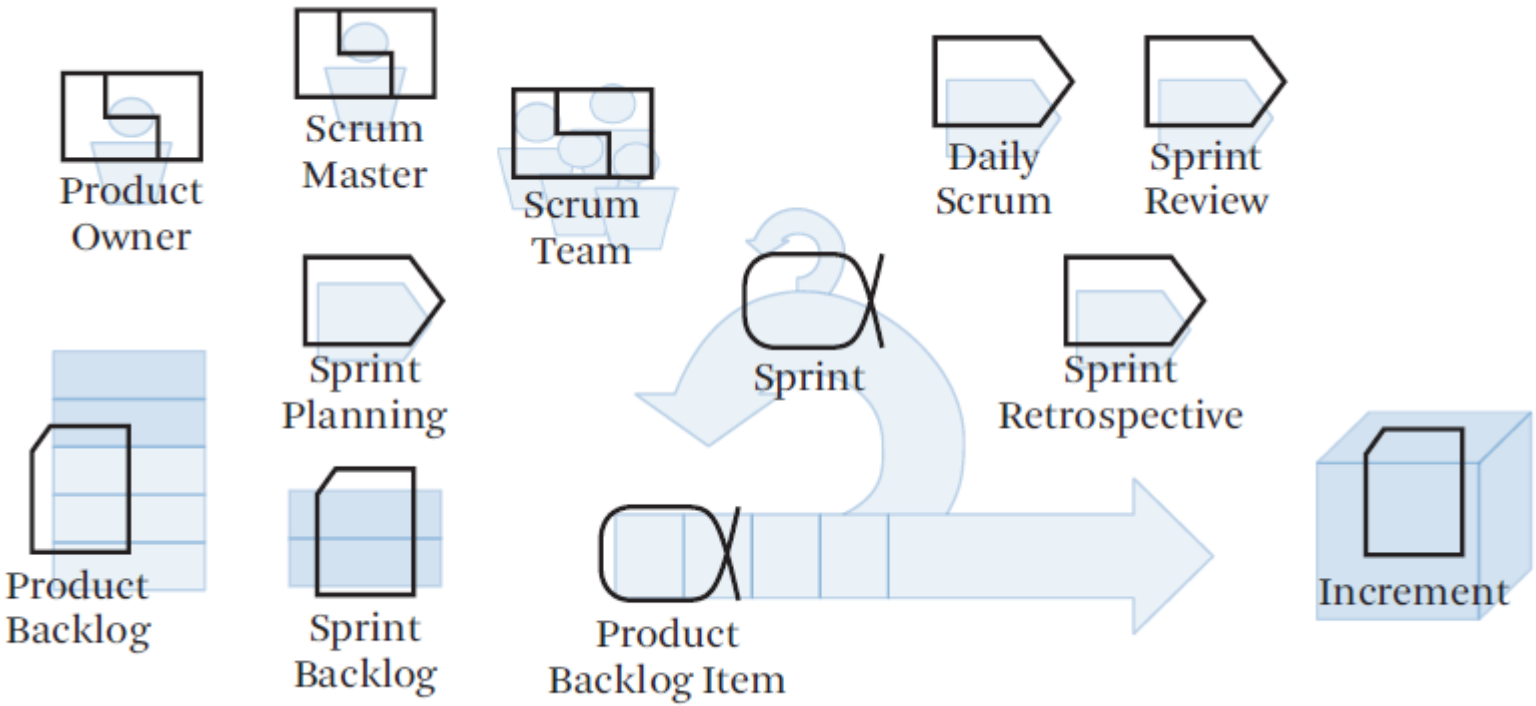
Microservices is a practice that helps teams break down a complex software system into a set of cooperating small independent modules, each with its own purpose and each with its own well-defined interface to other modules. Specific microservices can be viewed as sub-alphas of the Software System kernel alpha and monitored as alphas.

The (sub)-alphas specific to the selected practices

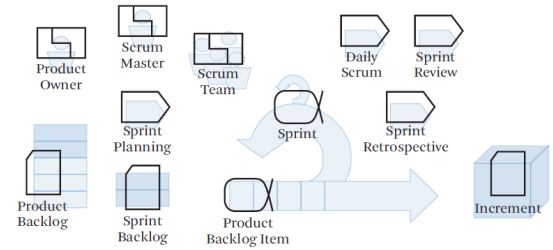
Practice	Description	Alphas
Scrum	A practice for the iterative development of software systems working off a backlog.	Sprint Product Backlog Item
User Stories	A way to capture functionality that will be of value to a user of a software system.	User Story
Use Cases	All of the ways of using a system to achieve a particular goal for a particular user.	Use Case Use Case Slice
Microservices	A software architecture style that uses small independent processes to communicate.	Microservice

Scrum practice

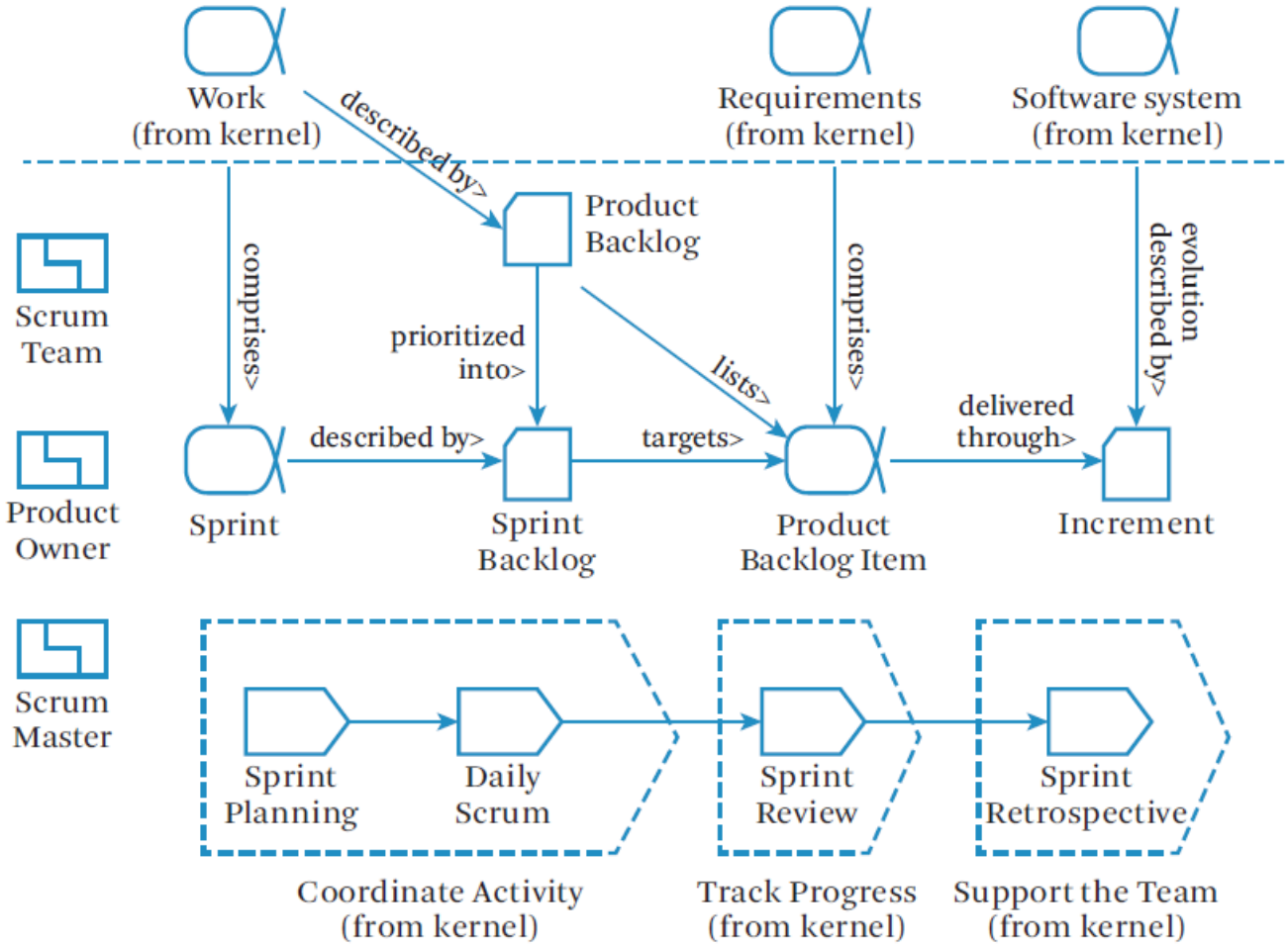
Scrum big picture mapped to the Essence Language



Scrum practice



Scrum Lite practice expressed in the Essence language.



Scrum-specific alphas and work products

Product Backlog

An ordered list of everything that might be needed in the product. The single source of requirements for any changes to be made to the product.

Items Ordered

Describes: Requirements

03.2015

Sprint Backlog

The set of Product Backlog Items selected for the Sprint to meet the Sprint goals. The Sprint Backlog makes visible all of the work the Scrum team identifies as necessary to meet the Sprint goal.

Goals Specified

Capacity Described

Work Forecast Described

Describes: Sprint

03.2015

Product Backlog Item

A change to be made to the product in a future release (for example a feature, function, requirement, enhancement, or fix).

To Do

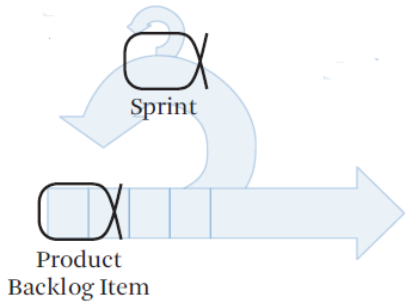
Ready

Doing

Done

Relates to: Requirements

03.2015



Increment

The sum of all Product Backlog Items completed during a Sprint and value of the Increments of all previous Sprints.

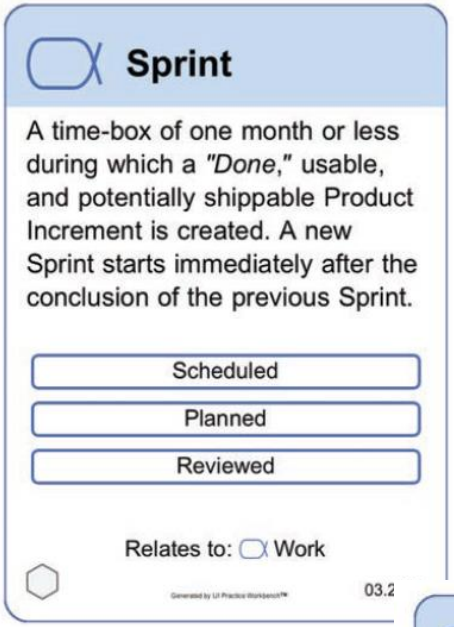
Completed Product Backlog Items Listed

Increment Notes Described

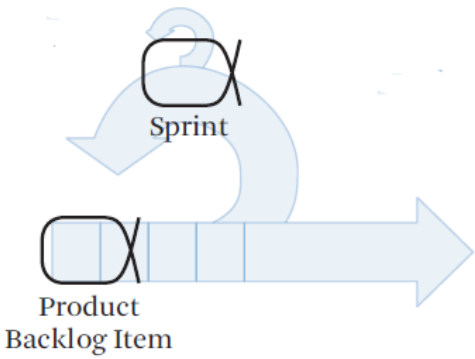
Describes: Software System

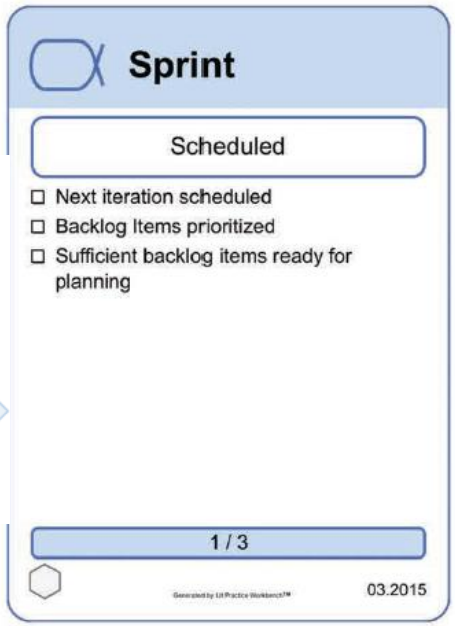
03.2015

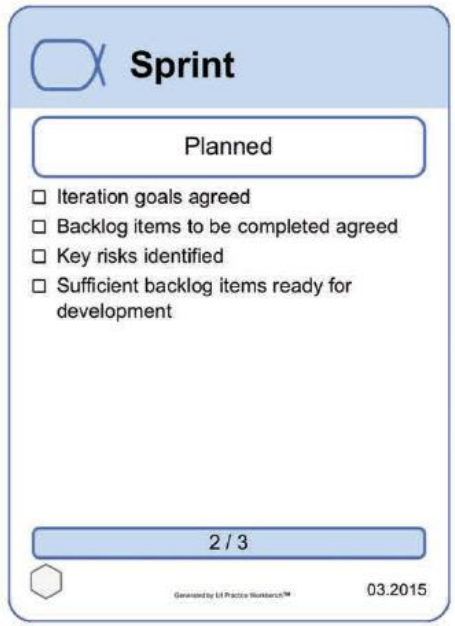
Scrum-specific alphas and alpha states

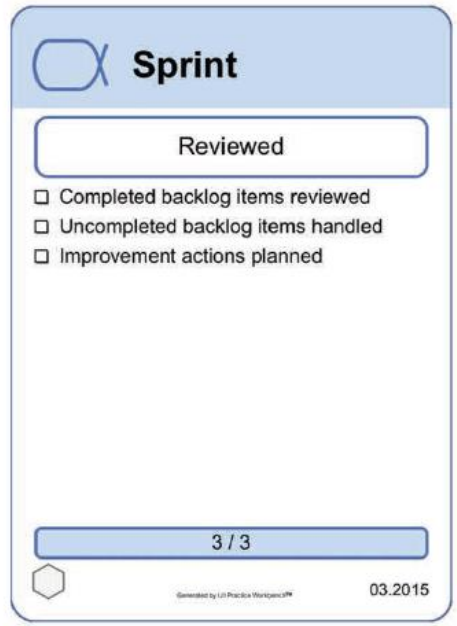


To Do. It has been agreed that the PBI needs to be completed within the next sprint. The scope and completion criteria of the PBI are clear.
Ready. The team works together with the product owner to agree on how they should go about completing the PBI.
Doing. At this state, the team is working on the item and bringing it to completion.
Done. The Product Backlog Item has been completed.









Scrum-specific role patterns

Product Owner

The Product Owner is the sole person responsible for managing the Product Backlog.

Accountable for ensuring:

- The Product Backlog items are clearly expressed
- The Product Backlog is ordered, transparent, and visible to the Scrum team.
- The Scrum team understands the Product Backlog items
- The value generated by the Scrum team is optimized

03.2015

Scrum Master

The Scrum Master is responsible for ensuring that Scrum is understood and enacted. He/she is a servant leader for the Scrum team.

Amongst other things, he/she helps:

- Facilitate Scrum activities
- Remove impediments
- Ensure tean members understand Scrum
- Promote agility

03.2015

Scrum Team

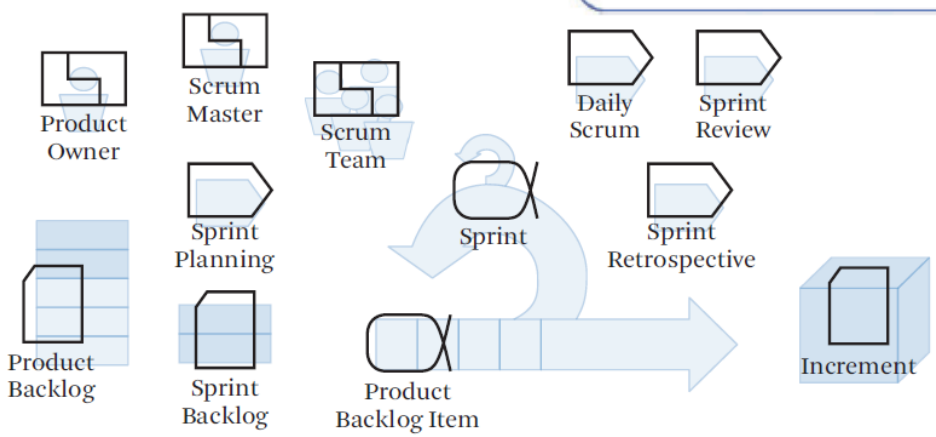
The Scrum team consists of a Product Owner, a Scrum Master, and other members, usually developers and testers.

Scrum teams deliver products iteratively and incrementally, maximizing opportunities for feedback.

Scrum teams are:

- Self-organizing
- Cross-functional

03.2015



Scrum-specific activity cards

Sprint Planning

Decide what can be delivered in the Sprint's Increment and how the work needed to deliver the agreement will be achieved.

Sprint: Scheduled

Coordinate Activity

Leadership Management

Work: Started

Sprint: Planned

Sprint Backlog: Work Forecast Described

03.2015

Sprint Review

A time-boxed review of the outcomes of the Sprint to gather feedback and discuss what should be done next.

Product Backlog: at any level

Sprint: Planned

Track Progress

Development Management

Increment: Product Backlog Items Listed

Sprint: Reviewed

03.2015

Daily Scrum

The team meets every day, same time and place, to assess progress, synchronize activity, and raise and action impediments. The meeting is time-boxed, typically to 15 minutes.

Coordinate Activity

Leadership Management

Work: Under Control

03.2015

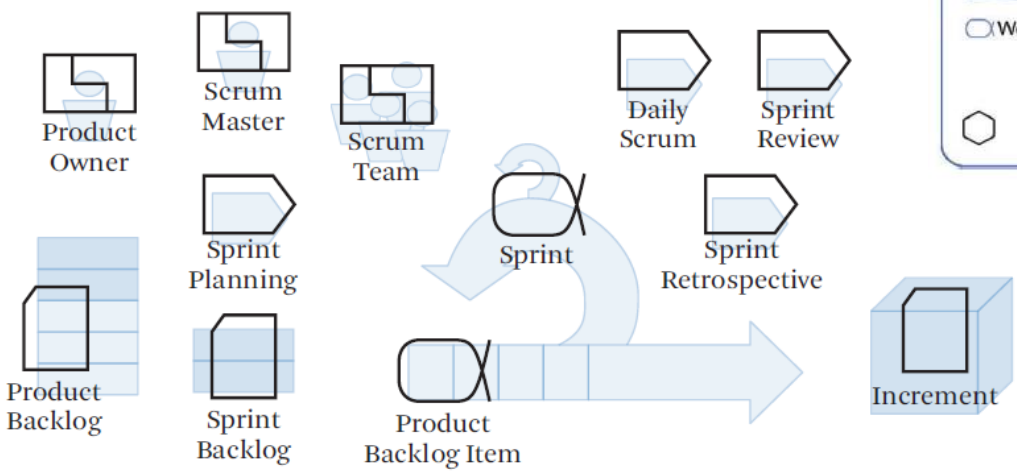
Sprint Retrospective

The whole team meets regularly to reflect on its way of working. Improvements are identified and prioritized, and actions agreed. At the next retrospective the results are evaluated.

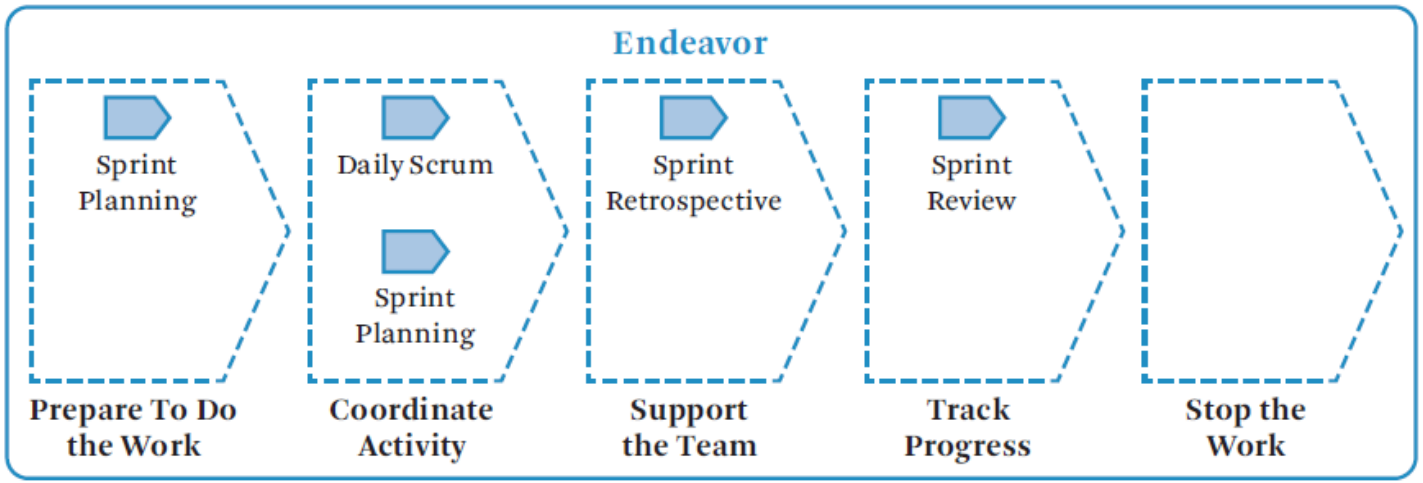
Support the Team

Leadership Management

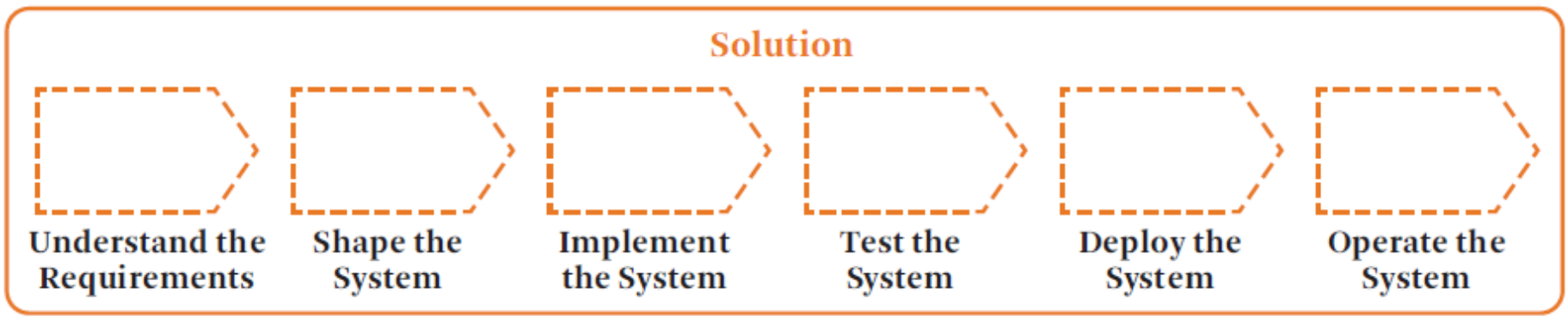
03.2015



Activity spaces addressed by Scrum Lite



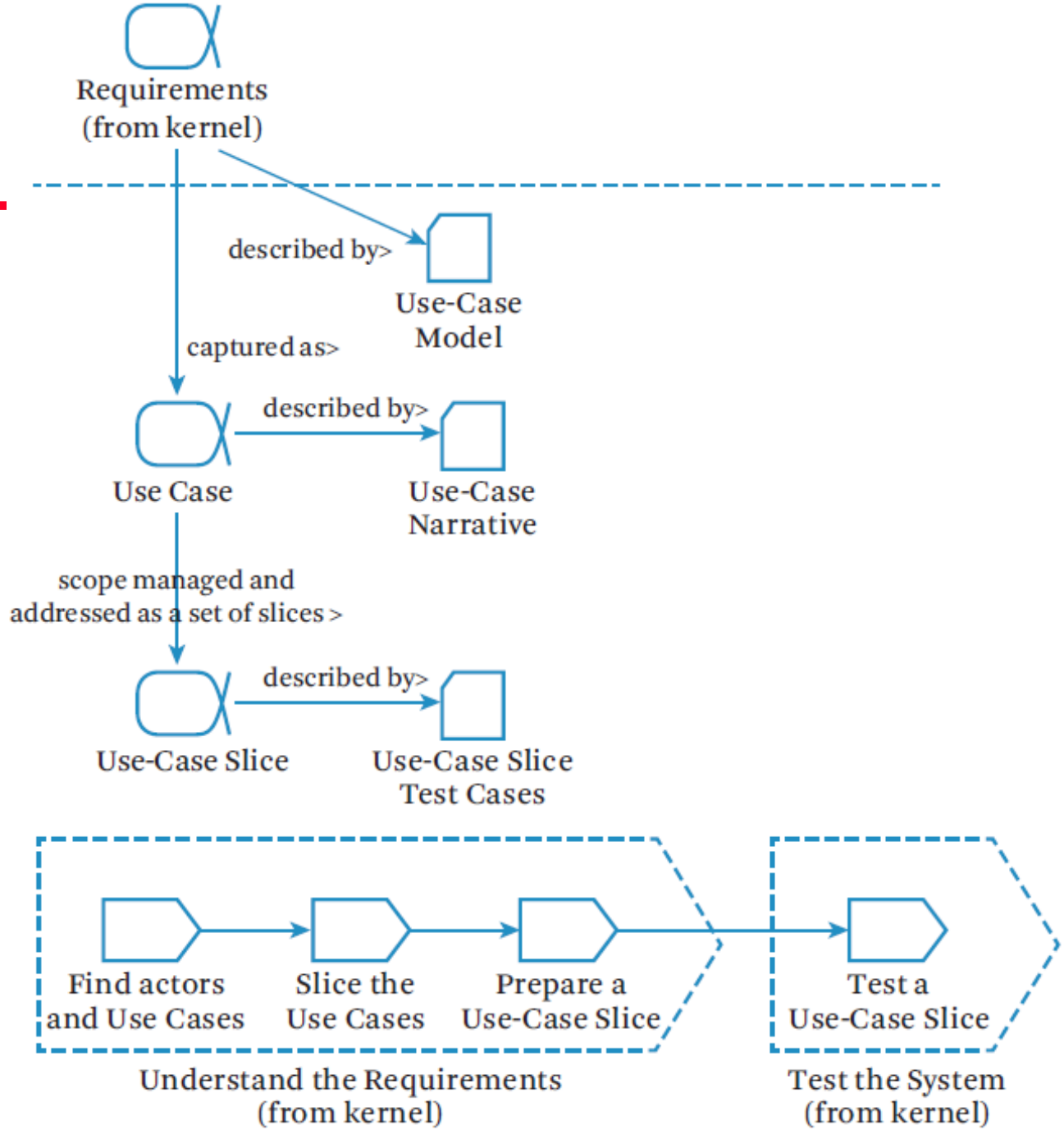
Endeavor activity spaces, partially covered by Scrum Lite.



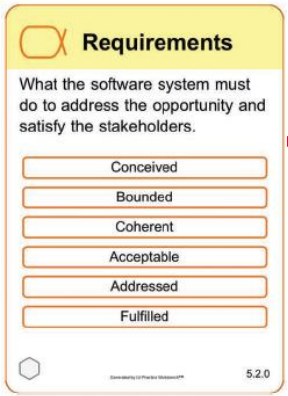
Solution activity spaces, not addressed by Scrum Lite.

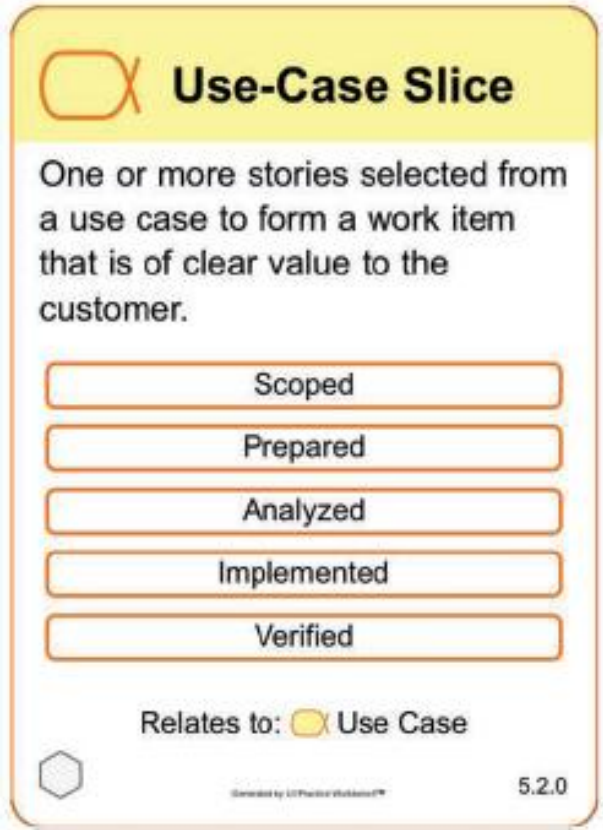
Use Case Lite practice

Scrum Lite practice expressed in the Essence language.

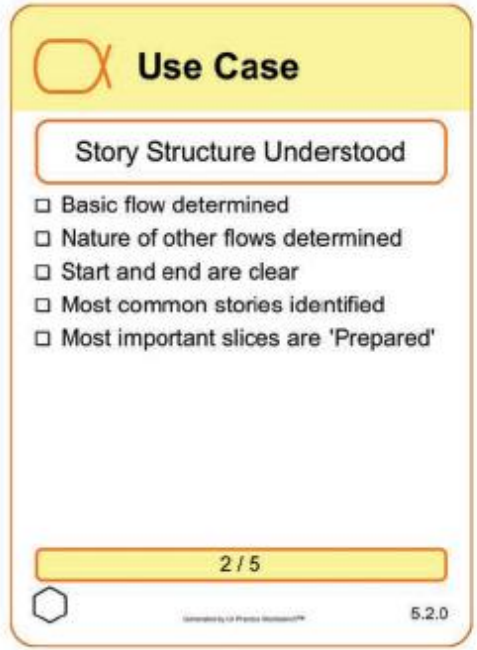


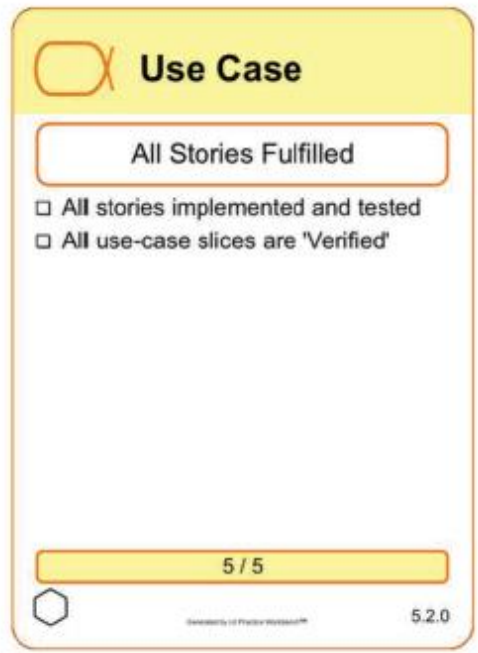
Use Case Lite specific-alphas



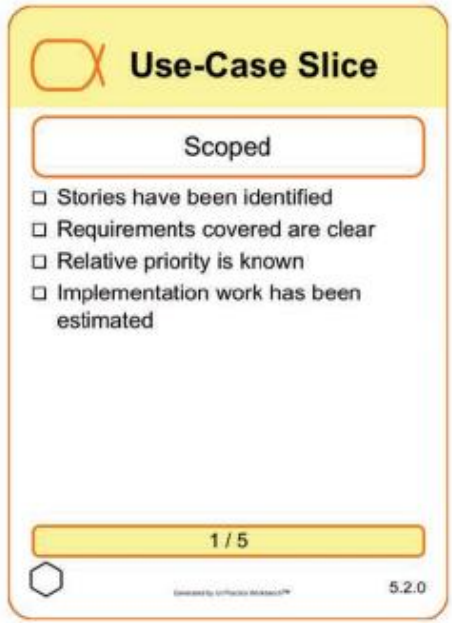


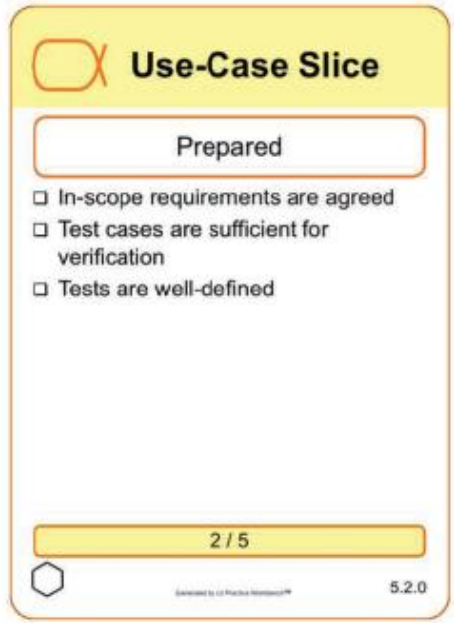
Use Case alpha states

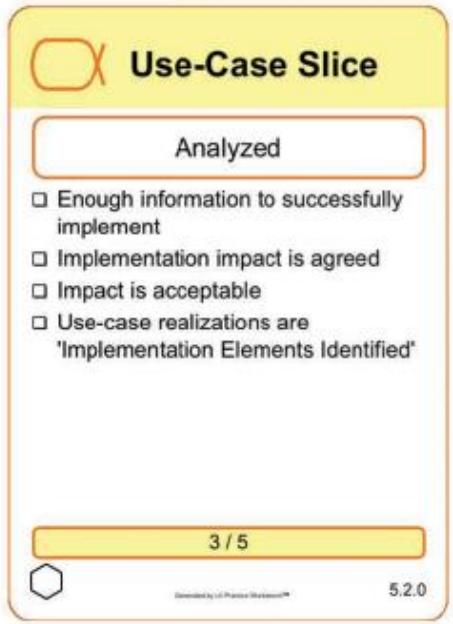


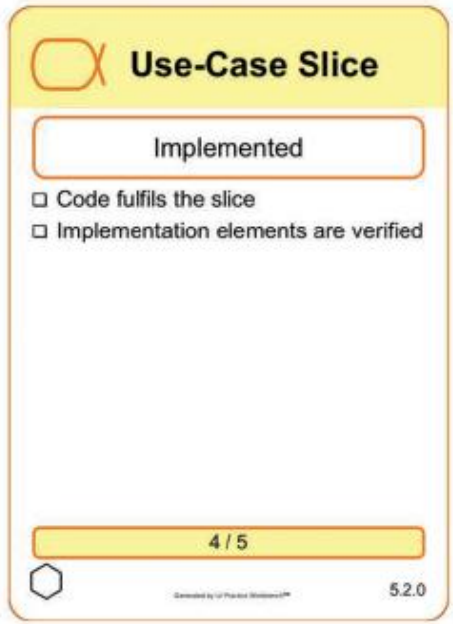


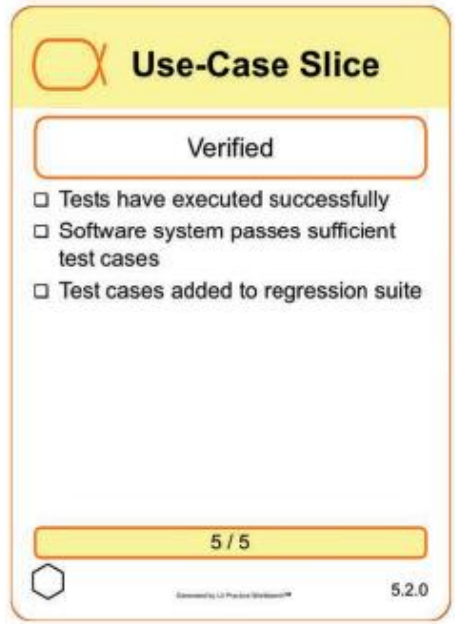
Use Case Slice alpha states











Use Case Lite – work products

Use-Case Model

A model that captures and visualizes all of the useful ways to use a system.

- Value Established
- System Boundary Established
- Structured

Describes: Requirements

5.2.0

Use-Case Narrative

Tells the story of how the system and its actors work together to achieve a particular goal.

- Briefly Described
- Bulleted Outline
- Essential Outline
- Fully Described

Describes: Use Case and Use-Case Slice

5.2.0

Use-Case Slice Test Case

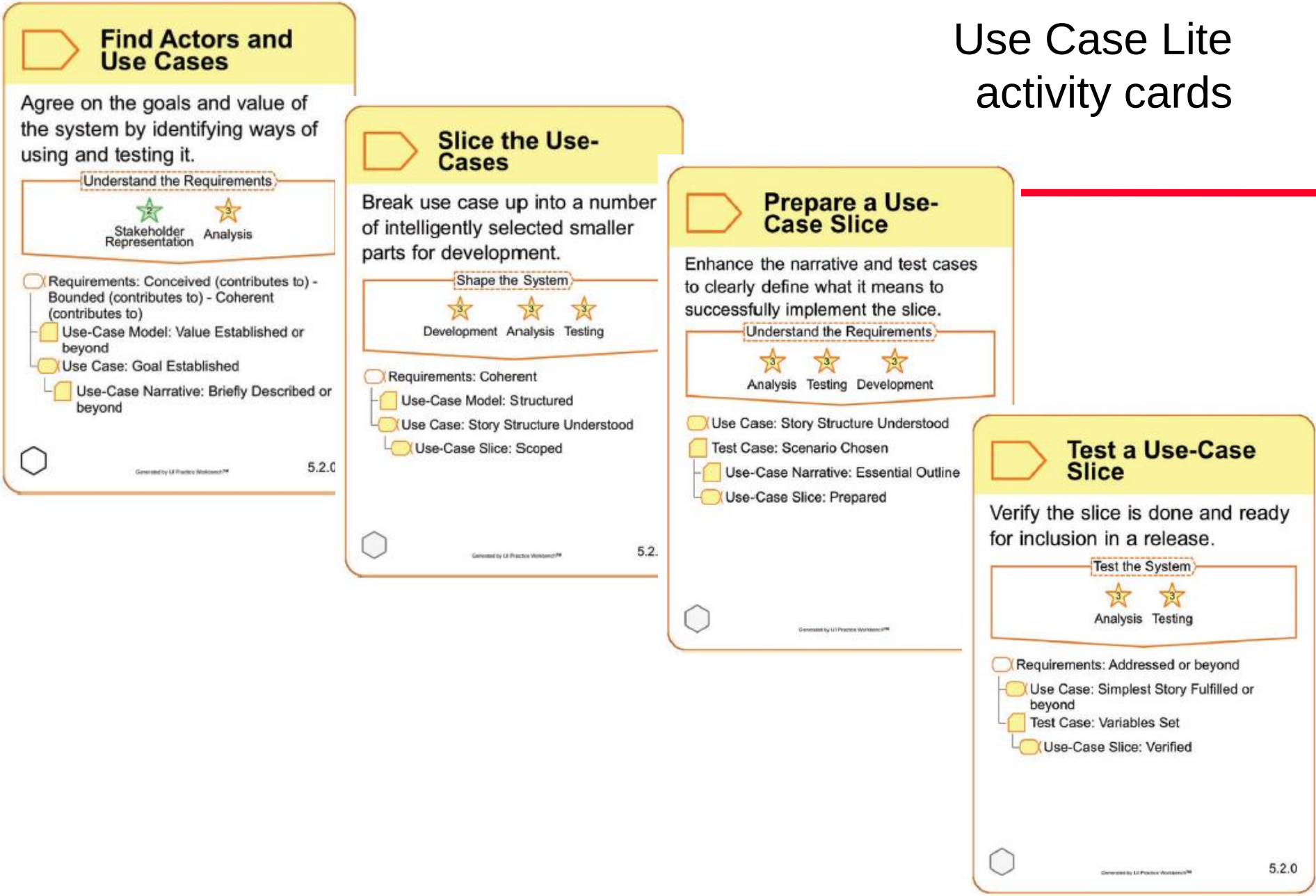
Defines test inputs and expected results to help evaluate whether a Use-Case Slice works correctly.

- Test Scenarios Chosen
- Variables Identified
- Test Variables Set
- Scripted or Automated

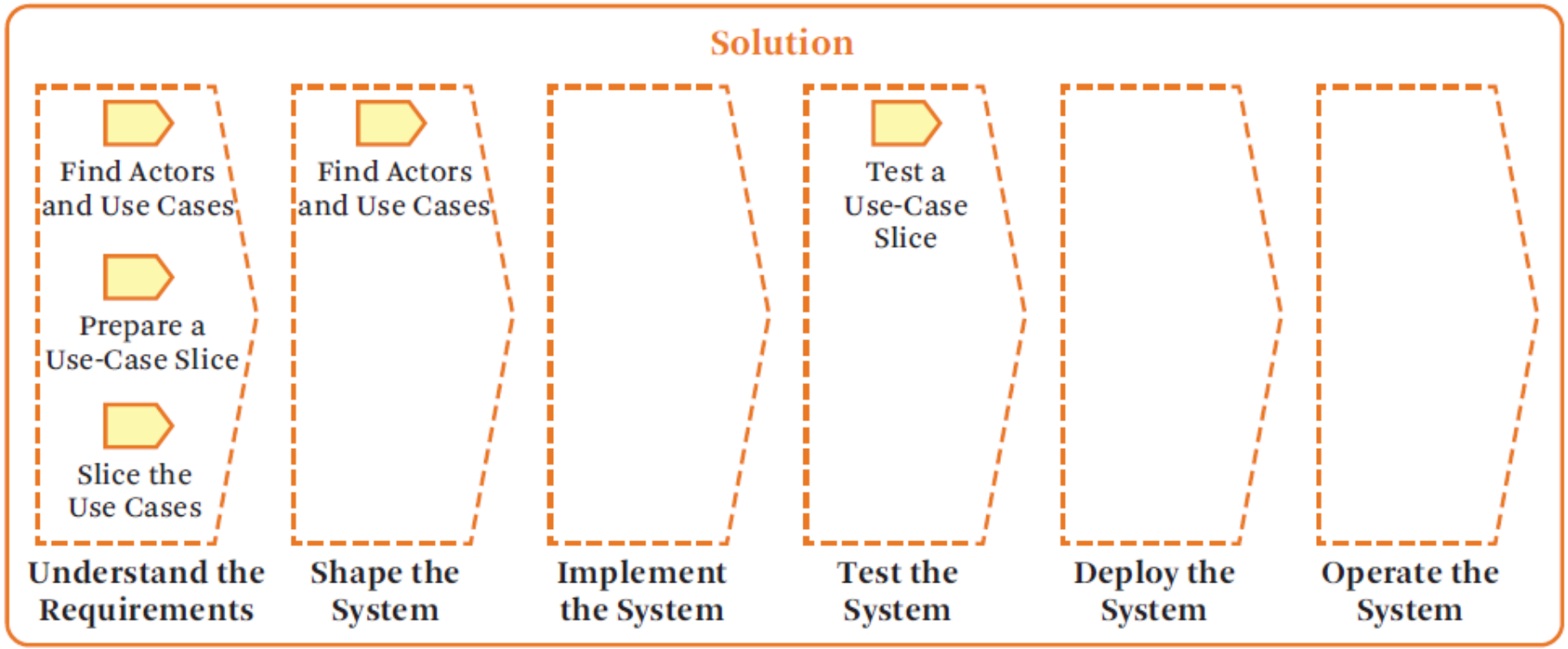
Describes: Use-Case Slice

5.2.0

Use Case Lite activity cards



Activity spaces addressed by Use Case Lite



Solution activity spaces and Use Case Lite.

Conclusion on practices

As a software professional, you will come in contact with many other practices.

We believe that in due time, popular practices will be essentialized.

As a student or professional who has a good understanding of Essence, you will learn these practices quickly.

Bibliography

For licence project:

https://www.ivarjacobson.com/publications/brochure/alpha-state-card-games

A case study:

https://dx.doi.org/10.1016/j.scico.2014.11.009

General

https://practicelibrary.ivarjacobson.com/start

http://software-engineering-essentialized.com/web/guest

https://puzzler.sim4seed.org/

http://www.software-engineering-essentialized.com/practices-with-deck-of-cards