
Software Engineering – Lecture 12

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

Topics covered

Software Engineering methods and SEMAT solution

Essence ideas and key concepts

The language of Essence

The kernel of Essence

Software Engineering Methods

Paradigm shifts regarding software development methods:

- structured methods
- component methods
- agile methods

Problems :

- new terminology, with little relation to the old ones
- transition extremely costly to the software industry in the form of training, coaching, and change of tooling

With every major technical innovation (ex. cloud computing) requiring a new set of practices, the method authors also “reinvent the wheel.”

Within every software engineering trend there are many competing methods.

- as early as 1990 there were about 30 competing object-oriented methods
- about 10 competing methods on scaling agile to large organizations (ex. Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD), Large Scale Scrum (LeSS), and Scaled Professional Scrum (SPS).) include
 - basic widely used practices (ex. Scrum), user stories / use cases, and continuous integration
 - some often useful practices that are specific for each one
 - no collaboration between method authors

Software Engineering Methods and SEMAT solution

- Methods are monolithic, not modular \Rightarrow difficult to mix and match practices from different methods.
- Once a method have adopted, you get the feeling you are in a *method prison* controlled by the guru of that method.

Proposed solution

- SEMAT (Software Engineering Method and Theory) initiative (started in 2009)

Initial observations :

- It can be guesstimated that there are over 100,000 different methods to develop software, since basically every team has developed their own way of working even if they didn't describe it explicitly.
- The number of methods is growing much faster than the number of reusable practices.

Objective of the solution :

- Provide means so that every team or organization to be able to set up its own method.

SEMAT call for action in 2009

The problem identified in this *call for action* :

Software engineering is gravely hampered today by immature practices.

Specific problems include:

- The prevalence of fads more typical of fashion industry than of an engineering discipline.
- The lack of a sound, widely accepted theoretical basis.
- The huge number of methods and method variants, with differences little understood and artificially magnified.
- The lack of credible experimental evaluation and validation.
- The split between industry practice and academic research.

SEMAT call for action in 2009

The solution proposed:

We support a process to re-found software engineering based on a solid theory, proven principles, and best practices that:

- include a kernel of widely agreed elements, extensible for specific uses
- address both technology and people issues
 - are supported by industry, academia, researchers and users
 - support extension in the face of changing requirements and technology.

The result:

Underlying language and kernel of software engineering was accepted in June 2014 as a standard by the OMG and it was given the name Essence.

Topics covered

Software Engineering methods and SEMAT solution

Essence ideas and key concepts

The language of Essence

The kernel of Essence

Essence

Essence key ideas:

Methods are composition of practices.

There is a common ground (kernel) shared among all methods and practices.

Focus on the essentials is needed when providing guidelines for a method or practice.

Providing software engineering experience is possible when teaching and learning methods and practices.

Realization of the ideas:

Concept of composition of practices.

Common ground : language and kernel of essential elements.

Practices and methods, build on top of Essence, to form method architecture.

Cards, as means for a tangible developer experience.

Methods and practices

Methods are composition of practices

Method = guide for the software development team during the software development process.

Practice = guides a team how to carry out one particular thing in their work (ex. requirements management, design, implement, test, organize the team, etc.)

Method relies on a composition of practices.

Practices are separate but not independent. Sometimes they overlap or are in conflict.

Example of practice content:

- guidelines for developer activities
- guidelines for work products

Two practices may contribute to the same work product => need to specify how the contributions must be combined in a meaningful and constructive way.

Practices may be compositions of smaller practices.

Ex. Scrum = composition of Daily Standup, Backlog-Driven Development, Retrospective.

Essence – common ground

Common ground – describe, teach, learn, use, modify, compare practices.

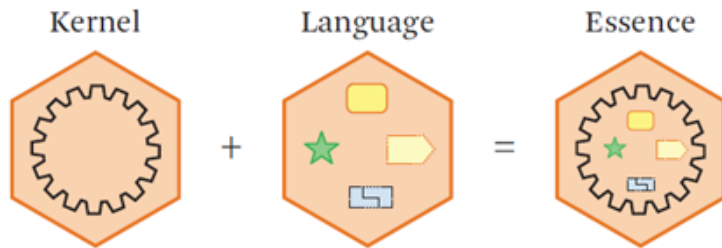


Fig. 3.2 Essence and its parts
The Essentials of Modern Software Engineering

The practices are *essentialized*, meaning they are described using Essence—the Essence kernel and the Essence language.



The methods, composed of practices, are also *essentialized*.

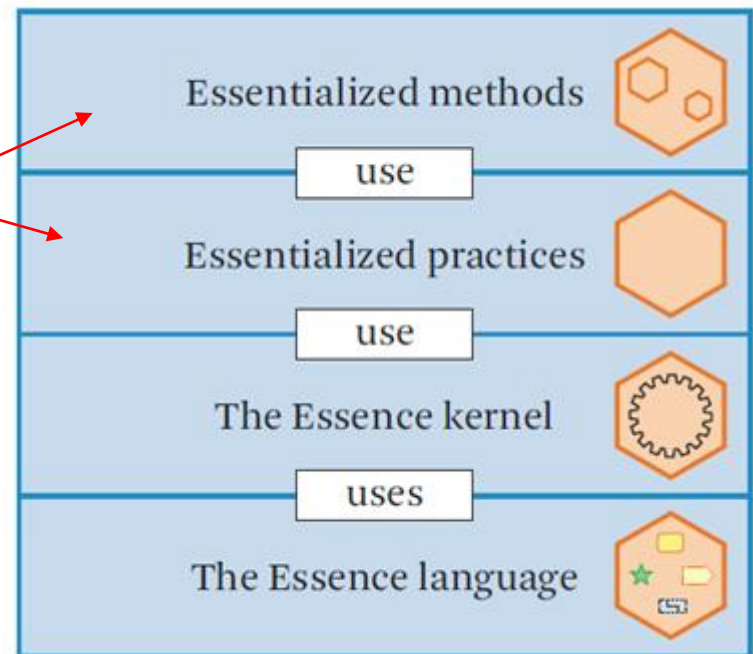


Fig. 3.3 Essence method architecture
The Essentials of Modern Software Engineering

Essence – common ground

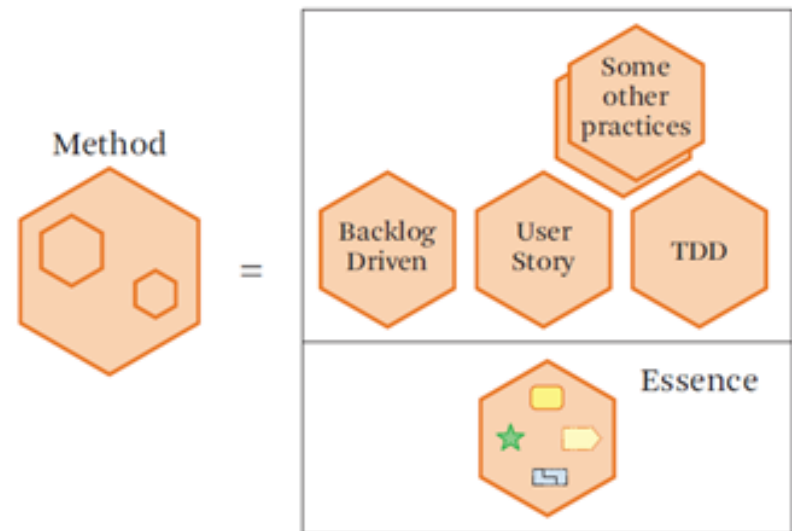
Essentialized Methods are composition of **Essentialized practices** That are described using Kernel Elements Using Essence Language

Libraries of practices coming from many different methods can be created.

Teams can mix and match practices from many methods to obtain a method they want.

If an idea for a new practice appears, that practice can be essentialized and added to a practice library for others to select.

When creating a new method, it is no need to “reinvent the wheel”.



*Fig. 3.4 A method is a composition of practices on top of the kernel
The Essentials of Modern Software Engineering*

Essence – common ground

Ways of learning methods:

- own working experience
- reading books
- reading navigable web sites with method presentation

Essence way

- learn Essence, the common ground
- use Essence cards



Fig. 3.5 Cards make the kernel and practices tangible.
The Essentials of Modern Software Engineering

Facets of all software development endeavors

There are *customers* with needs to be met.

- Someone has a problem or *opportunity* to address.
- There are *stakeholders* who use and/or benefit from the solution produced, and some of these will fund the endeavor.

There is a *solution* to be delivered.

- There are certain *requirements* to be met.
- A *software system* of one form or another will be developed.

There is an *endeavor* to be undertaken.

- The *work* must be initiated.
- An empowered *team* of competent people must be formed, with an appropriate *way of working*.

Essence areas of concerns

Customer – contains everything to do with the actual use and exploitation of the software system to be produced.

Solution - contains everything related to the specification and development of the software system.

Endeavor - contains everything related to the development team and the way that they approach their work.

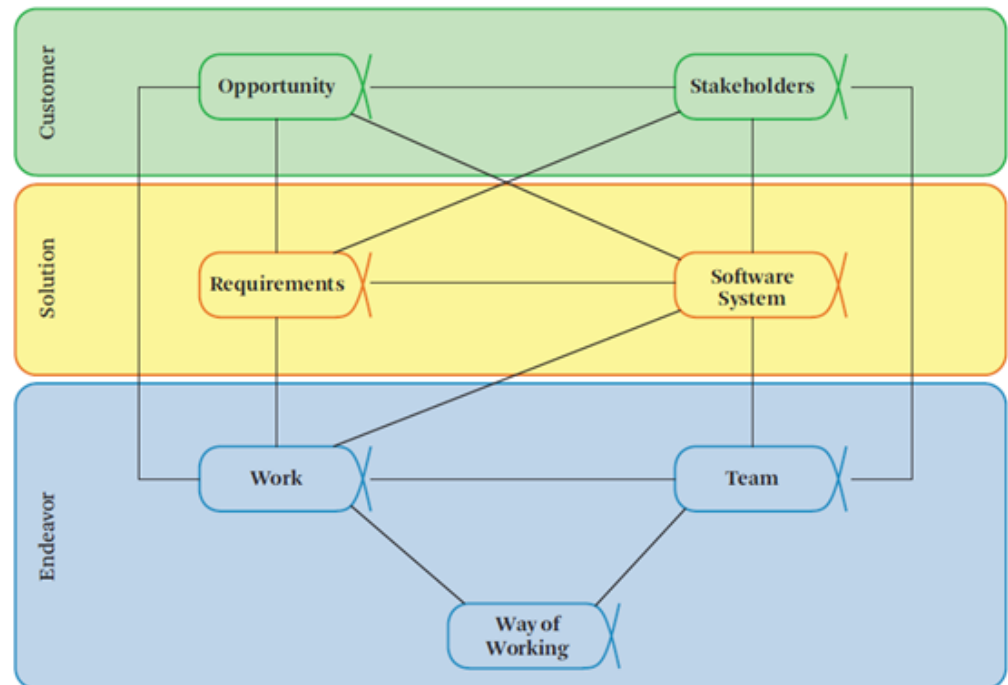


Fig. 4.1 The things involved in all development endeavors
The Essentials of Modern Software Engineering

Topics covered

Software Engineering methods and SEMAT solution

Essence ideas and key concepts

The language of Essence

The kernel of Essence

The language of software engineering

The language constructs :

- alpha,
- alpha state,
- work product,
- activity,
- activity space,
- competency,
- pattern

The language constructs are materialized as cards.

The cards are a practical way to use the various language elements of Essence.

The language of software engineering

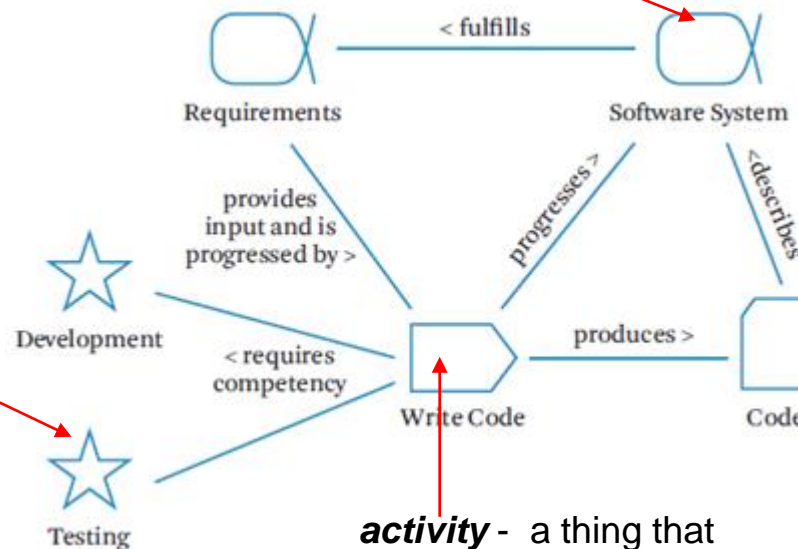
A simple practice example.

- Purpose : to produce higher quality code.
- Description : 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.

Fig. 5.1 Simple programming practice
The Essentials of Modern Software Engineering

alpha – Essential element of the development endeavor that is relevant to an assessment of the progress and health of the endeavor.

competency – an ability, capability, attainment, knowledge, or skill necessary to do a certain kind of work..



work product – tangible thing that practitioners produce when conducting software engineering activities.

activity - a thing that practitioners do.

The language of software engineering

alpha – essential element of the development endeavor that is relevant to an *assessment of the progress and health* of the endeavor.

- *intangible, conducting* element
- have *states* to evaluate progress and health of the endeavor
- understood and described by the *work products associated* with it.

work product – tangible thing that practitioners produce when conducting software engineering activities.

Examples : requirements specifications, design models, code.

Example: *Requirements alpha*

- always exist in a software development endeavor.
- sometimes requirements may just exist in the heads of people.
- may be evidenced by associating work products like requirements items, test cases, user manuals.

The language of software engineering

Alpha
Work Product
Activity
Competency
Activity Space
Pattern

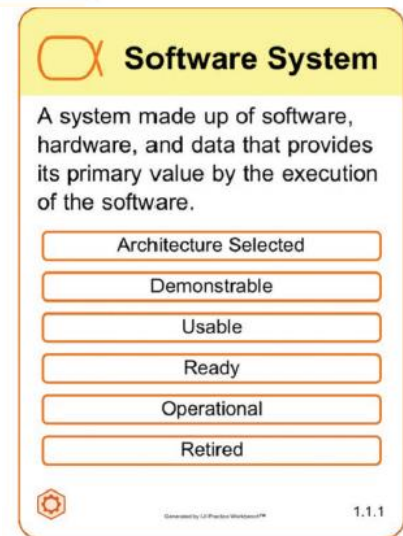
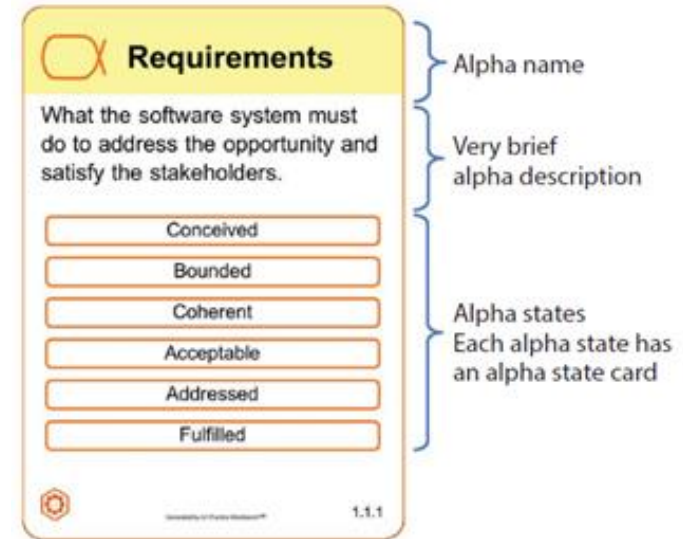
alpha – essential element of the development endeavor that is relevant to an *assessment of the progress and health* of the endeavor.

Bound the work to be accomplished in progressing toward the provisioning of the software system.

The most important things that must be attended to, and progressed, in order to be successful in a development endeavor.

alpha states

- describe progression through a lifecycle
- are specific to each alpha



*Figures 5.2 and 5.4 Requirements and Software System alpha cards
The Essentials of Modern Software Engineering*

Requirements alpha state cards

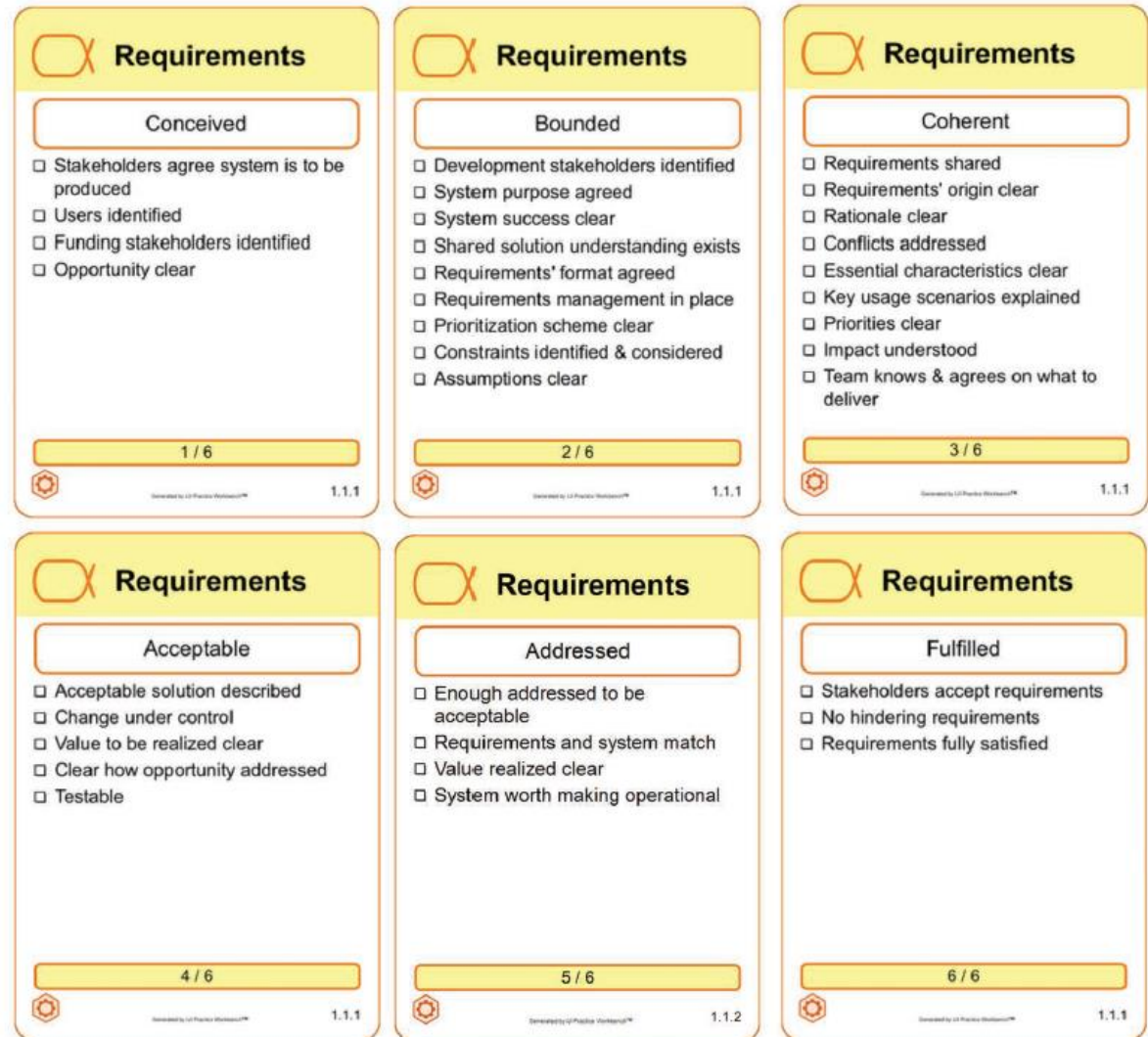


Fig. 5.3 Requirements alpha state cards
The Essentials of Modern Software Engineering

States for Requirements alpha

Alpha
Work Product
Activity
Competency
Activity Space
Pattern

State	Description
-------	-------------

States for Software System alpha

Alpha
Work Product
Activity
Competency
Activity Space
Pattern

State	Description
-------	-------------

Work Products

Work product = tangible thing that may provide evidence to verify the achievement of an *alpha state*.

Examples :

- Requirements (alpha) - Requirements Specification (work product)
- Software System (alpha) – Code (work product)

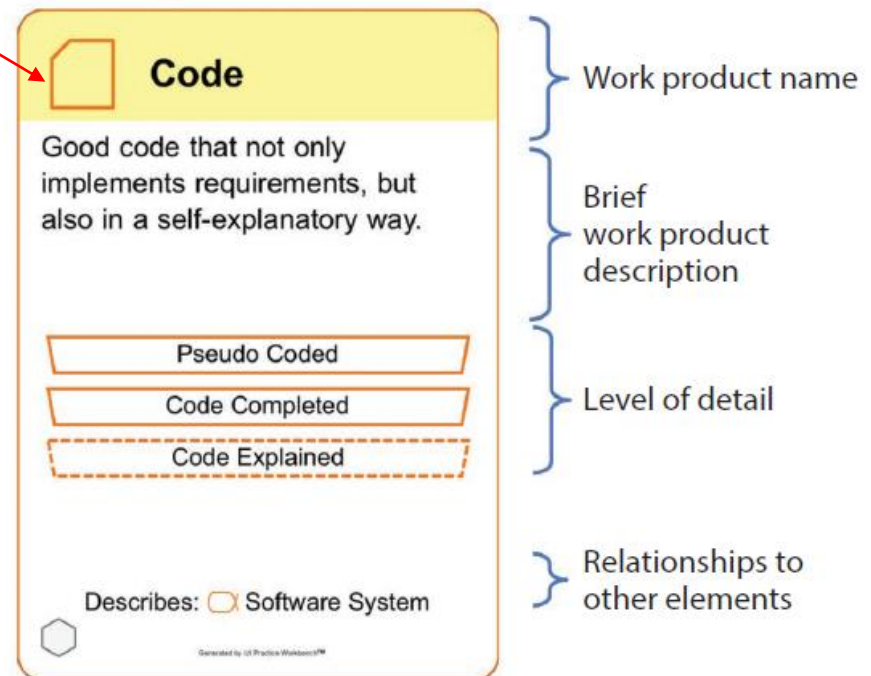


Fig. 5.5 The Code work product card
The Essentials of Modern Software Engineering

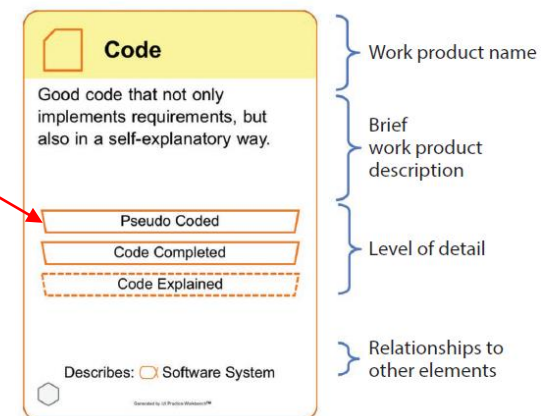
Work Products

Work product = tangible thing that may provide evidence to verify the achievement of an *alpha state*.

- Code is an instance of Work Product and is specific to a programming practice.
- It is not part of Essence kernel.
- Essence specifies only the type, i.e. what work products are, how to represent them, and what can be done with them.
- Work product can have different levels of detail in different teams.

Definition of detail levels depends on factors like:

- past history of team members working together
 - customer requirements,
 - regulatory requirements
- (e.g., regulation for software validation of medical devices),
- organizational policies.



Competency

Competency = one of the abilities needed when applying a practice.

Example :
Development competency

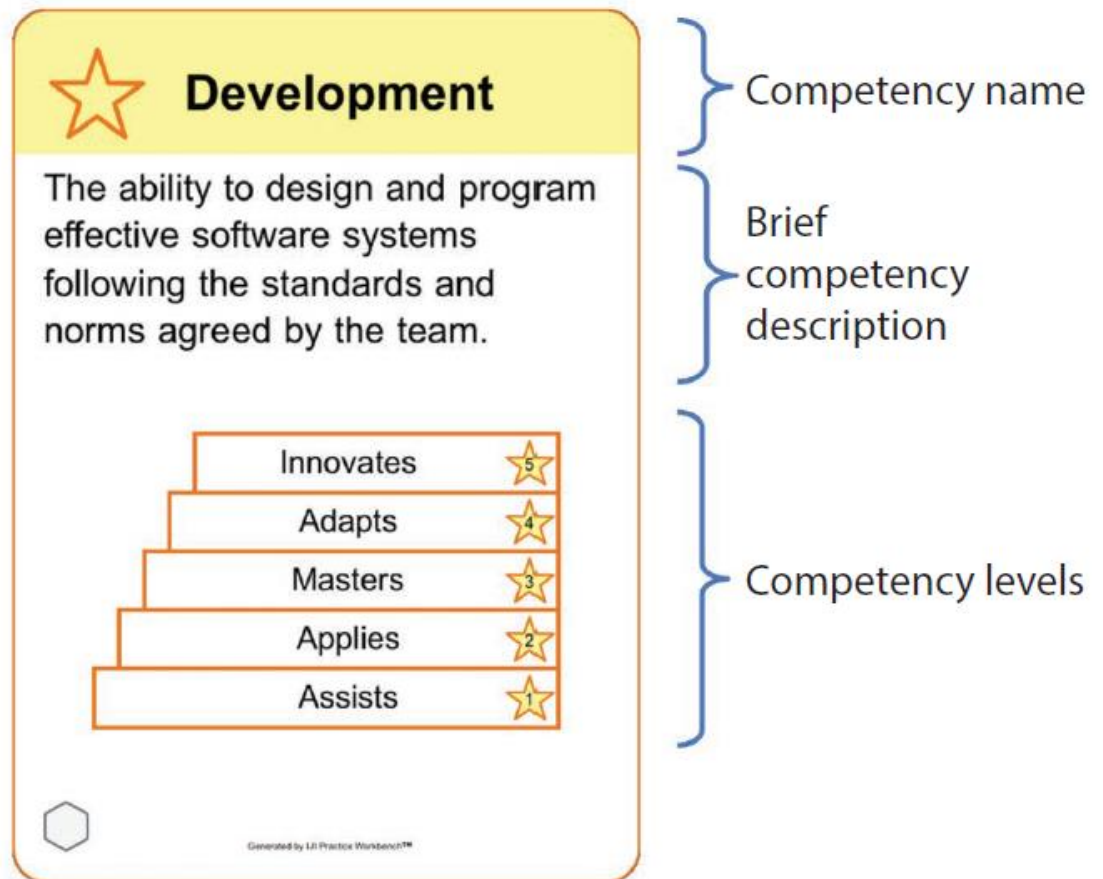


Fig. 5.6 The Development competency card
The Essentials of Modern Software Engineering

Levels of achievement for Development competency

Alpha
Work Product
Activity
Competency
Activity Space
Pattern

Competency level	Description

Activity

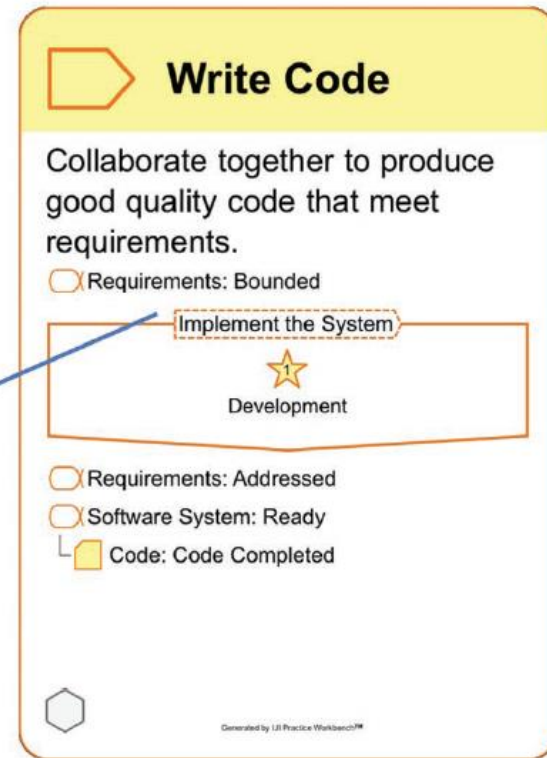
Activity = a thing that practitioners do.

- Instance of the *Activity Type*
- Is defined at the level of each practice.
- A practice includes more activities which are specific to that practice,

Examples:

- holding a meeting,
- analyzing a requirement,
- writing code, testing,
- peer review.







Activity space which
this activity belongs to



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

Fig. 5.7 The Write Code activity card
The Essentials of Modern Software Engineering

Essence language. Elements

Element Type	Syntax	Description
Alpha		An essential element of the development endeavor that is relevant to an assessment of the progress and health of the endeavor.
Work Product		A tangible thing that practitioners produce when conducting software engineering activities.
Activity		A thing that practitioners do.
Competency		An ability, capability, attainment, knowledge, or skill necessary to do a certain kind of work.
Activity Space		A placeholder for something to do in the development endeavor. A placeholder may consist of zero to many activities.
Pattern		An arrangement of other elements represented in the language.

Essence language Relationships

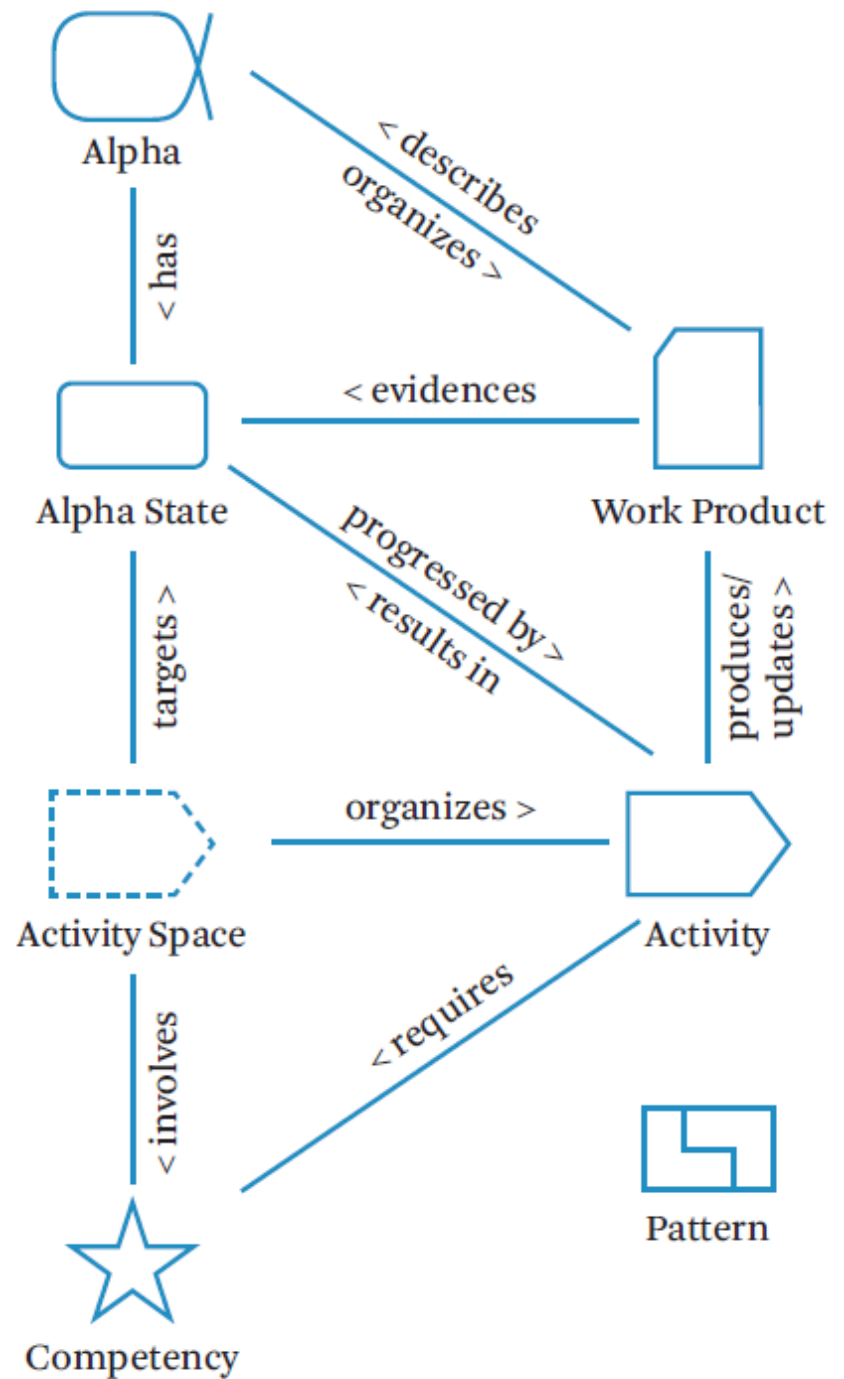


Fig. 5.8 Elements of Essence language and their relationships
The Essentials of Modern Software Engineering

Essence language and essentializing practices

Steps to essentialize a practice :

- **Identify the elements.**
 - First, build a list of elements that make up a practice.
 - The output is essentially a diagram.
- **Draft the relationships between the elements and the outline of each element.**
 - At this point, the cards are created.
- **Provide 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.

Topics covered

Software Engineering methods and SEMAT solution

Essence ideas and key concepts

The language of Essence

The kernel of Essence

Essence kernel

Structured approach in organizing the elements of software engineering.

- 3 discrete areas of concern
 - customer
 - solution
 - endeavor
- 4 fundamental types of kernel elements
 - alpha
 - activity space
 - competency
 - pattern

Essence kernel

Areas of concern

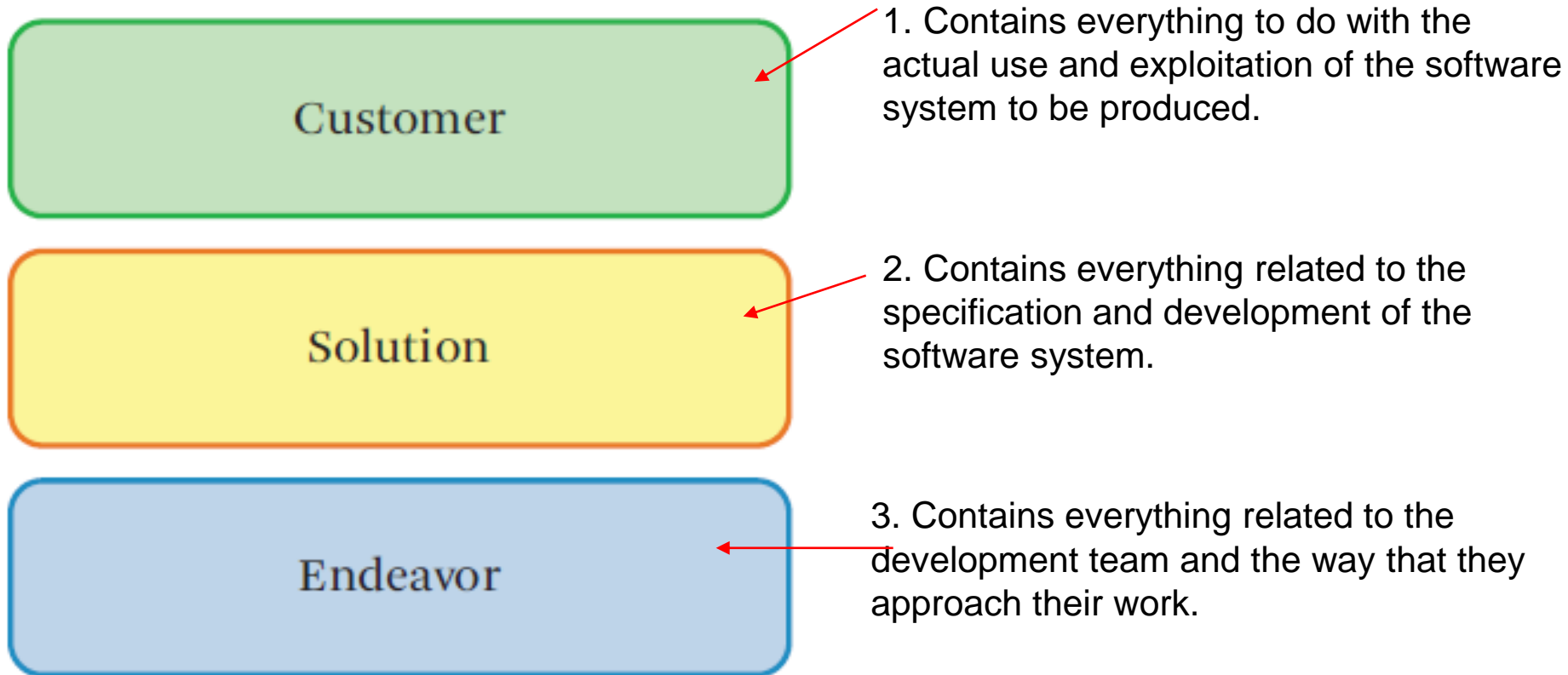


Fig. 6.1 The three areas of concern
The Essentials of Modern Software Engineering

Essence kernel Elements

1. The essential things to work with: the *alphas*
2. The essential things to do: the *activity spaces*
3. The essential capabilities needed: the *competencies*
4. The essential arrangements of elements: the *patterns*

Essence kernel

Alphas – things to work with

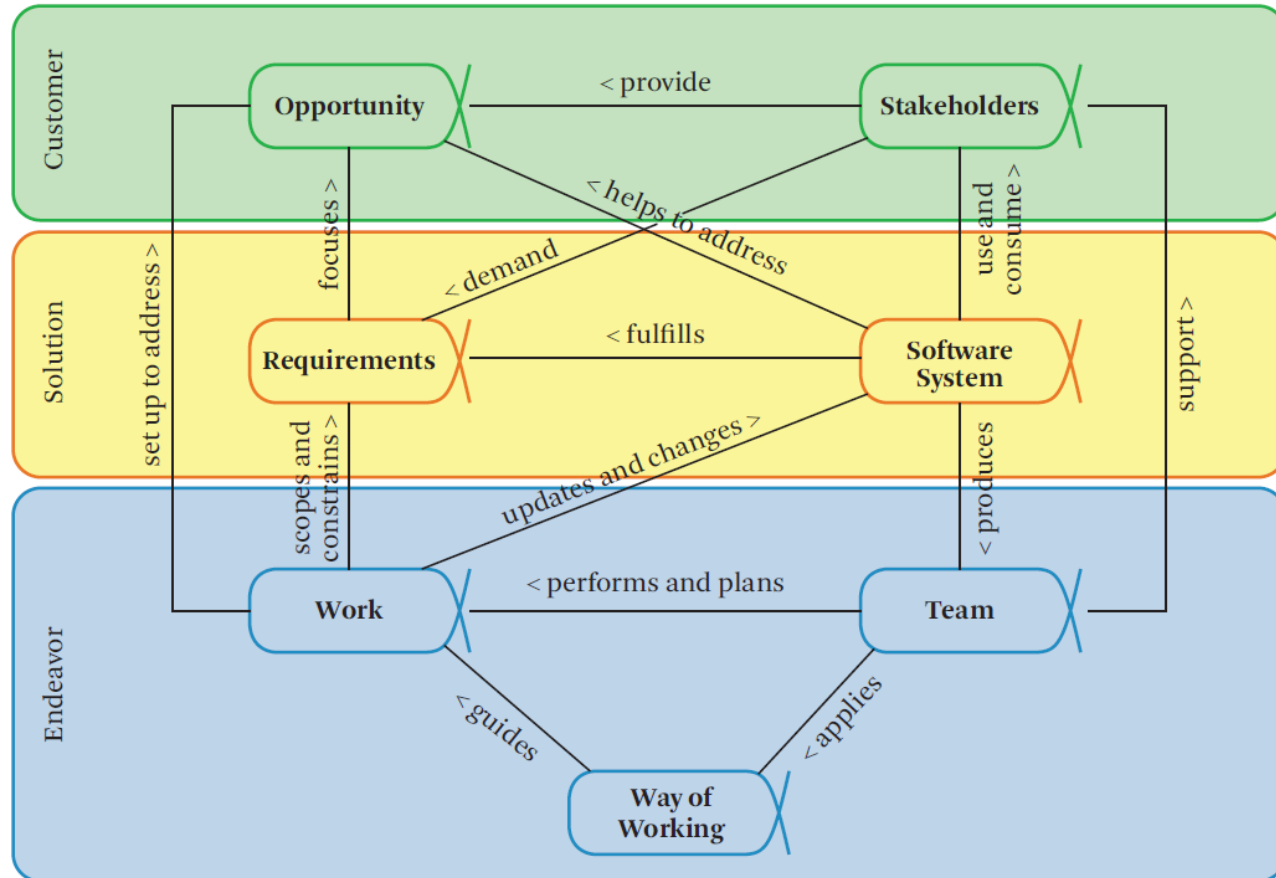
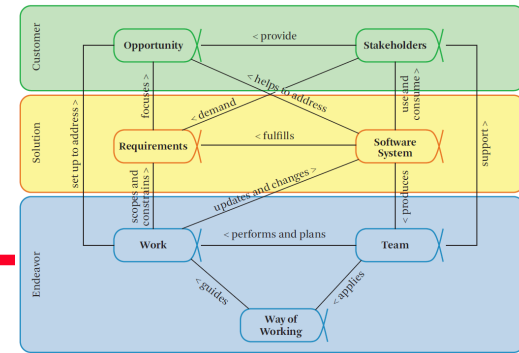


Fig.6.2 The Essence alphas and their relationships
The Essentials of Modern Software Engineering

Essence kernel Customer alphas

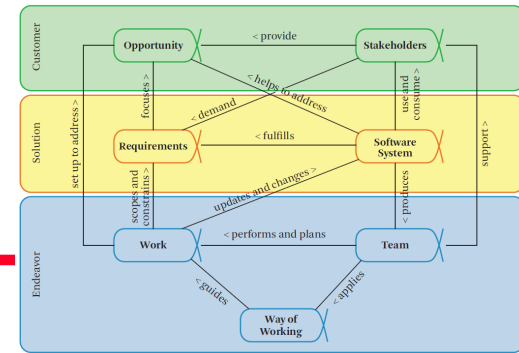


In the Customer area of concern, the team needs to understand the stakeholders' needs and the opportunity to be addressed.

Opportunity. The set of circumstances that makes it appropriate to develop or change a software system. The opportunity articulates the reason for the creation of the new, or changed, software system. It represents the team's shared understanding of the stakeholders' needs and helps shape the requirements for the new software system by providing justification for its development.

Stakeholders. The people, groups, or organizations that affect or are affected by a software system. The stakeholders provide the opportunity and are the source of the requirements and funding for the software system. The team members are also stakeholders. As much stakeholder involvement as possible throughout a software engineering endeavor is important to support the team and ensure that an acceptable software system is produced.

Essence kernel Solution alphas

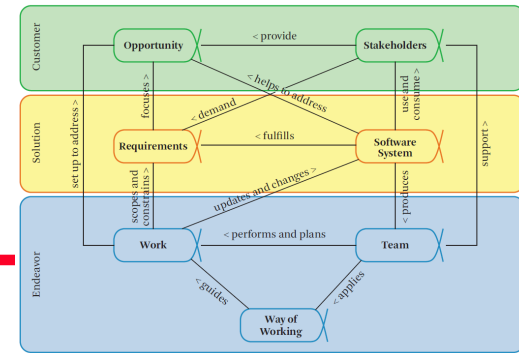


In the Solution area of concern, the team needs to establish a shared understanding of the requirements, and then implement, build, test, deploy, and support a software system that fulfills them.

Requirements. What the software system must do to address the opportunity and satisfy the stakeholders. It is important to discover what is needed from the software system, share this understanding among the stakeholders and the team members, and use it to drive the development and testing of the new system.

Software System. A system made up of software, hardware, and data that provides its primary value by the execution of the software. The primary product of any development endeavor, a software system can be part of a larger software, hardware, business, or social system solution.

Essence kernel Endeavor alphas



In the Endeavor area of concern, the team and its way of working have to be formed, and the work has to be done.

Team. A group of people actively engaged in the development, maintenance, delivery, or support of a specific software system. The team plans and performs the work needed to create, update, and/or change or retire the software system.

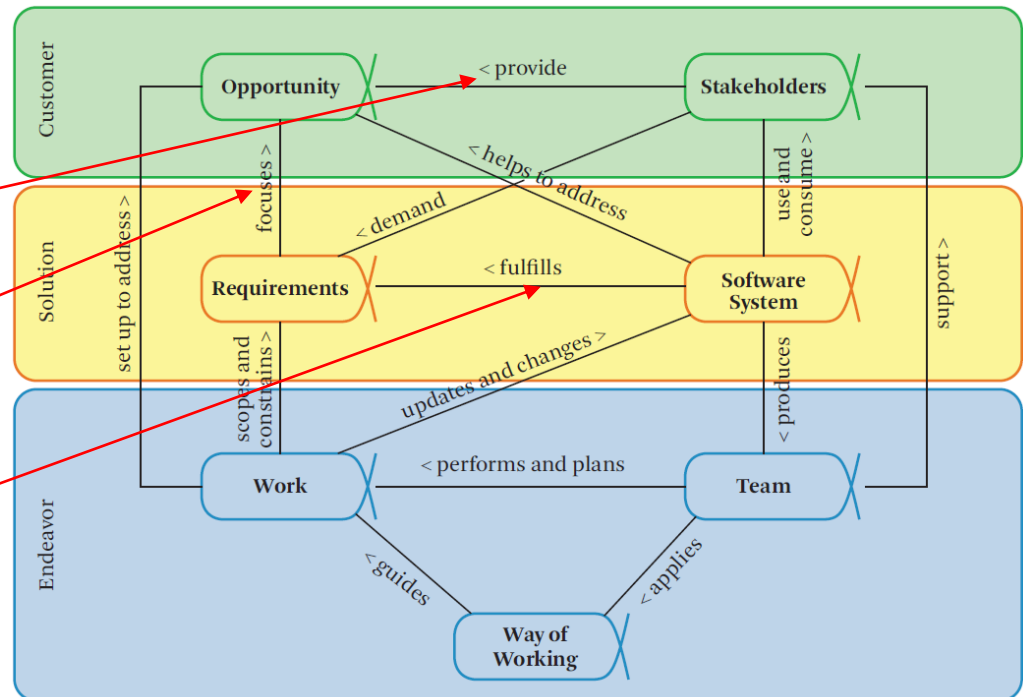
Work. Activity involving mental or physical effort done in order to achieve a result. In the context of software engineering, work is everything that the team does to meet the goals of producing a software system matching the requirements, and addressing the opportunity, that has been presented by the stakeholders. The work is guided by the practices that make up the team's way of working.

Way of Working. The tailored set of practices and tools used by a team to guide and support their work. The team evolves their way of working alongside their understanding of their mission and their working environment. As their work proceeds, they continuously reflect on their way of working and adapt it as necessary to their current context.

Essence kernel Alpha relationships

All alphas are related to one another and they complement each other by addressing their own aspects of the development endeavor.

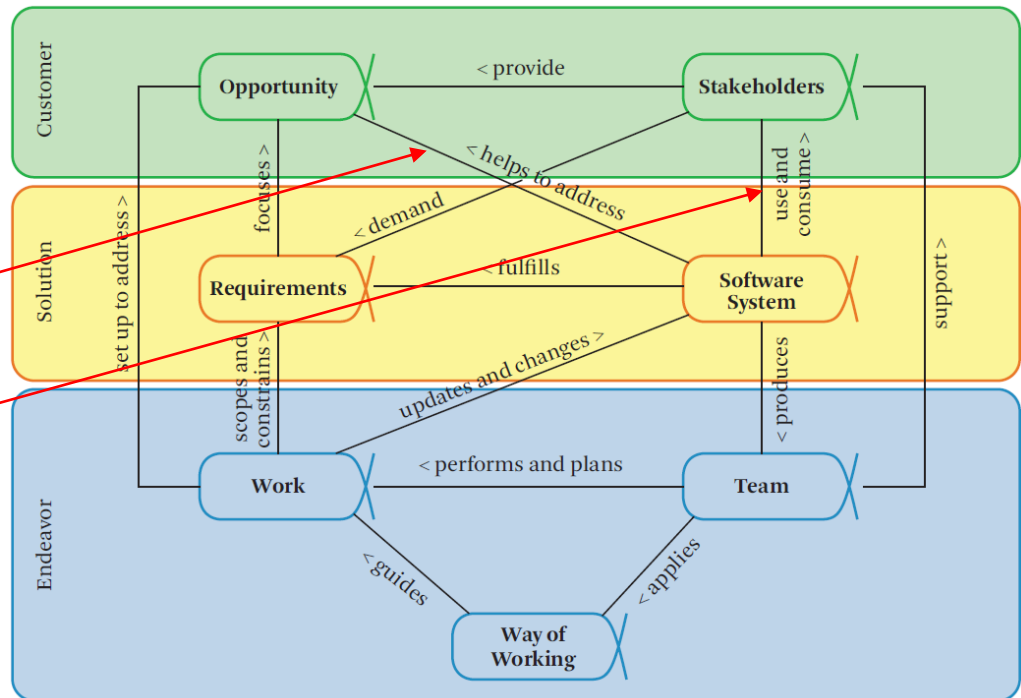
Stakeholders provide *Opportunity*, which then helps to identify *Requirements* and focuses on the most important ones. These *Requirements* are then fulfilled by implementing a *Software System*.



Essence kernel Alpha relationships

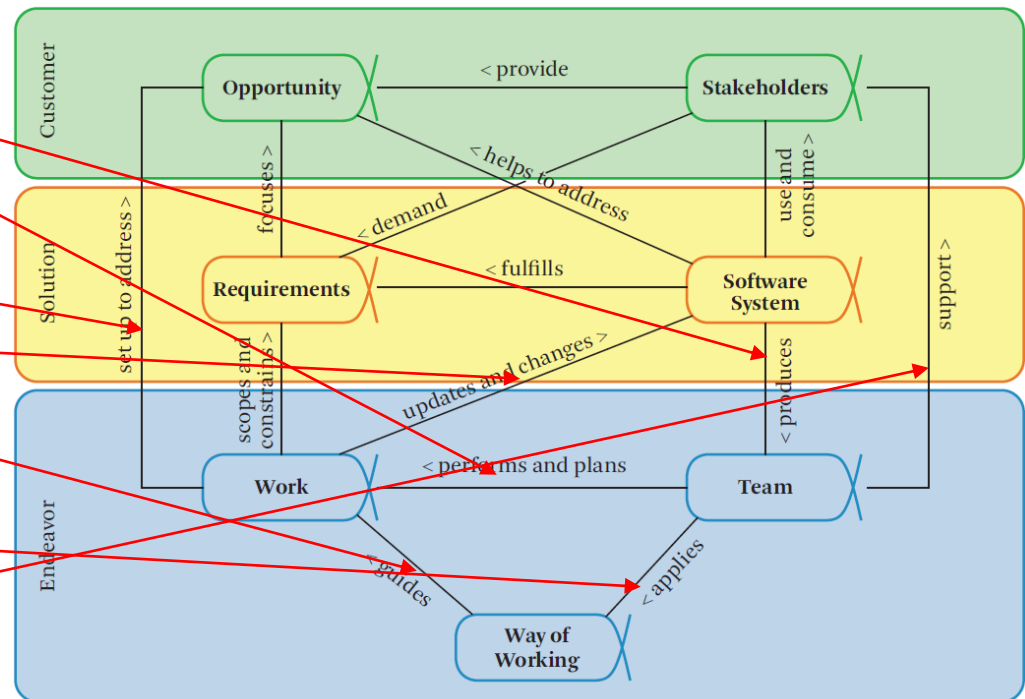
All alphas are related to one another and they complement each other by addressing their own aspects of the development endeavor.

The *Software System* implementation addresses the *Opportunity*, and it is used and consumed by *Stakeholders*.



Essence kernel Alpha relationships

The *Team* produces the *Software System* by doing *Work*. The *Work* is set up to address the *Opportunity* and it implies updating and changing the *Software System*. *Work* is guided by a *Way of Working* that is applied by the *Team* while performing its *Work*. The *Team* is continuously supported by *Stakeholders* who provide feedback about the *Software System* to the *Team*.



Essence kernel Alpha states cards

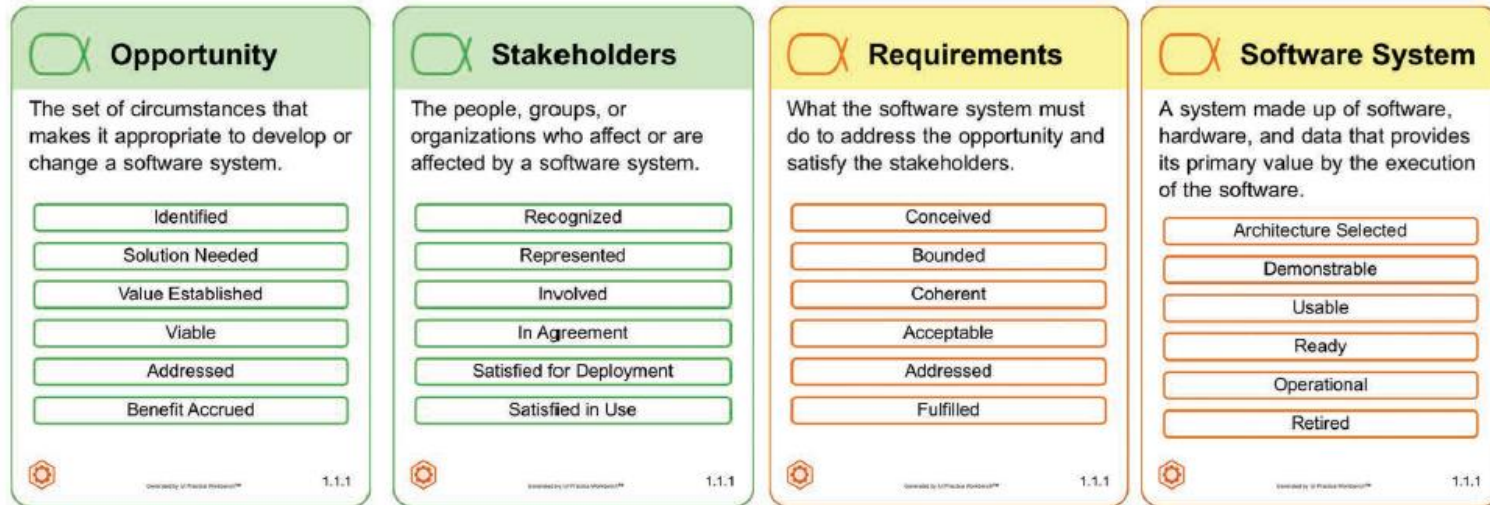


Figure 6.3 Essence kernel
alpha state cards
*The Essentials of Modern
Software Engineering*



Details at OMG and on the site
of the book.

Essence kernel

Activity spaces – things to do

Activity space = generic (i.e., method-independent) placeholders for specific activities that will be added later, on top of the kernel.

- packages used to group activities that are related to one another.
- guidance for achieving the states specified in the alphas

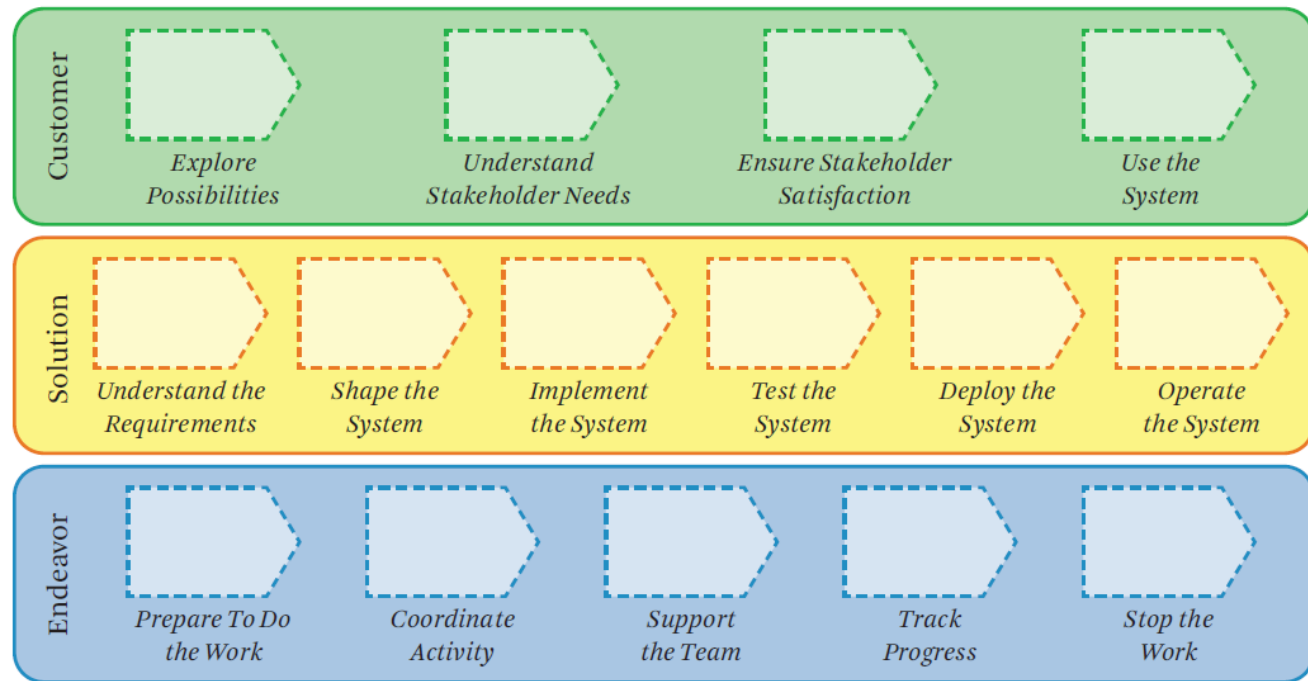
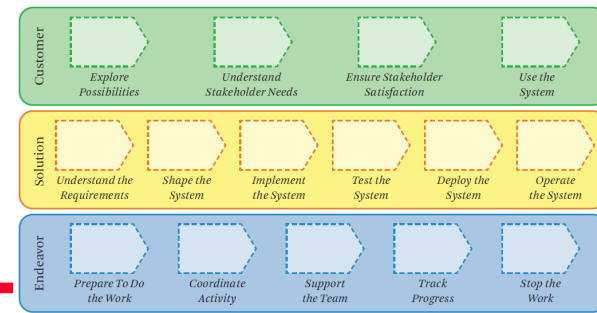


Figure 6.4 Essence activity spaces
The Essentials of Modern Software Engineering

Obs. The sequences indicate the order in which things are finished and not necessarily the order in which they are started. Overlaps may occur.

Essence kernel

Customer activity spaces



Activities necessary to understand the opportunity, and to support and involve the stakeholders.

Explore Possibilities. Explore the possibilities presented by the creation of a new or improved software system. This includes the analysis of the opportunity and the identification of the stakeholders.

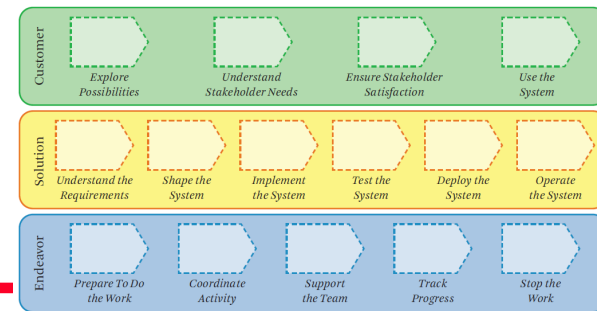
Understand Stakeholder Needs. Engage with the stakeholders to understand their needs and ensure that the right results are produced. This includes identifying and working with the stakeholder representatives to progress the opportunity.

Ensure Stakeholder Satisfaction. Share the results of the development work with the stakeholders to gain their acceptance of the system produced and verify that the opportunity has been addressed.

Use the System. Observe the use of the system in a live environment and how it benefits the stakeholders.

Essence kernel

Solution activity spaces



Activities necessary to develop an appropriate solution to exploit the opportunity and satisfy the stakeholders.

Understand the Requirements. Establish a shared understanding of what the system to be produced must do.

Shape the System. Form and structure, i.e., shape the system so that it is easy to develop, change, and maintain, and can cope with current and expected future demands. This includes the architecting and overall design of the system to be produced.

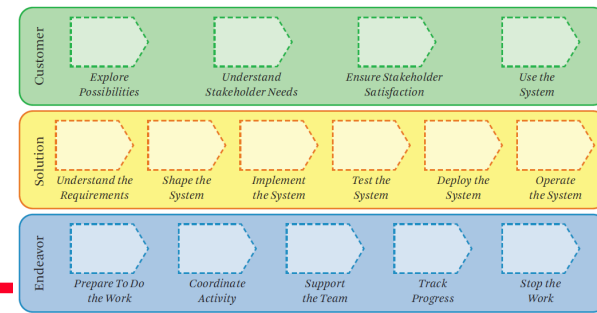
Implement the System. Build a system by implementing, testing, and integrating one or more system elements. This includes bug fixing and unit testing.

Test the System. Verify that the system produced meets the stakeholders' requirements.

Deploy the System. Take the tested system and make it available for use outside the development team.

Operate the System. Support the use of the software system in the live environment.

Essence kernel Endeavor activity spaces



Activities necessary to form a team and to progress the work in line with the agreed way of working.

Prepare to Do the Work. Set up the team and its working environment. Understand and commit to completing the work.

Coordinate Activity. Coordinate and direct the team's work. This includes all ongoing planning and re-planning of the work, and re-shaping of the team.

Support the Team. Help the team members to help themselves, collaborate, and improve their way of working.

Track Progress. Measure and assess the progress made by the team.

Stop the Work. Shut down the development endeavor and handover of the team's responsibilities.

Essence kernel

Activity spaces – things to do

Example : Implement the System

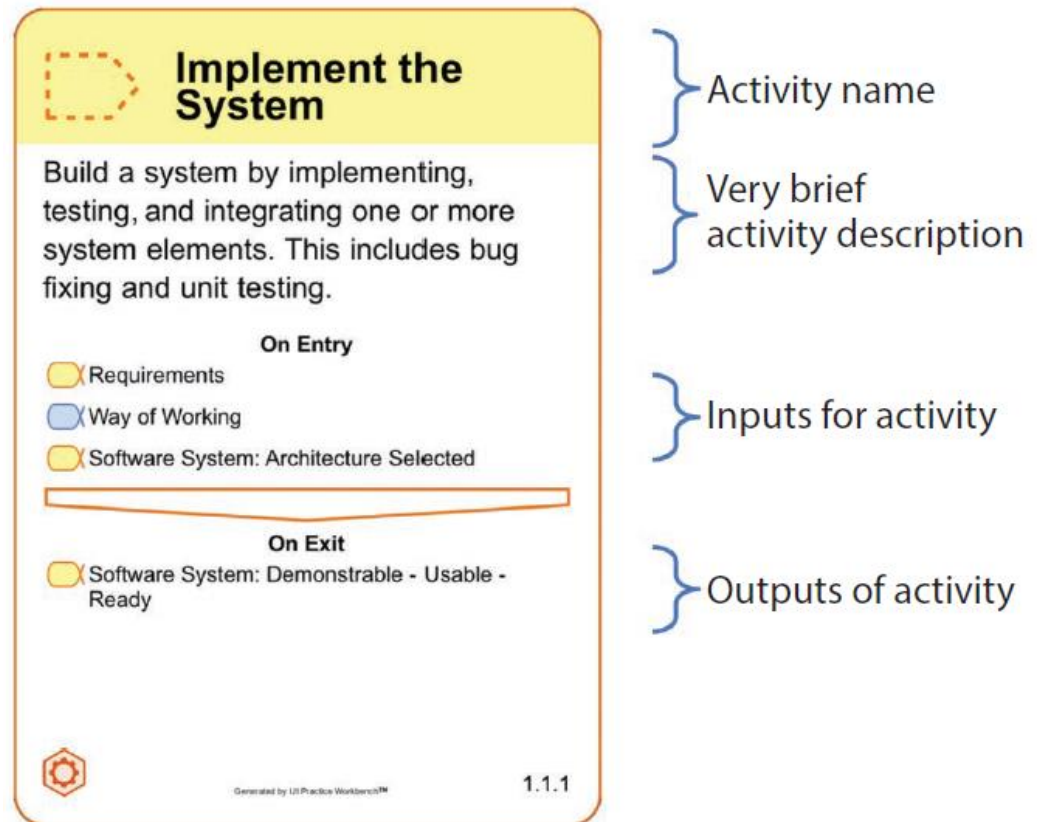


Figure 6.5 Implement the System activity space card
The Essentials of Modern Software Engineering

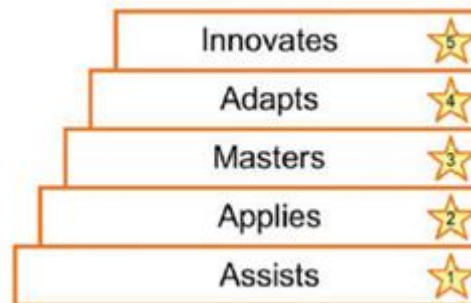
Essence kernel Competencies

Competencies - defined in the kernel as generic containers for specific skills.

Individual teams identify the specific skills needed for their particular software endeavor.

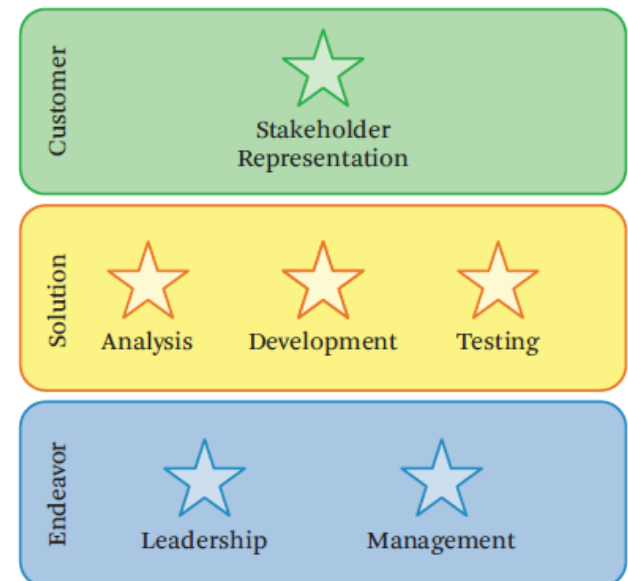
- Competency may be:
 - relevant to the specific tasks
 - other competencies to understand what the teammates are working on.

Competency levels
The Essentials of Modern Software Engineering

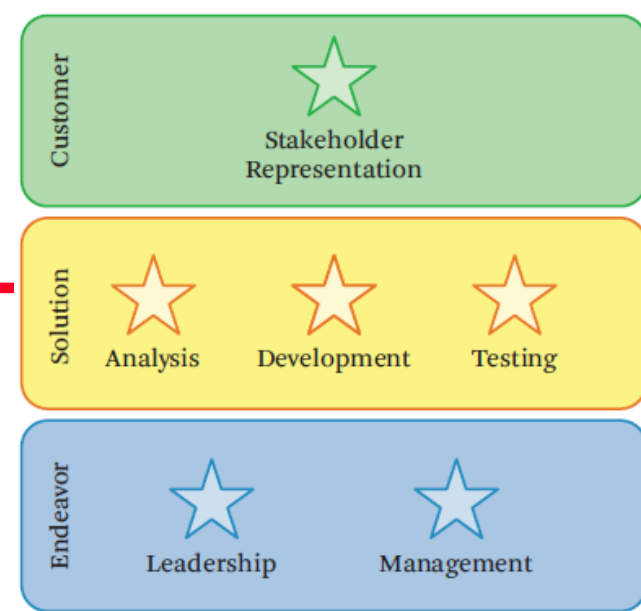


Competency level	Description
Assists	Demonstrates a basic understanding of the concepts and can follow instructions.
Applies	Able to apply the concepts in simple contexts by routinely applying the experience gained so far.
Masters	Able to apply the concepts in most contexts and has the experience to work without supervision.
Adapts	Able to apply judgment on when and how to apply the concepts to more complex contexts. Can enable others to apply the concepts.
Innovates	A recognized expert, able to extend the concepts to new contexts and inspire others.

Figure 6.6 The kernel competencies
The Essentials of Modern Software Engineering



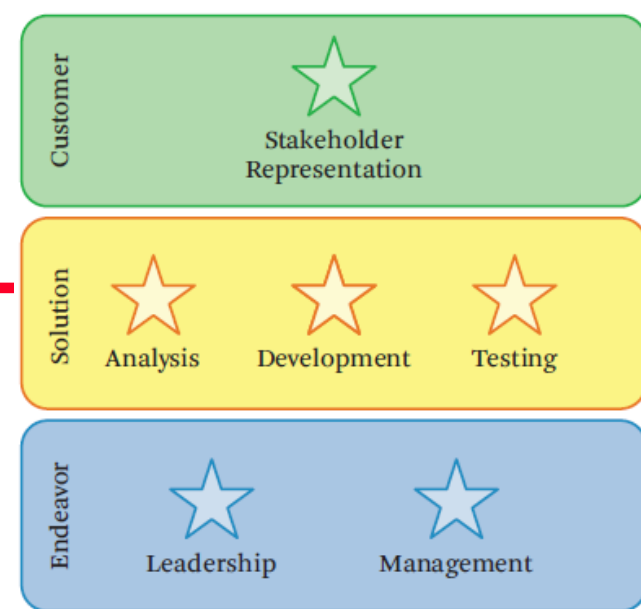
Essence kernel Customer competency



A clear understanding of the business and technical aspects of the domain of the problem and ability to accurately communicate the views of the stakeholders.

Stakeholder Representation. The ability to gather, communicate, and balance the needs of other stakeholders, and accurately represent their views.

Essence kernel Solution competency



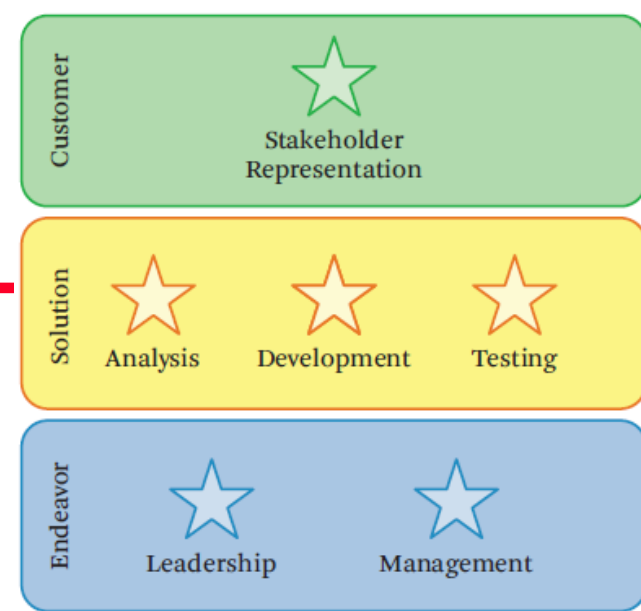
The ability to capture and analyze the requirements and build and operate a software system that fulfills them.

Analysis. The ability to understand opportunities and their related stakeholder needs, and to transform them into an agreed upon and consistent set of requirements.

Development. The ability to design, program, and code effective and efficient software systems following the standards and norms agreed upon by the team.

Testing. The ability to test a system, and verify that it is usable and that it meets the requirements.

Essence kernel Endeavor competency



Abilities to organize itself and manage its workload.

Leadership. The competency enables a person to inspire and motivate a group of people to achieve a successful conclusion to their work and to meet their objectives.

Management. The ability to coordinate, plan, and track the work done by a team.

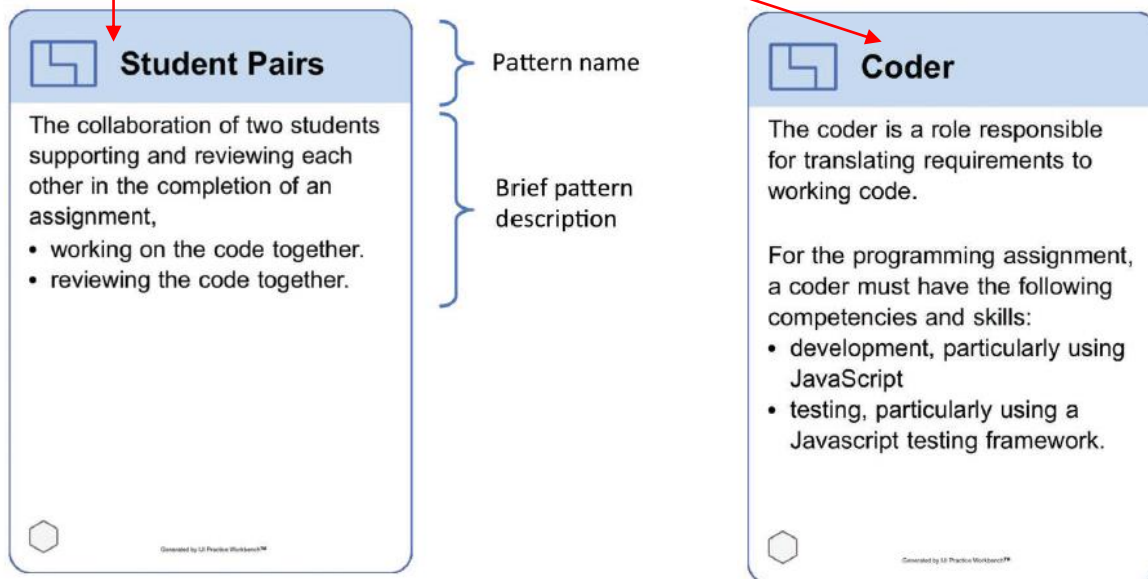
Essence kernel Patterns

Pattern = generic solutions to a typical problem.

Optional elements (not required in a practice definition) that may be associated with any other language element.

Role pattern : a set of specific responsibilities plus the competencies required to fulfill them and a minimum level for each competency.

Examples :

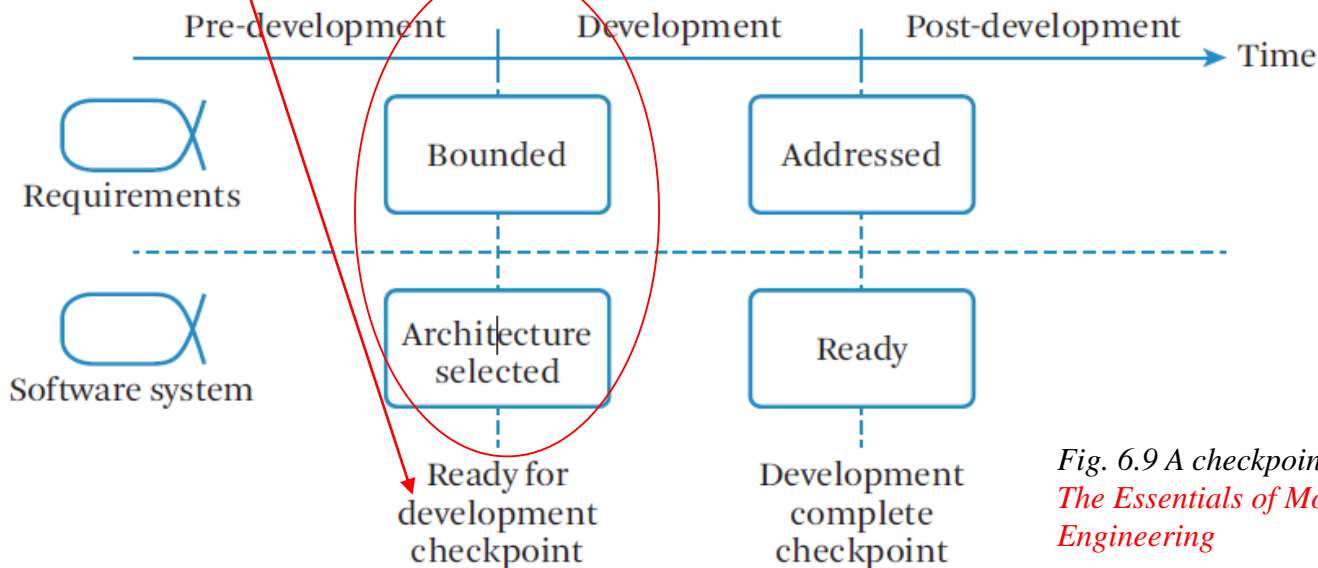


Figures 6.7 and 6.8 Student Pairs and Coder (role) patterns
The Essentials of Modern Software Engineering

Essence kernel Patterns

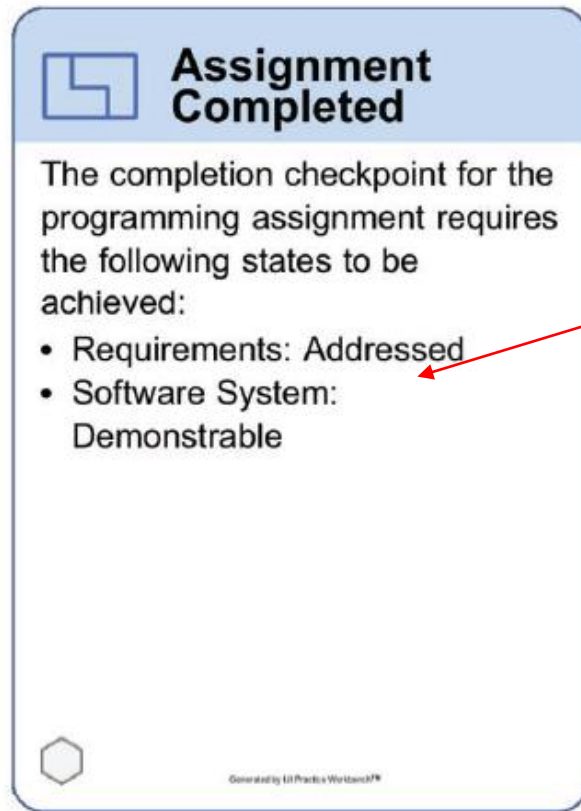
Checkpoint = a set of criteria to be achieved at a specific point in time in a development endeavor; key point in the lifecycle of a software endeavor where an important decision must be made.

Checkpoint pattern - expressed in Essence by a set of *alpha states* that must have been achieved in order to pass the *checkpoint (milestone)*.



*Fig. 6.9 A checkpoint pattern example
The Essentials of Modern Software
Engineering*

Essence kernel Patterns



*Checkpoint pattern card - expressed in Essence by a set of α states that must have been achieved in order to pass the *checkpoint*.*

*Fig. 6.10 A checkpoint pattern card
The Essentials of Modern Software
Engineering*

Bibliography

<http://semat.org>

<http://semat.org/tool-support>

<https://essence.ivarjacobson.com/>

<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>

Licence projects

Alpha State Cards are a simple, easy way to track status of a software project and help plan next steps.

Alpha State Cards are another tool in your kit, and can be helpful with:

- Understanding the current state of your development project
- Troubleshooting problem areas within a project
- Setting objectives for future iterations
- Facilitating retrospectives
- Identifying areas for improvement

We typically use them on a wall or whiteboard to visually display where we are and what we want to focus on next.

<https://essence.ivarjacobson.com/>



Topics covered

Software Engineering methods and SEMAT solution

Essence ideas and key concepts

The language of Essence

The kernel of Essence

Serious games

Essence cards

card provides a concise description of the most important information about its element.

act as reminders to practitioners

Additional details are available in complementary guidelines.

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

Cards 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

Serious 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

Serious games Progress Poker

Track progress based on the state transitions of alphas.

Items in a checklist

- provide a hint of what needs to be done.
- are subject to interpretation by the team members with different opinions on the meaning



need for agreement

Solution : Progress Poker

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

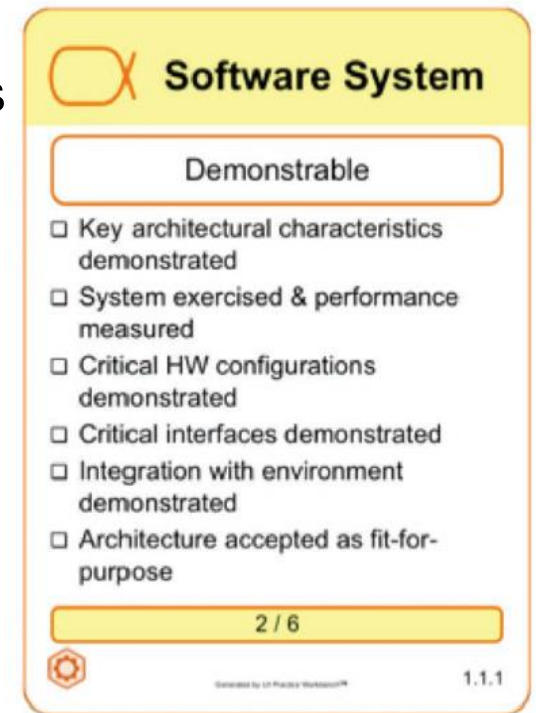
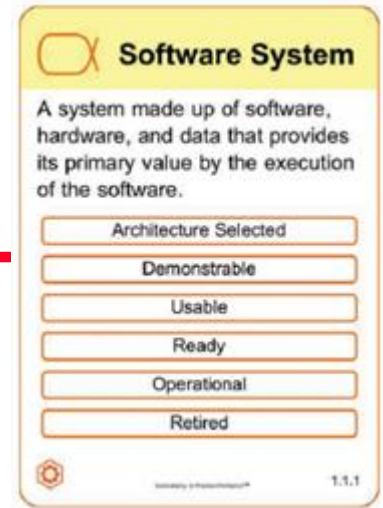


Fig. 8.1 Software System: Demonstrable alpha state card
The Essentials of Modern Software Engineering

Serious games

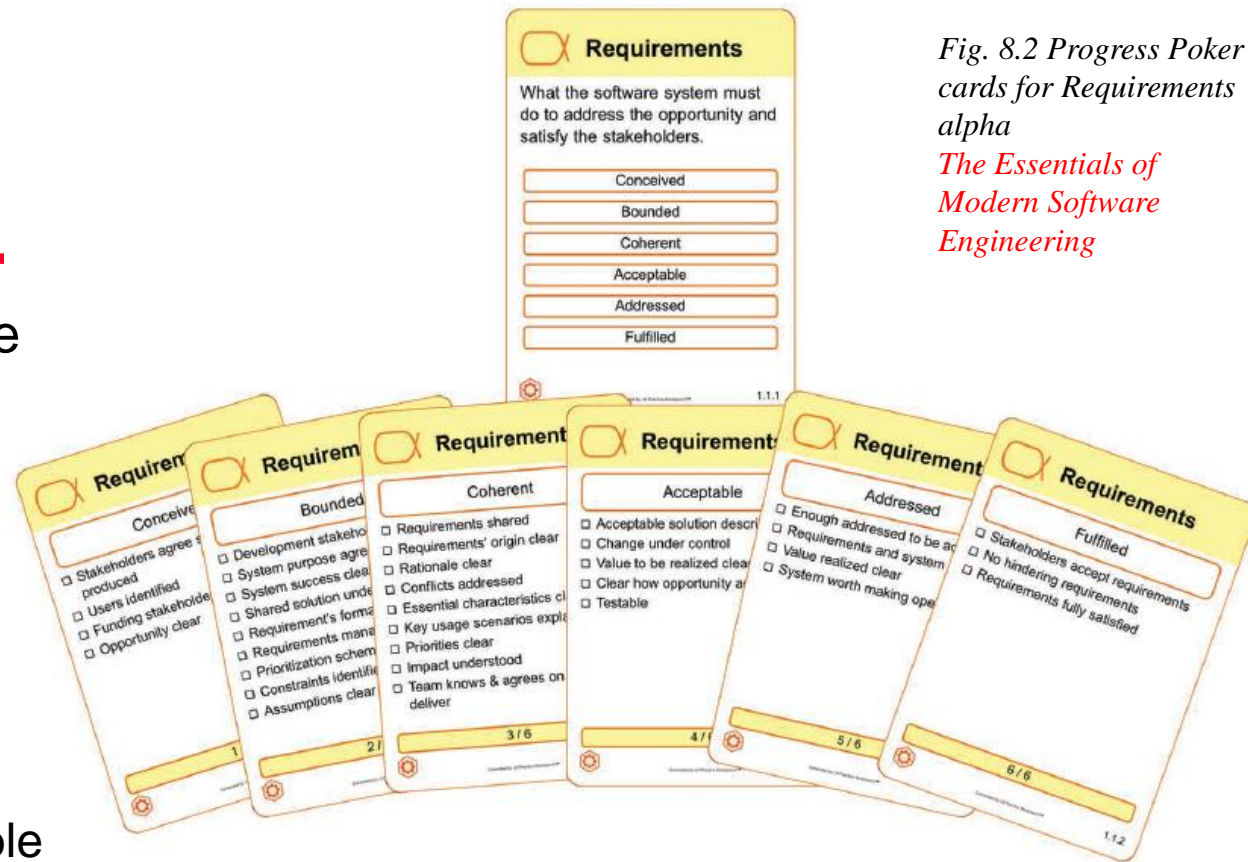
Progress Poker

- one alpha at a time
- Alpha overview card
- Alpha state cards
- 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

Ends when a consensus has been reached on the current state that has been achieved for a particular alpha.



Serious games

Chasing the state



- all the alphas
- Alpha overview card
- Alpha state cards
- 3-9 players

Rules

For each alpha

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

Serious games

Chasing the state

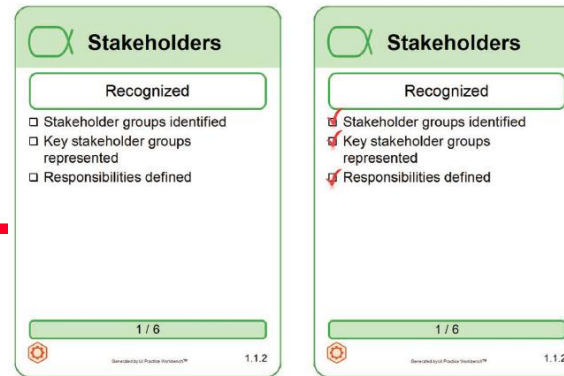


Fig. 8.5 Stakeholders alpha before and after discussion
The Essentials of Modern Software Engineering

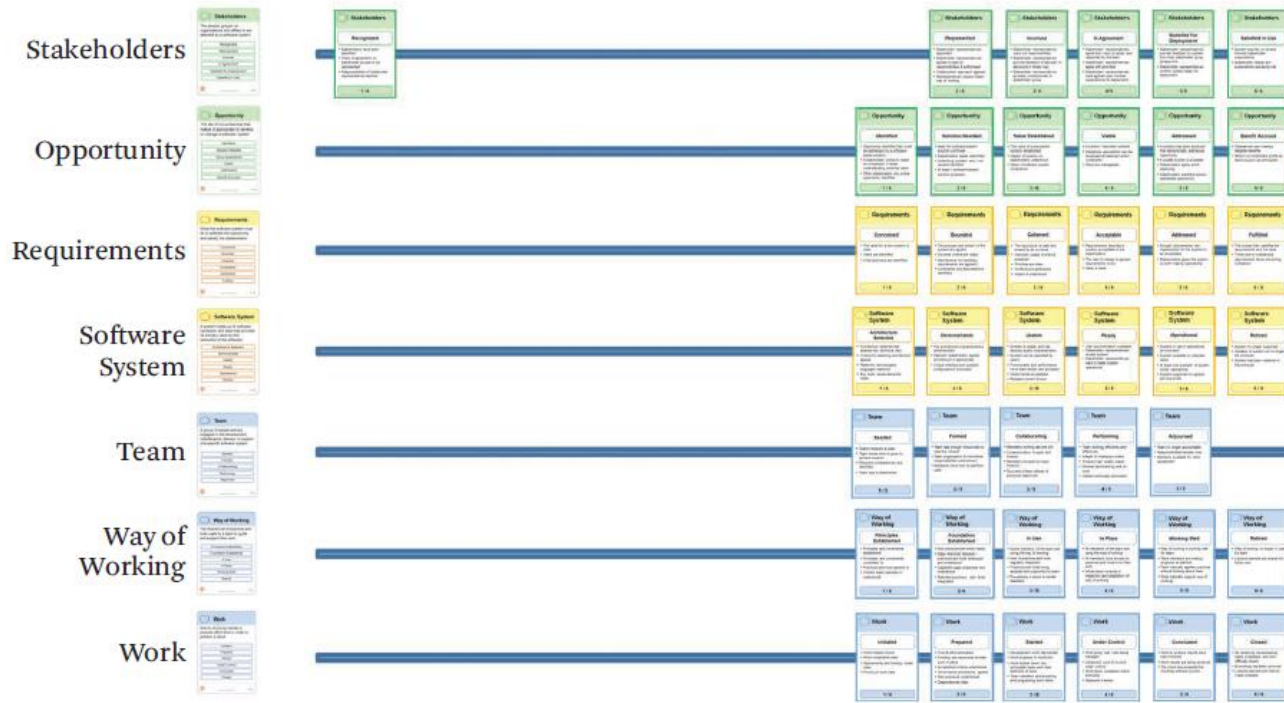


Fig. 8.6 Stakeholders alpha has reached first state
The Essentials of Modern Software Engineering

Serious games

Chasing the state result

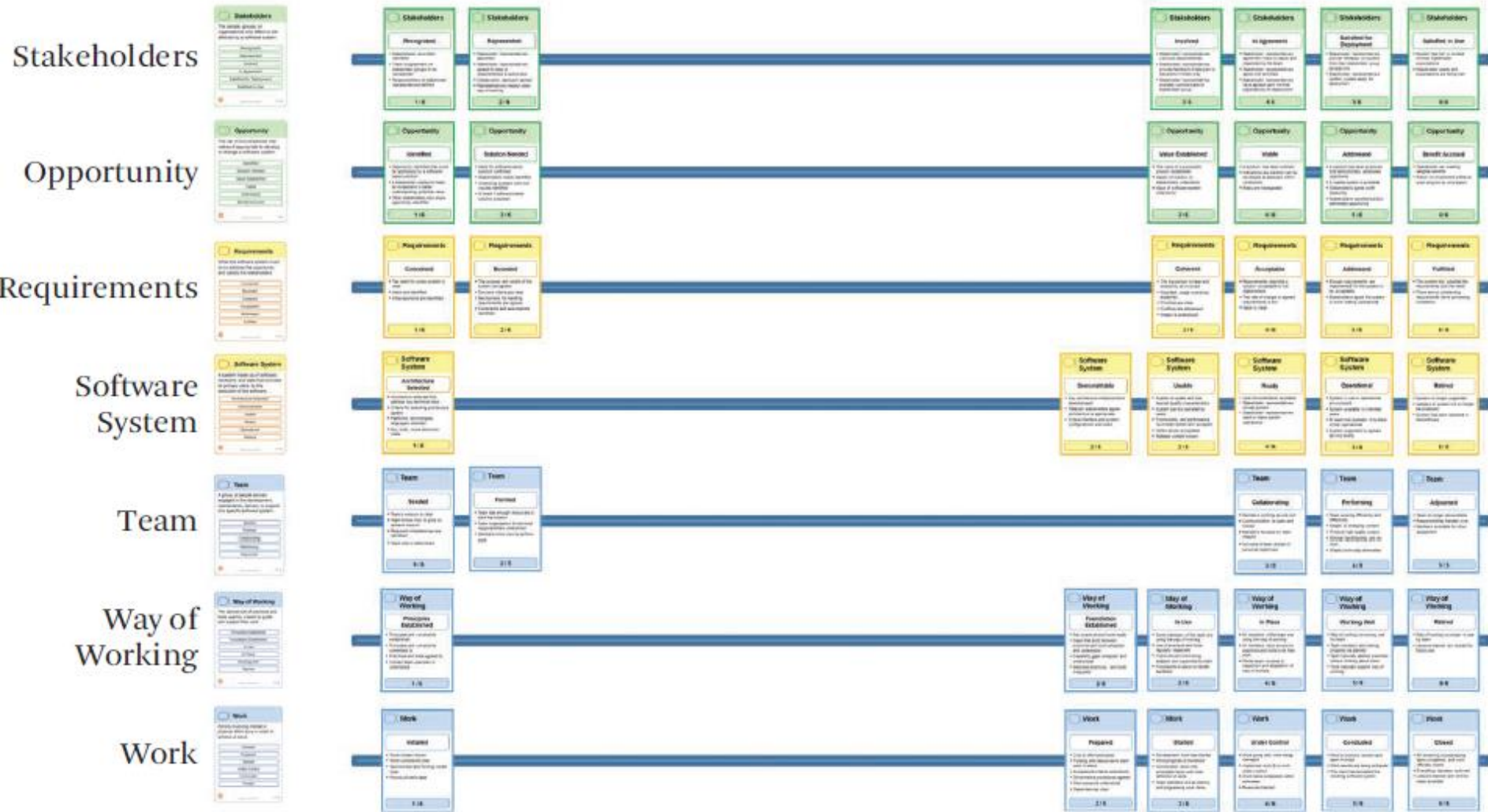


Fig. 8.8 The current states for all alphas have been identified
The Essentials of Modern Software Engineering

Serious games

Objective Go

Aim - agree upon the next steps

Played after the Chasing the State game

Objective : some or all alphas moved to the next state

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

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

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

1.1.2

Stakeholders

Involved

- Representatives assist the team
- Timely feedback and decisions provided
- Changes promptly communicated

3 / 6

1.1.2

Serious games

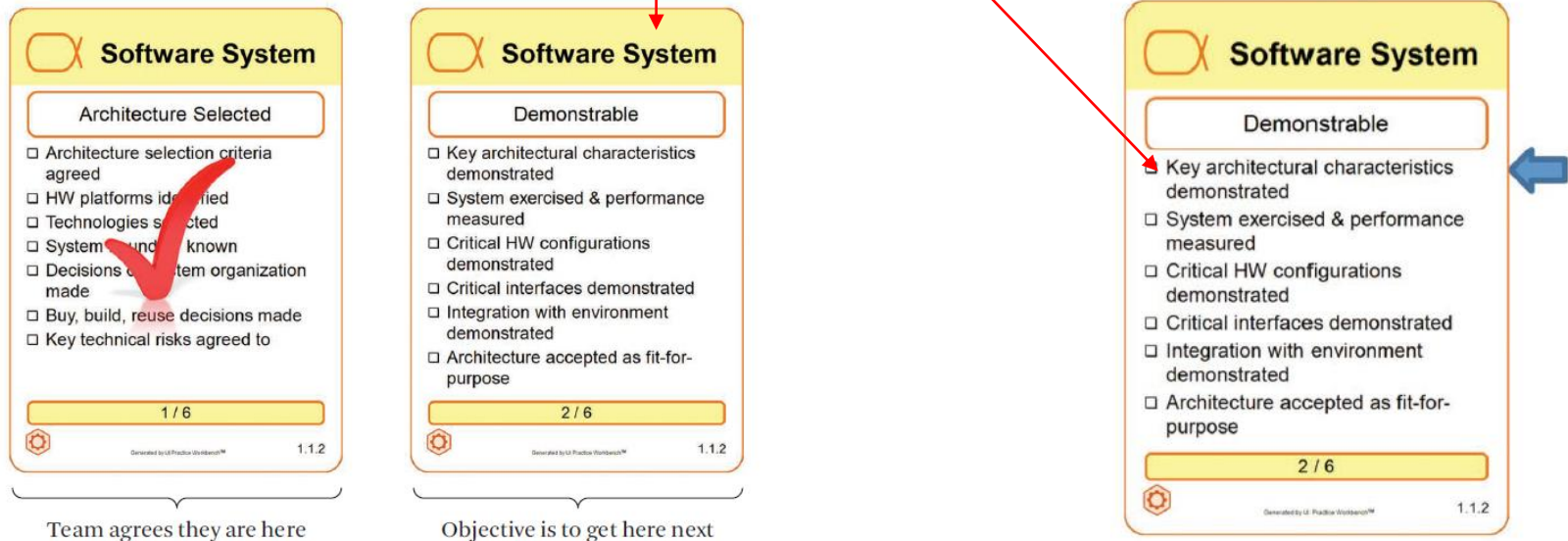
Objective Go example

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

Serious 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

Serious games

Checkpoint construction

Checkpoint = key point in the lifecycle of a software endeavor where an important decision must be made.

defined as a set of criteria to be achieved at a specific point in time in a development endeavor;

defined using alpha states

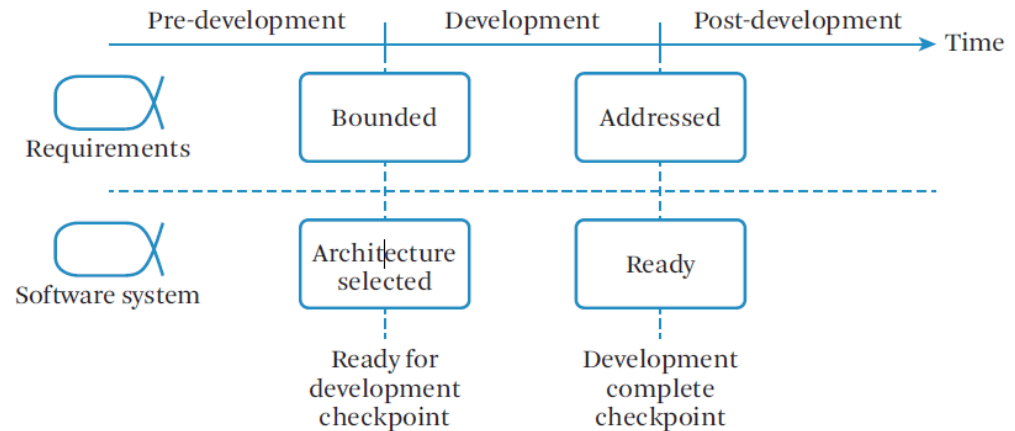
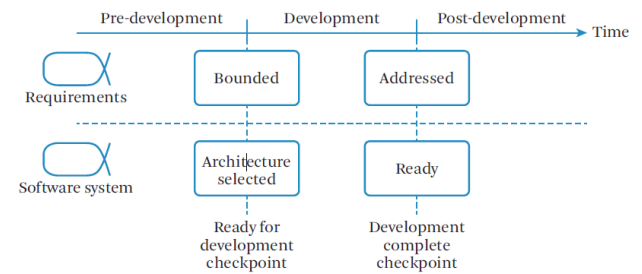


Fig. 8.9 Requirements and Stakeholders Alpha Wave
The Essentials of Modern Software Engineering

Serious games

Checkpoint construction



used to synchronize teams working in parallel \Rightarrow specified by the stakeholders of the whole endeavor and not by every team participating in the endeavor \Rightarrow game is played by the stakeholder team

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

played for one checkpoint and in two rounds.

Rules

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.

2. for each selected alpha

- each team member identifies the state he believes the alpha needs to be in to pass the checkpoint
- discuss different choices (explain and motivate, starting with extreme ones)
- until consensus is obtained

Serious games

Checkpoint construction

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



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.

Bibliography

<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>

Bibliography

<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>

Key points

<https://www.youtube.com/watch?v=BEp-L8ChSkQ>

Scrum is an agile method that focuses on agile planning and management. Unlike XP, it does not define the engineering practices to be used. The development team may use any technical practices that they believe are appropriate for the product being developed.

In Scrum, work to be done is maintained in a product backlog – a list of work items to be completed. Each increment of the software implements some of the work items from the product backlog.

Sprints are fixed-time activities (usually 2–4 weeks) where a product increment is developed. Increments should be ‘potentially shippable’ i.e. they should not need further work before they are delivered.

A self-organizing team is a development team that organizes the work to be done by discussion and agreement amongst team members.

Scrum practices such as the product backlog, sprints and self-organizing teams can be used in any agile development process, even if other aspects of Scrum are not used.