Software Engineering Project

Morgan Ericsson

morgan@cse.gu.se

@morganericsson

morganericsson

UNIVERSITY OF GOTHENBURG

CHALMERS

Software Engineering

- The branch of computer science that creates practical, cost-effective solutions to computing and information processing problems
- *"Application of engineering to software"*
  - systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software
- Assure the quality of the process and the product

Five Steps

Requirements

Design

Implementation

Quality Assurance

Evolution

A sequential model of software development

Change is Ubiquitous

Design

Requirements

Implementation

Quality Assurance

An iterative model of software development

Production vs. Creation

SEMAT

Software engineering is gravely hampered today by immature practices. Specific problems include:

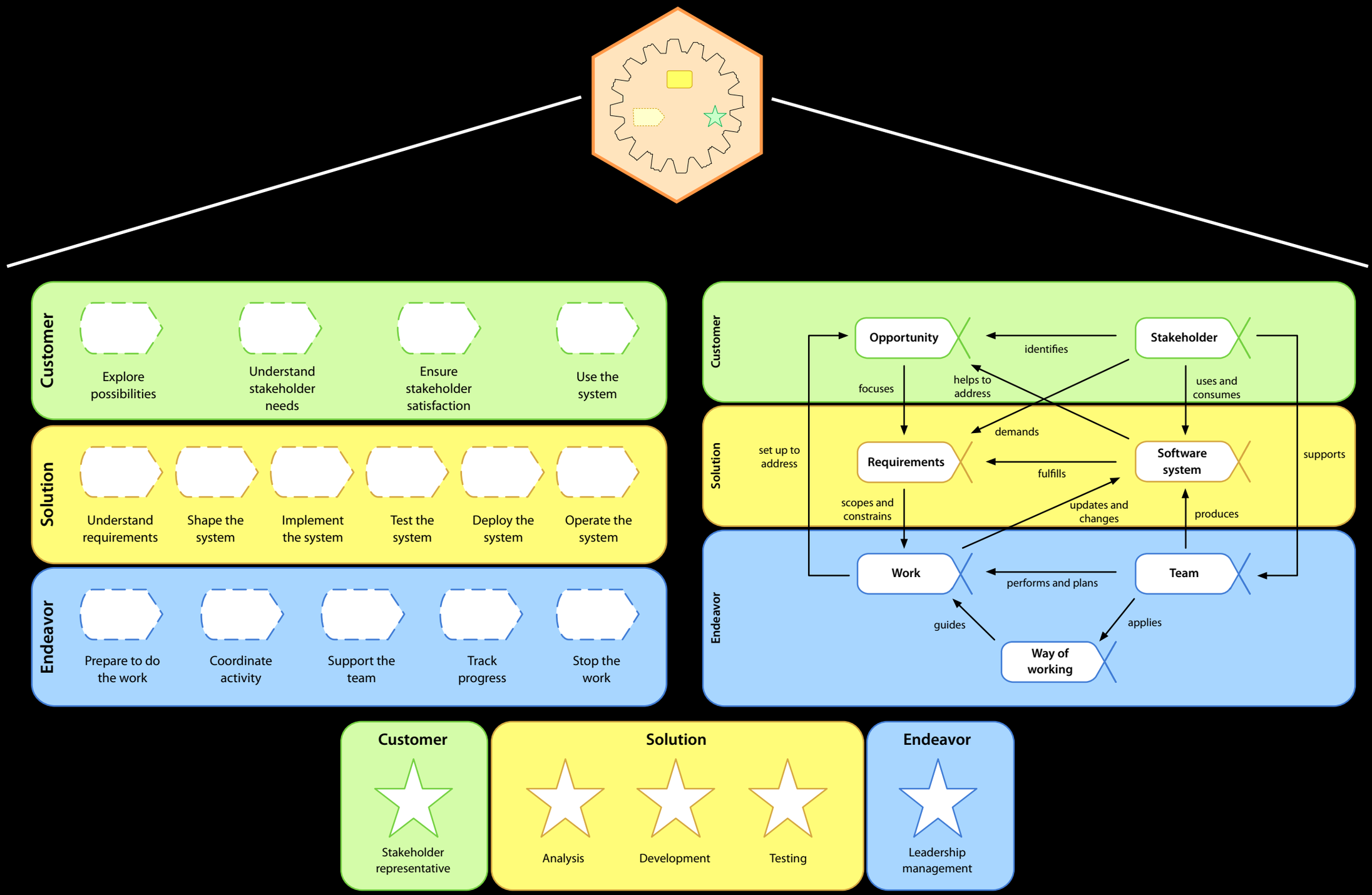
- The prevalence of fads more typical of a fashion industry than of an engineering discipline
- The lack of a sound, widely accepted theoretical basis
- The huge number of methods and method variants, with differences little understood and artificially magnified
- The lack of credible experimental evaluation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

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

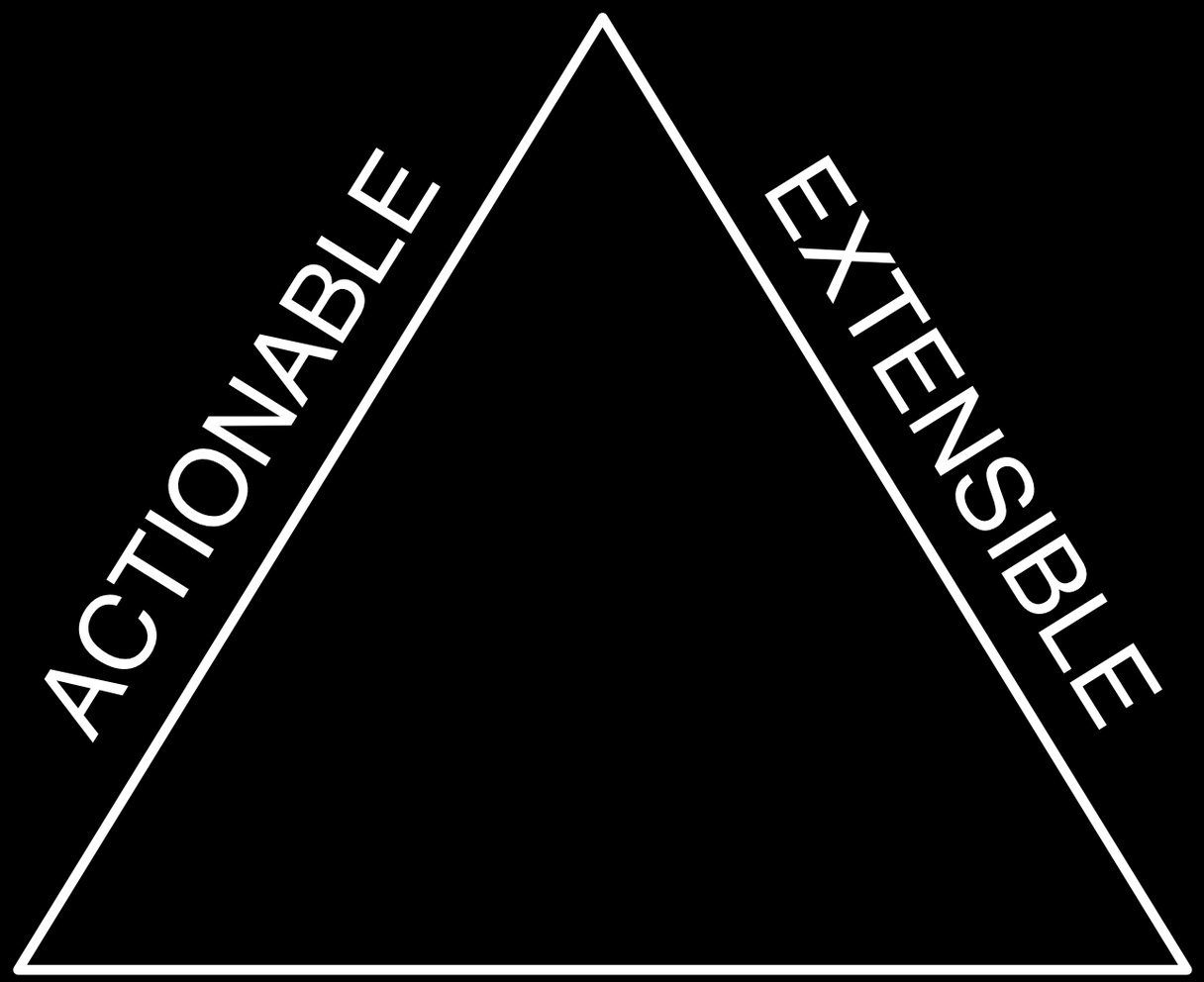
- Include a kernel of widely-agreed elements, extensible for specific uses
- Address both technology and people issues
- Are supported by industry, academia, researchers and users
- Support extension in the face of changing requirements and technology

Excerpt from the *SEMAT Call for Action*

SEMAT (cont'd)


Customer
Explore possibilities
Understand stakeholder needs
Ensure stakeholder satisfaction
Use the system
Solution
Understand requirements
Shape the system
Implement the system
Test the system
Deploy the system
Operate the system
Endeavor
Prepare to do the work
Coordinate activity
Support the team
Track progress
Stop the work
Customer
Opportunity
identifies
Stakeholder
focuses
helps to address
uses and consumes
Solution
set up to address
Requirements
demands
fulfills
Software system
supports
scopes and constrains
updates and changes
produces
Endeavor
Work
performs and plans
Team
guides
applies
Way of working
Customer
Stakeholder representative
Solution
Analysis
Development
Testing
Endeavor
Leadership management

SEMAT (cont'd)



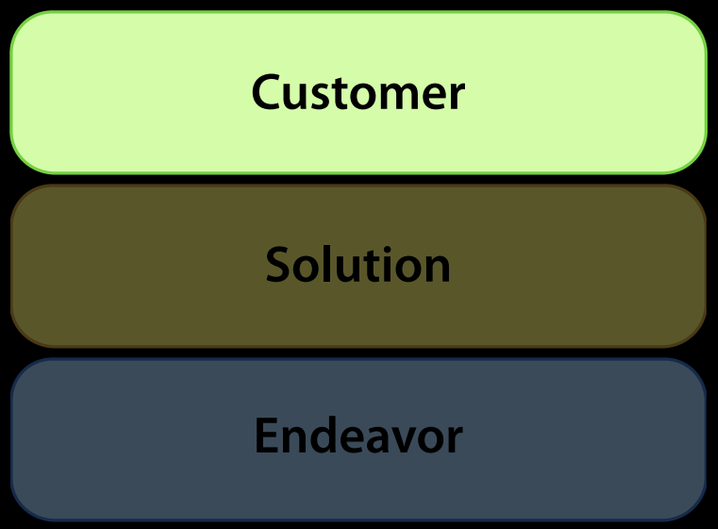
Areas of concern

Customer

Solution

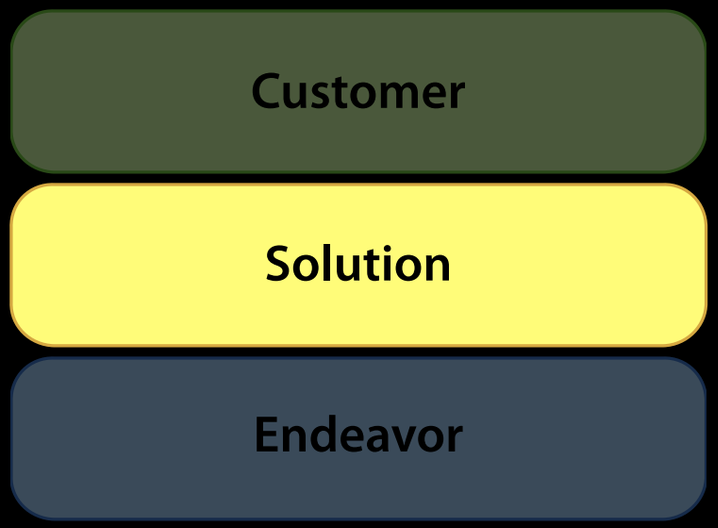
Endeavor

Areas of concern



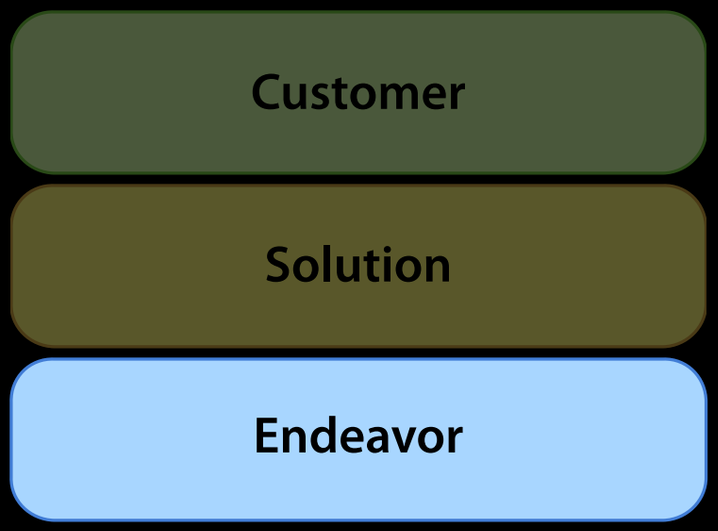
- Software development always involves at least one customer
- The Customer AoC involves everything to do with use and exploitation of the developed system
- The customer perspective should be integrated into day to day work

Areas of concern



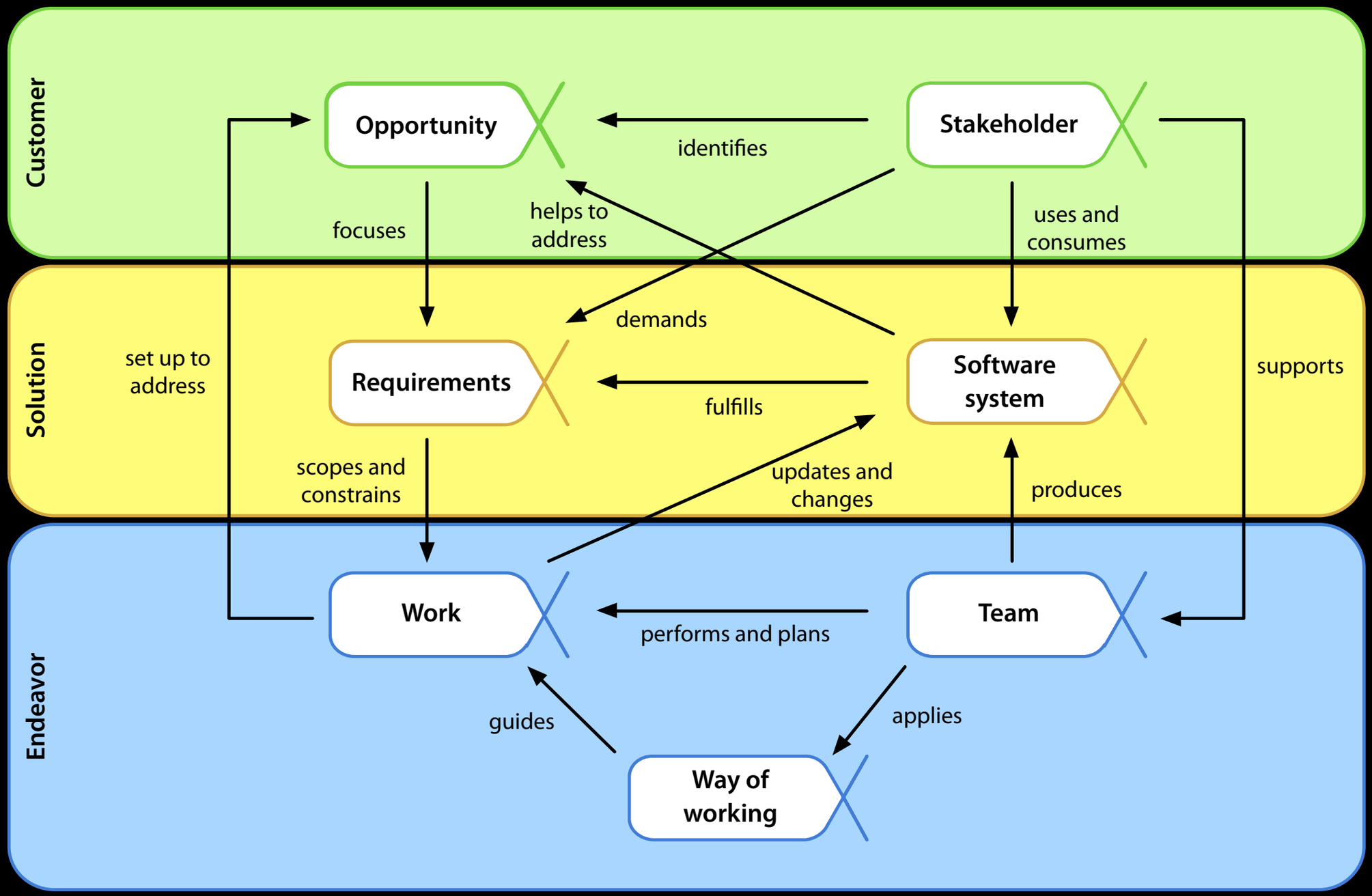
- The goal is to develop a working system to solve some problem
- The Solution AoC contains everything related to specification and development

Areas of concern



- Software usually takes significant time and effort to develop …
- … affects many people …
- … and involves a team
- The Endeavor AoC contains everything related to the team and way of work (WoW)

Kernel alphas



- Circumstances that make it appropriate to change or develop a software system
- The opportunity articulates the reasons why
- The team's shared understanding of stakeholders' need
- Helps shape the requirements

- People, groups, or organizations that affect or are affected by a software system

- Provide the opportunity

- Involved throughout the endeavor

  - support the team

  - ensure acceptable product

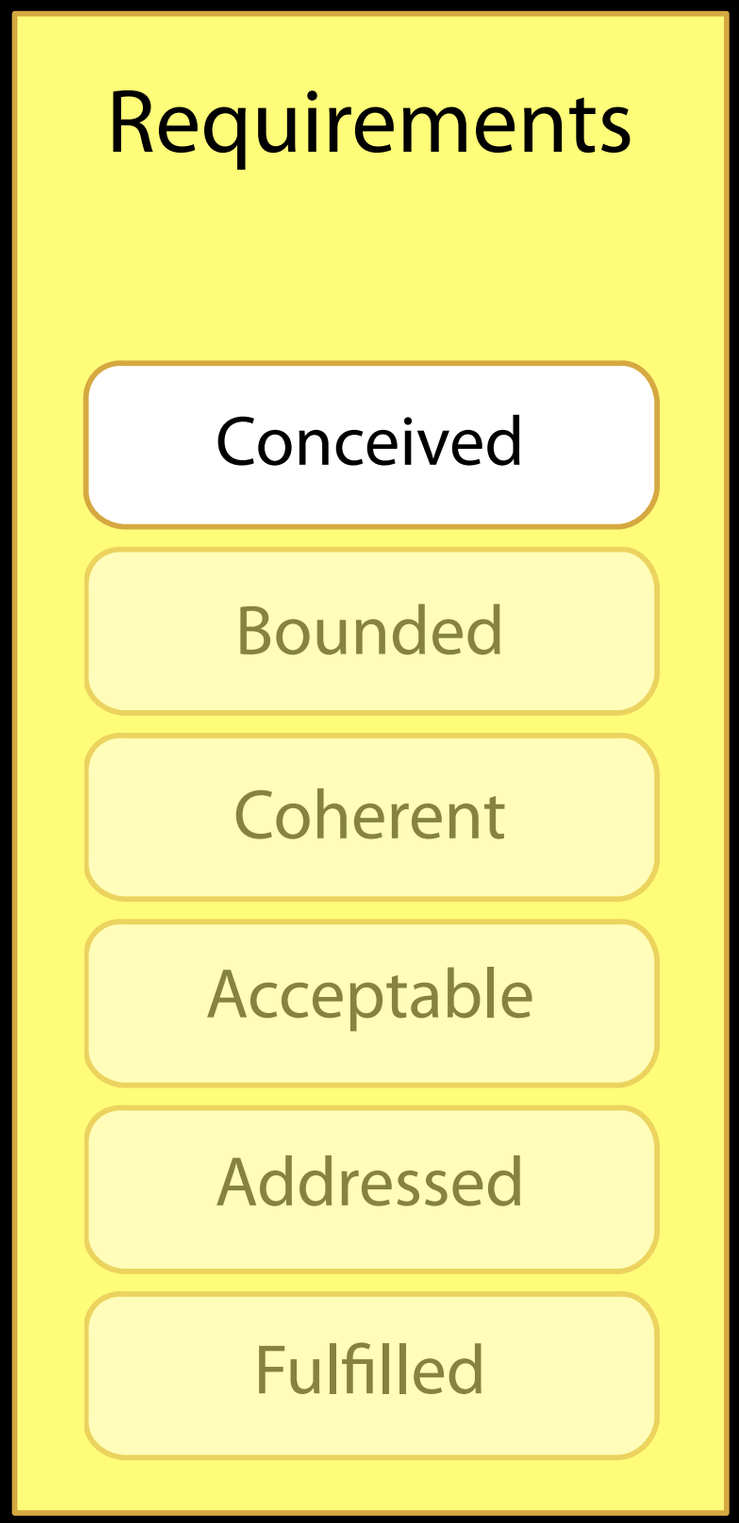
- What the software system must do to
  - address the opportunity
    - satisfy the stakeholders
- Important to discover, share and understand
- Drive the development and testing

- A system made up of software, hardware, and data
  - provides primary value by execution of software
- Can be part of a larger software, hardware, business, or social situation

- Group of people actively engaged in
  - development
  - maintenance
  - delivery
  - support
- Plans and performs work to create, update, or change the software system

- Everything the team does to work with a software system
  - matching requirements
  - addressing opportunity
  - …
- Guided by practices

- The practices and tools used by a team to guide and support their work
- Evolves the team's understanding
- Continuously reflected upon and adapted



- The initial set of stakeholders agree that a system is to be produced
- There is a clear opportunity for the new system to address
- The stakeholders that will use the new system are identified
- The stakeholders that will fund the initial work on the new system are identified

Requirements

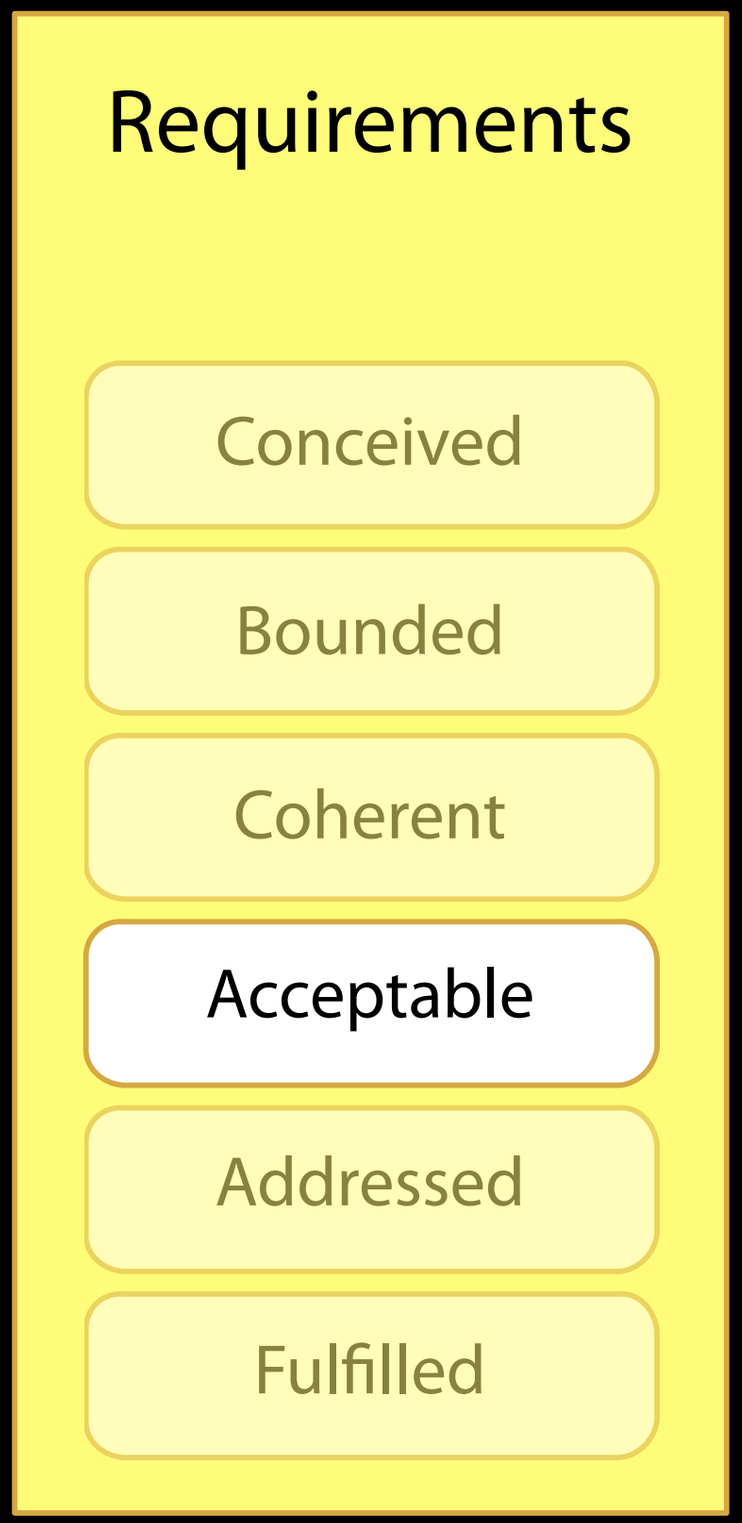
- Conceived
- Bounded
- Coherent
- Acceptable
- Addressed
- Fulfilled

- The stakeholders involved in developing the new system are identified
- The stakeholders agree on the purpose of the new system
- It is clear what success is for the new system
- The stakeholders have a shared understanding of the extent of the proposed solution
- The way the requirements will be described is agreed on
- The mechanisms for managing the requirements are in place
- The prioritisation scheme is clear
- Constraints are identified and considered
- Assumptions are clearly stated

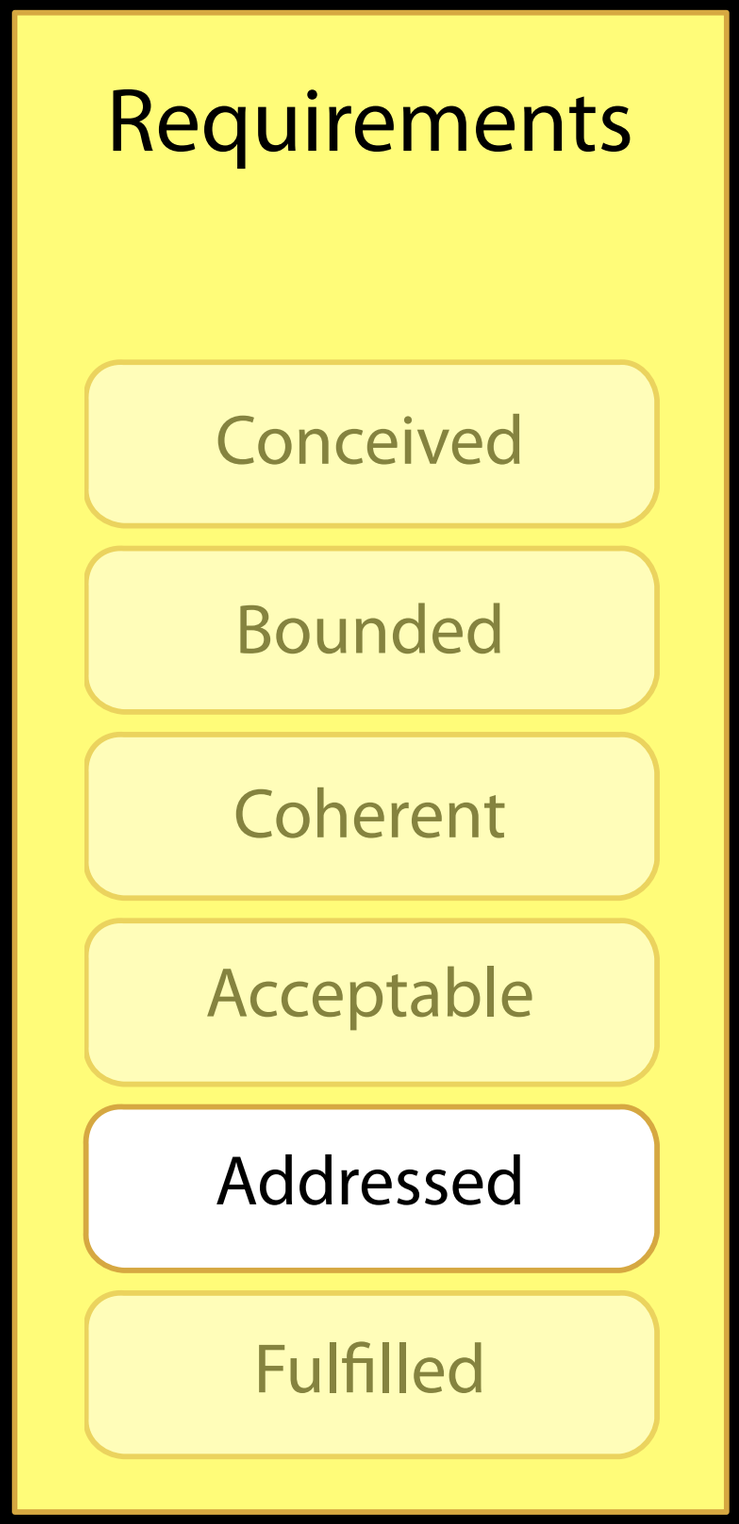
Requirements

- Conceived
- Bounded
- Coherent
- Acceptable
- Addressed
- Fulfilled

- The requirements are captured and shared with the team and the stakeholders
- The origin of the requirements is clear
- The rational behind the requirements are clear
- Conflicting requirements are identified and attended to
- The requirements communicate the essential characteristics of the system to be delivered
- The most important usage scenarios for the system can be explained
- The priority of the requirements is clear
- The impact of implementing the requirements is understood
- The team understands what has to be delivered and agrees to deliver it



- The stakeholders accept that the requirements describe an acceptable solution
- The rate of change to the agreed-on requirements is relatively low and under control
- The value provided by implementing the requirements is clear
- The parts of the opportunity satisfied by the requirements are clear



- Enough of the requirements are addressed for the resultant system to be acceptable to the stakeholders
- The stakeholders accept the requirements as accurately reflecting what the system does and does not do
- The set of requirement items implemented provides clear value to the stakeholders
- The system implementing the requirements is accepted by the stakeholders as worth making operational

- The stakeholders accept the requirements as accurately capturing what they require to fully satisfy the need for a new system

- There are no outstanding requirement items preventing the system from being accepted as fully satisfying the requirements

- The stakeholders accept the system as fully satisfying the requirements

Alphas, states, and cards

REQUIREMENTS

Fulfilled

- The system fully satisfies the requirements and the need
- There are no outstanding requirements items preventing completion

1 / 6

REQUIREMENTS

Bounded

- The purpose and extent of the system are agreed on
- Success criteria are clear
- Mechanisms for handling requirements are agreed on
- Constraints and assumptions are identified

2 / 6

REQUIREMENTS

Coherent

- The big picture is clear and shared by all involved
- Important usage scenarios are explained
- Priorities are clear
- Conflicts are addressed
- Impact is understood

3 / 6

REQUIREMENTS

Acceptable

- Requirements describe a solution acceptable to the stakeholders
- The rate of change to agreed requirements is low
- Value is clear

4 / 6

REQUIREMENTS

Addressed

- Enough requirements are implemented for the system to be acceptable
- Users are identified
- Initial sponsors are identified

5 / 6

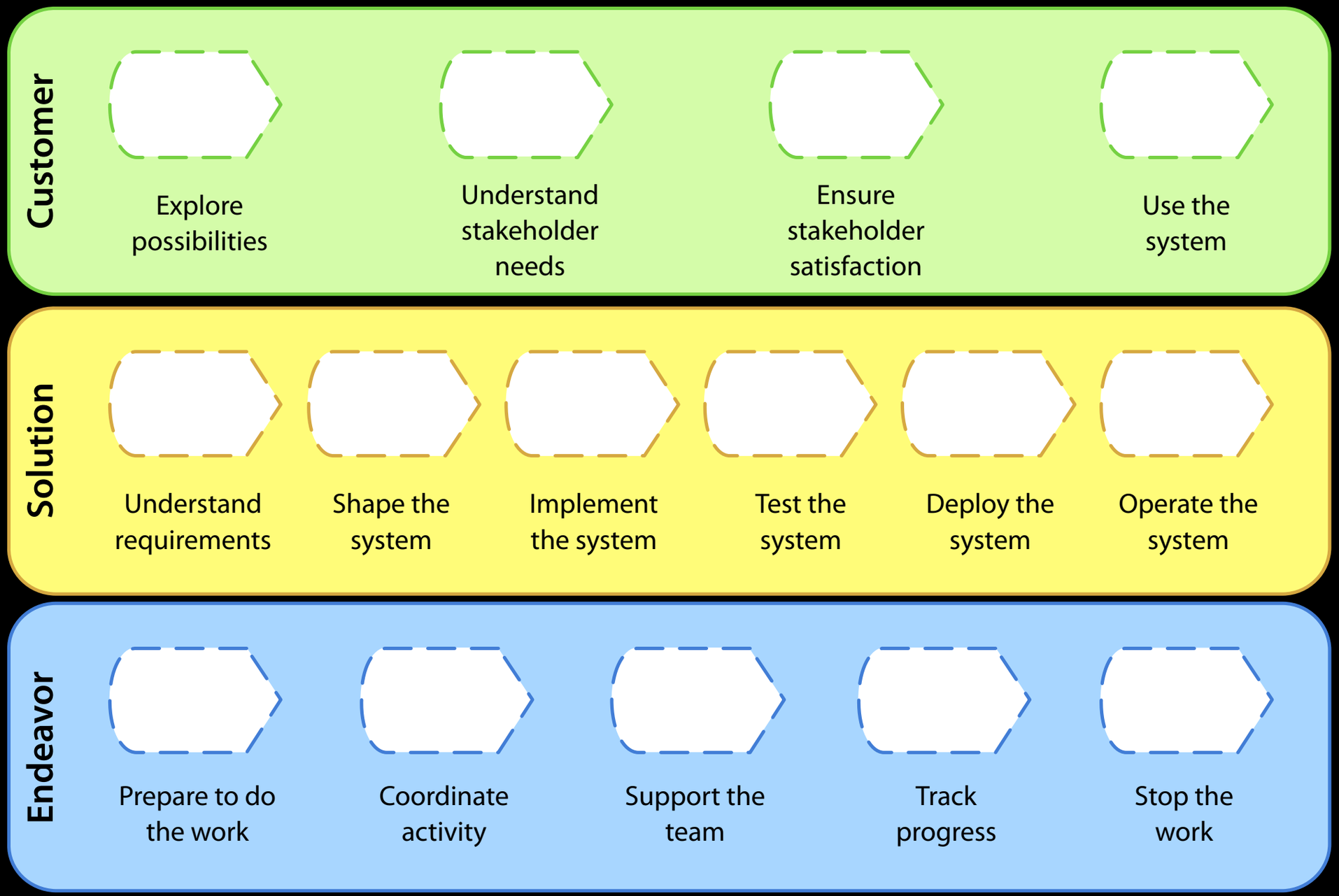
REQUIREMENTS

Fulfilled

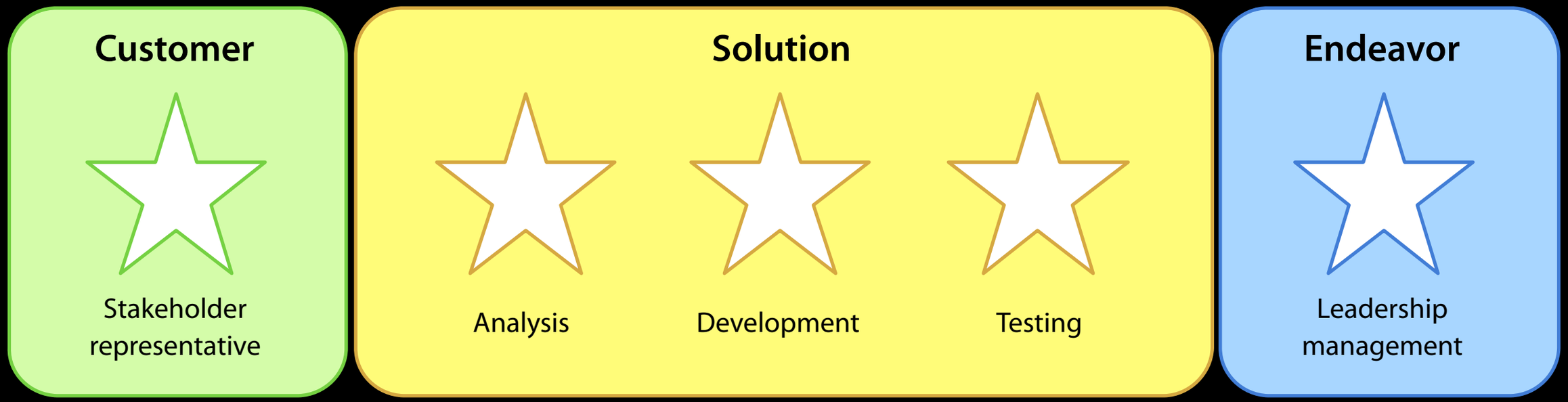
- The system fully satisfies the requirements and the need
- There are no outstanding requirements items preventing completion

6 / 6

Kernel activity space

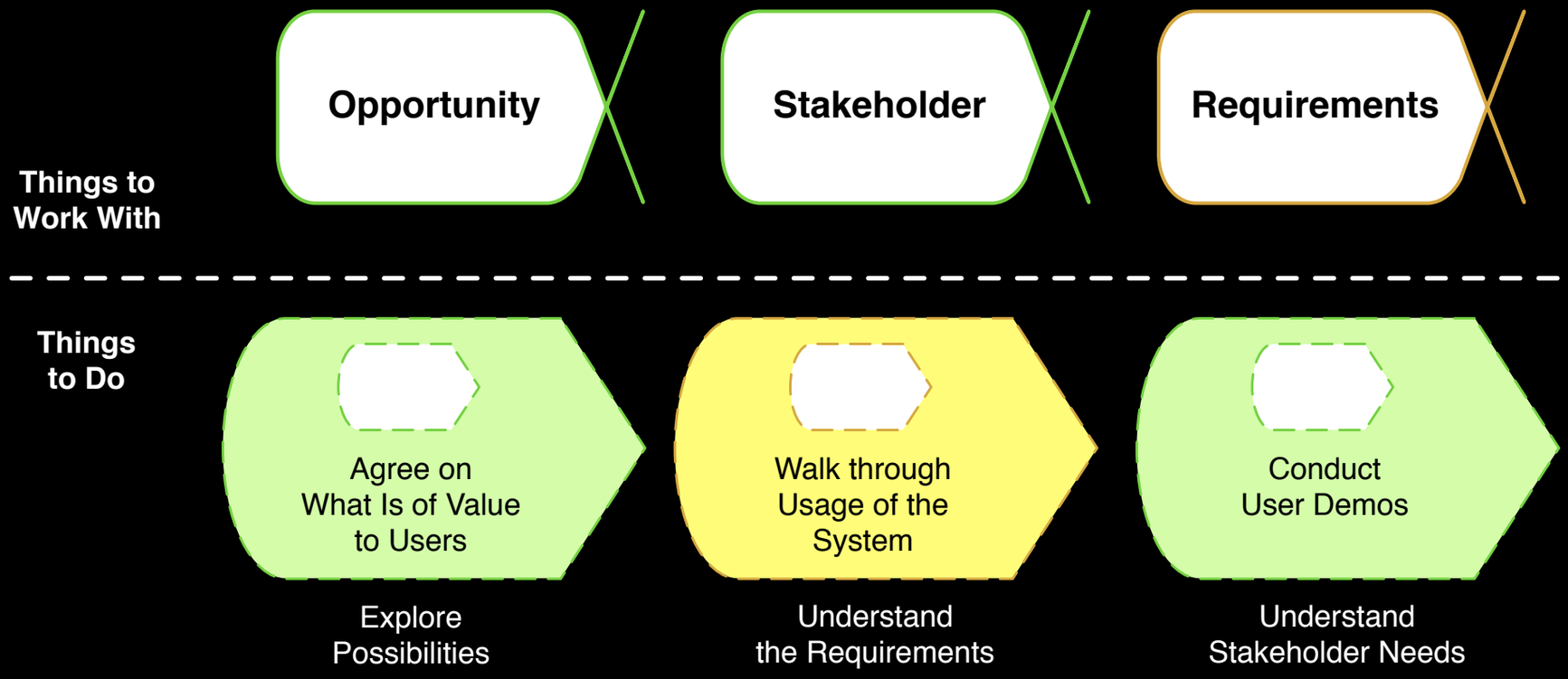


Competencies



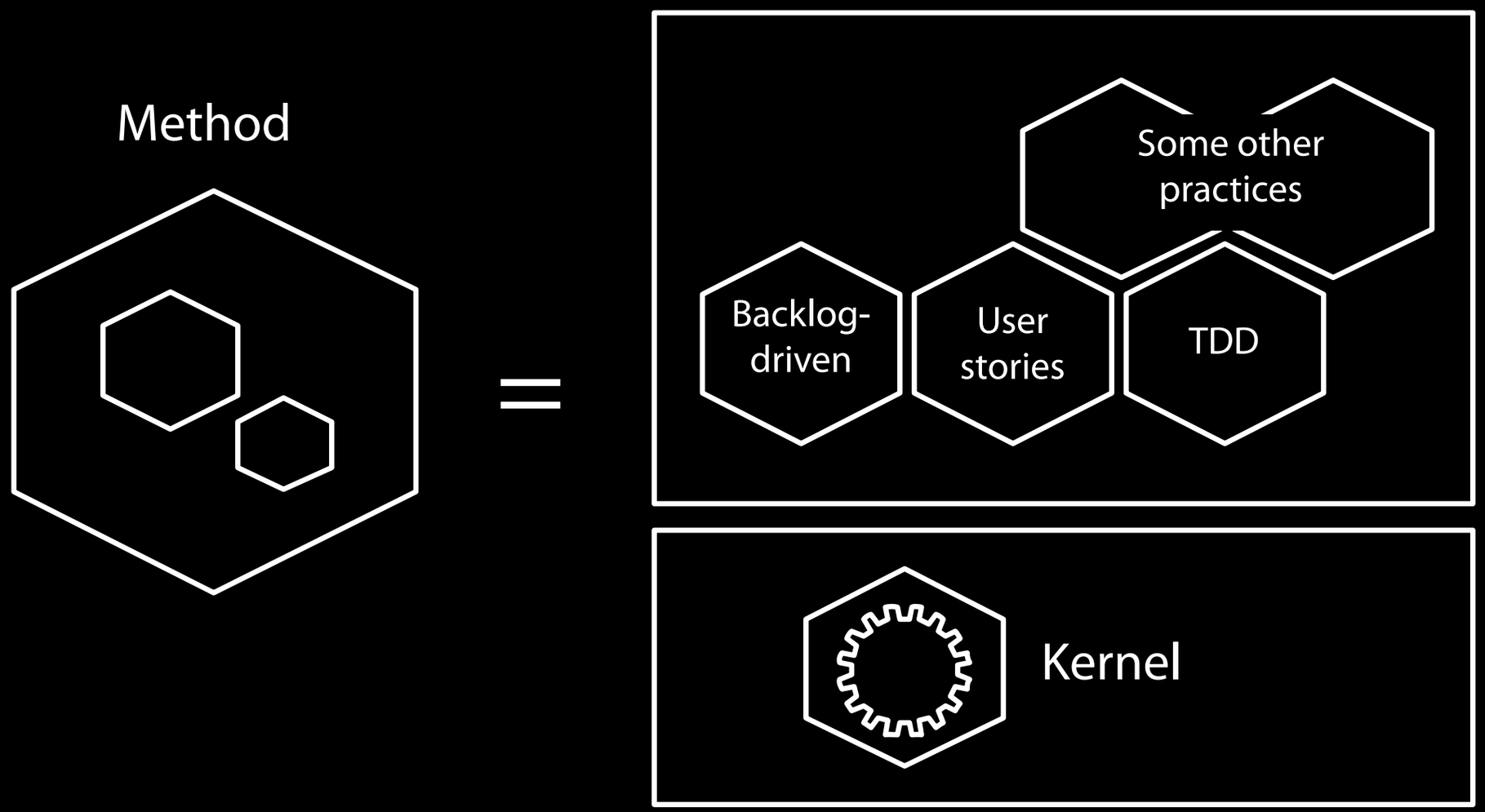
More details through practices

A practice is a repeatable approach to doing something with a specific purpose in mind.



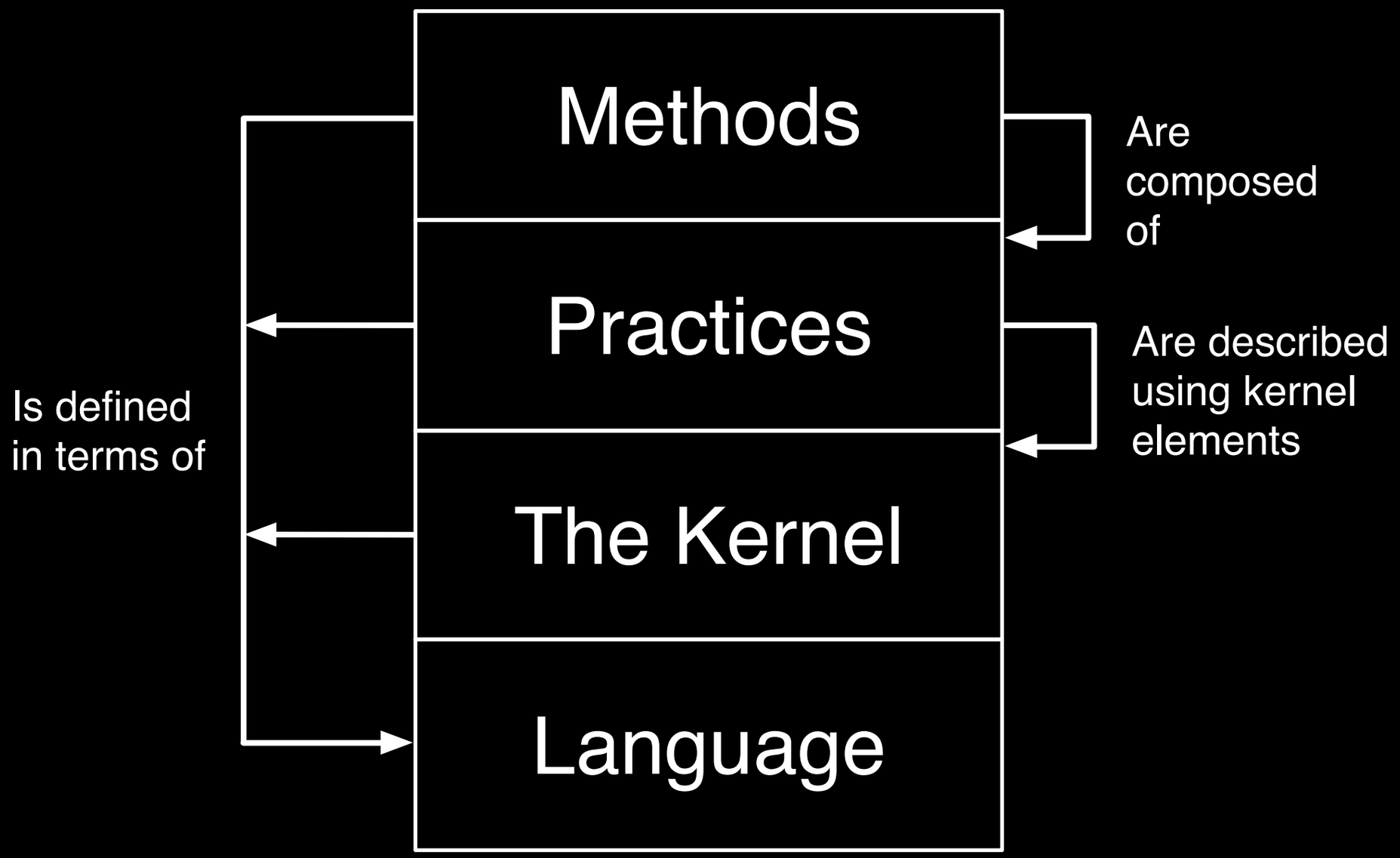
A requirements elicitations practice

Building methods from practice



A method is a composition of practices on top of the kernel

The bigger picture



Running iterations with the kernel

Example from *The Essence of Software Engineering: The SEMAT Kernel*

Background

- Our hero, Smith, needs to develop a mobile application to browse a social network offline
- The idea for the application was provided by Angela, the customer representative from the marketing department
- The department manager, Dave, is concerned with delivering value on time and within budget
- The application should cache contents, e.g., photos

Terminology

- **Iteration,** a timebox intended to work with a stable increment of a system. Typically two to four weeks. Can include any activities
- **Iteration objective,** specific objectives that should be achieved during the iteration. Often tied to progressing one more or alpha states
- **Iteration backlog,** objectives that should be achieved during the iteration. Can be refined with more detail by adding the tasks necessary to achieve the objectives.
- **Tasks,** a portion of work that can be clearly identified, isolated, and completed.

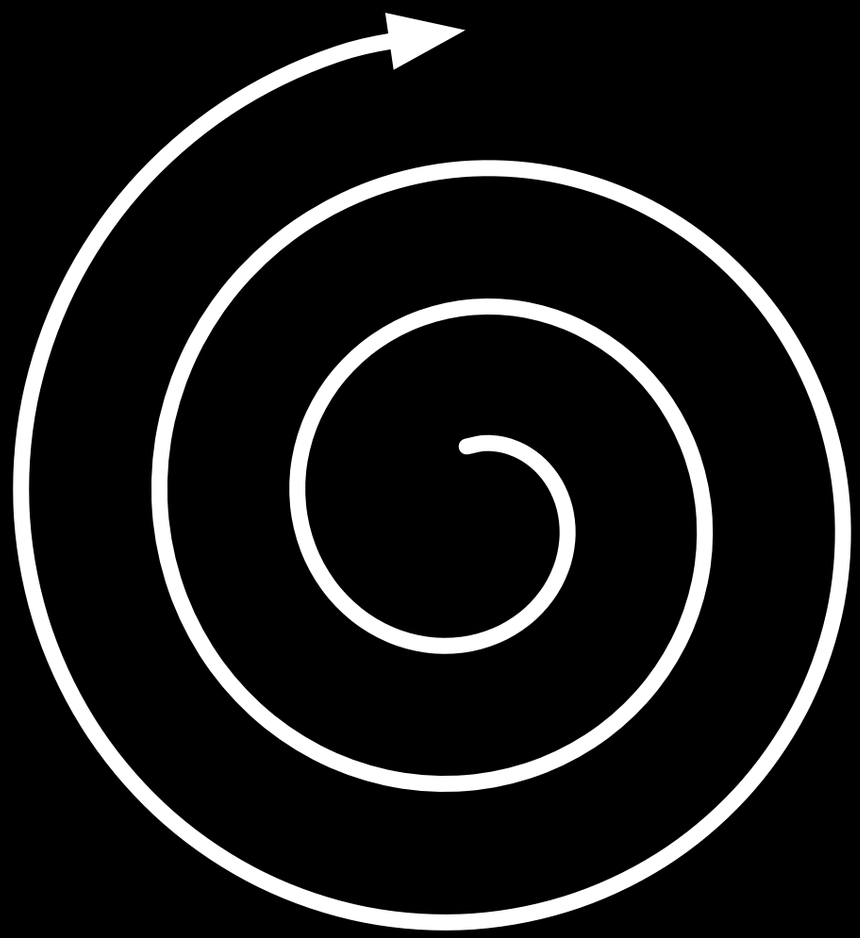
Plan-Do-Check-Adapt

Plan

Do

Check

Adapt



Plan and Do

- Plan
  - determine the current state
  - determine the next state
  - determine how to achieve the next state
- Do
  - work toward achieving the next state
  - remove obstacles as they occur

Check and Adapt

- Check
  - track the work
  - check that work is done
- Adapt
  - reflect on what happened
  - look for more suitable ways to work
  - improve the quality of work
  - reduce waste

Translated

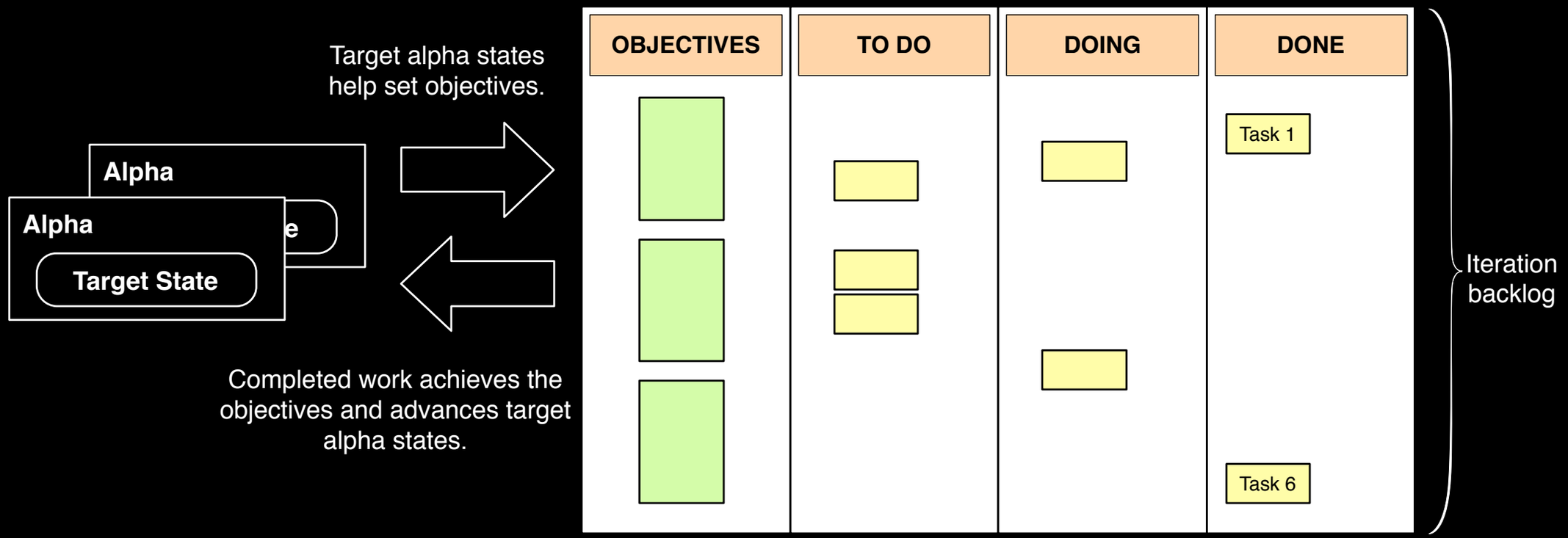
- Plan
  - determine the current state of the whole endeavor by looking at each alpha
  - define what alphas and what states to progress in the coming iteration
  - Identify tasks to complete to achieve these states

Translated (cont'd)

- Do
  - work on the identified tasks
    - e.g., write code, test, discuss requirements, write documentation, etc.
- Check
  - track objectives and tasks (and re-plan if needed)
  - track way of working

Translated (cont'd)

- Adapt
  - review way of working
  - identify better ways of doing things
  - if needed, change plans and way or working



The relationship between alphas, and objectives and tasks in the backlog

Setting the scene

- The company had very little in the way of formal processes
  - and was growing with many new hires
- Smith's team had two developers at start (himself and Bob)
  - later joined by Dick and Harriet
- The team is currently six weeks into development and have provided an early demonstration to stakeholders.

	Angela	Dave	Smith	Tom	Dick and Harriet
Opportunity	✔	✔	✔		
Stakeholder	✔	✔			
Requirements	✔	✔	✔	✔	✔
Software system	✔	✔	✔	✔	✔
Work	✔	✔	✔		
Team		✔	✔		
Way of working		✔	✔		✔

Views of the kernel of different participants

Alphas in focus

Requirements

Software system

Way of working

Alphas are universal, so even small endeavors will work with all of them. To simplify things, we focus on the there three.

Determining the current state

REQUIREMENTS

Fulfilled

- The system fully satisfies the requirements and the need
- There are no outstanding requirements items preventing completion

1 / 6

REQUIREMENTS

Bounded

- The purpose and extent of the system are agreed on
- Success criteria are clear
- Mechanisms for handling requirements are agreed on
- Constraints and assumptions are identified

2 / 6

REQUIREMENTS

Coherent

- The big picture is clear and shared by all involved
- Important usage scenarios are explained
- Priorities are clear
- Conflicts are addressed
- Impact is understood

3 / 6

REQUIREMENTS

Acceptable

- Requirements describe a solution acceptable to the stakeholders
- The rate of change to agreed requirements is low
- Value is clear

4 / 6

REQUIREMENTS

Addressed

- Enough requirements are implemented for the system to be acceptable
- Users are identified
- Initial sponsors are identified

5 / 6

REQUIREMENTS

Fulfilled

- The system fully satisfies the requirements and the need
- There are no outstanding requirements items preventing completion

6 / 6

SOFTWARE SYSTEM

Architecture Selected

- Architecture selected that address key technical risks
- Criteria for selecting architecture agreed
- Platforms, technologies, languages selected
- Buy, build, reuse decisions made

1 / 6

SOFTWARE SYSTEM

Demonstrable

- Key architecture characteristics demonstrated
- Relevant stakeholders agree architecture is appropriate
- Critical interface and system configurations exercised

2 / 6

SOFTWARE SYSTEM

Usable

- System is usable and has desired quality characteristics
- System can be operated by users
- Functionality and performance have been tested and accepted
- Defect levels acceptable
- Release content known

3 / 6

SOFTWARE SYSTEM

Ready

- User documentation available
- Stakeholder representatives accept system
- Stakeholder representatives want to make system operational

4 / 6

SOFTWARE SYSTEM

Operational

- System in use in operational environment
- System available to intended users
- At least one example of system is fully operational
- System supported to agreed service levels

5 / 6

SOFTWARE SYSTEM

Retired

- System no longer supported
- Updates to system will no longer be produced
- System has been replaced or discontinued.

6 / 6

WAY OF WORKING

Principles Established

- Principles and constraints established
- Principles and constraints committed to
- Practices and tools agreed to
- Context team operates in understood

1 / 6

WAY OF WORKING

Foundation Established

- Key practices and tools ready
- Gaps that exist between practices and tools analyzed and understood
- Capability gaps analyzed and understood
- Selected practices, and tools integrated

2 / 6

WAY OF WORKING

In Use

- Some members of the team are using the way of working
- Use of practices and tools regularly inspected
- Practices and tools being adapted and supported by team
- Procedures in place to handle feedback

3 / 6

WAY OF WORKING

In Place

- All members of the team are using the way of working
- All members have access to practices and tools to do their work
- Whole team involved in inspection and adaptation of way of working

4 / 6

WAY OF WORKING

Working Well

- Way of working is working well for team
- Team members are making progress as planned
- Team naturally applies practices without thinking about them
- Tools naturally support way of working

5 / 6

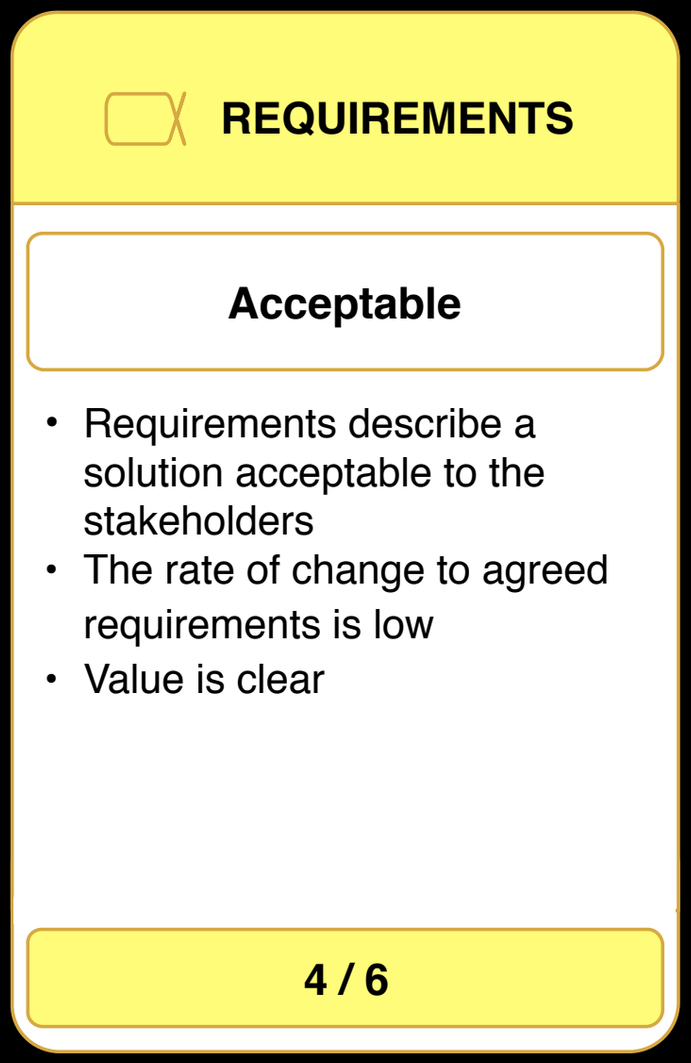
WAY OF WORKING

Retired

- Way of working no longer in use by team
- Lessons learned are shared for future use

6 / 6

Determining the current state (cont'd)

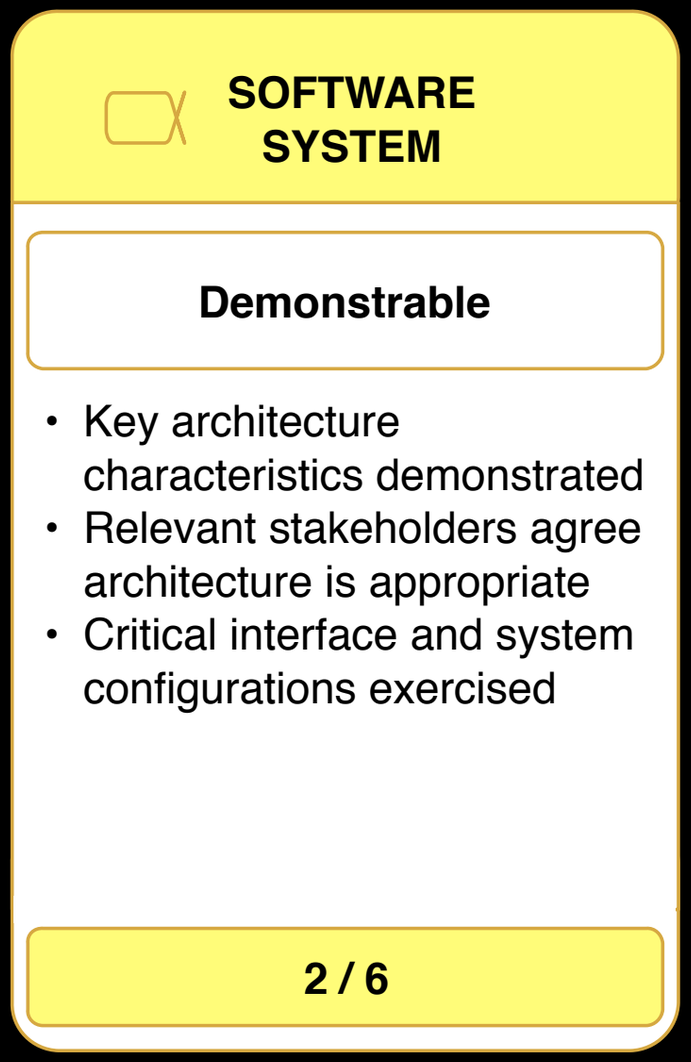


Smith's team had demonstrated an early version of the application based on an initial set of requirements. After the demonstration, the stakeholders agreed that the understanding of the requirements was acceptable.

The agreed-on requirement items were online and offline browsing of the social network, and making posts offline. However, these requirement items were only partially implemented at the time of demonstration.

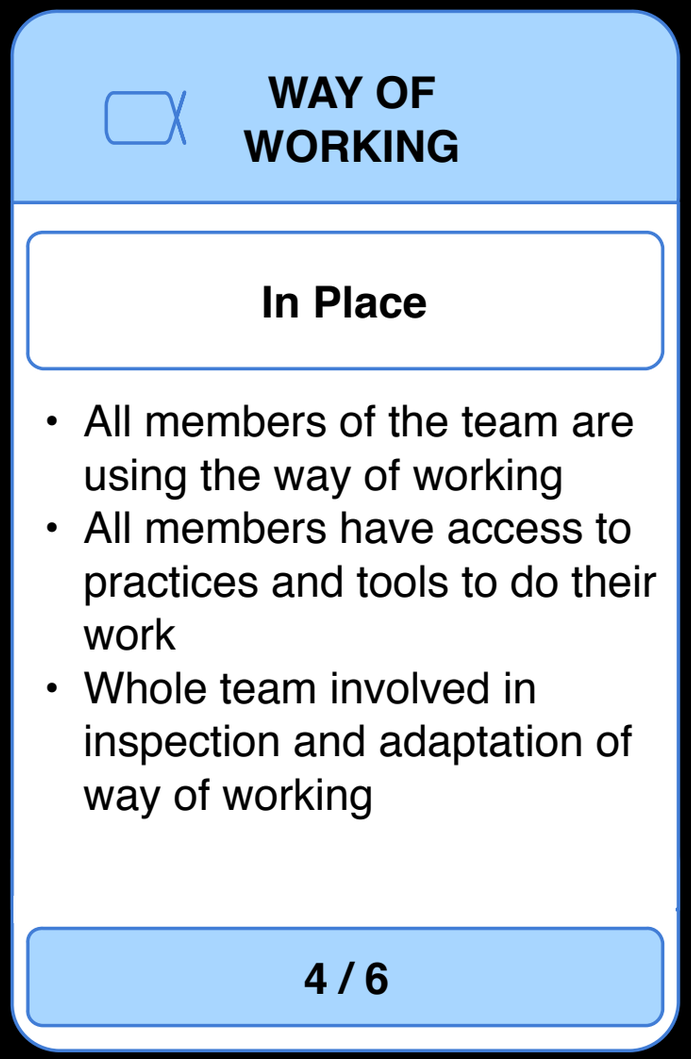
According to the state definition, our team has achieved the Requirements: Acceptable state.

Determining the current state (cont'd)



Early during development, Smith's team had identified the critical technical issues for the software system and outlined the architecture. This had allowed the to achieve the Software System: Architecture Selected state. Moreover, Smith's team had demonstrated an early version of the system to their stakeholders. This means that Smith's team had achieved the Software System: Demonstrable state. However, since Smith's team had not completed enough functionality to allow users to employ the system on their own, Smiths's team had not yet achieved the Software System: Usable state.

Determining the current state (cont'd)

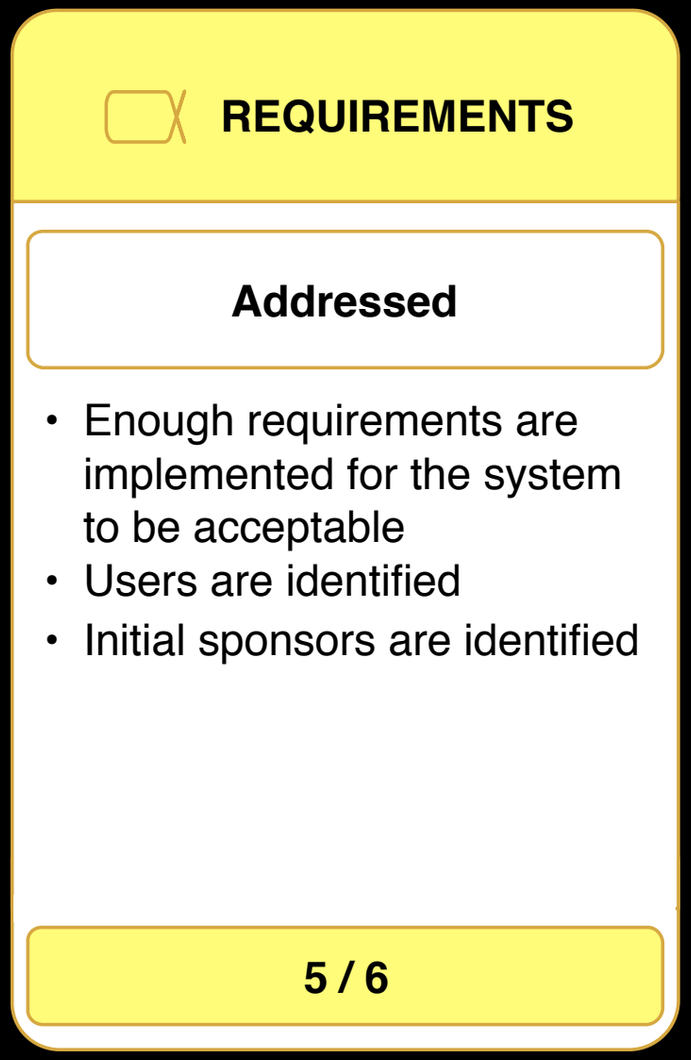


The two new members, Dick and Harriet, who had just come on board were not fully productive yet. In particular, they seemed to have trouble with the approach to automated testing, which the team agreed was important to maintain high quality during development. They had difficulty identifying good test cases and writing good test code. As such, the team agreed that the Way of Working is currently in the In Place state. But they had not yet achieved the Working Well state.

Determining the next state

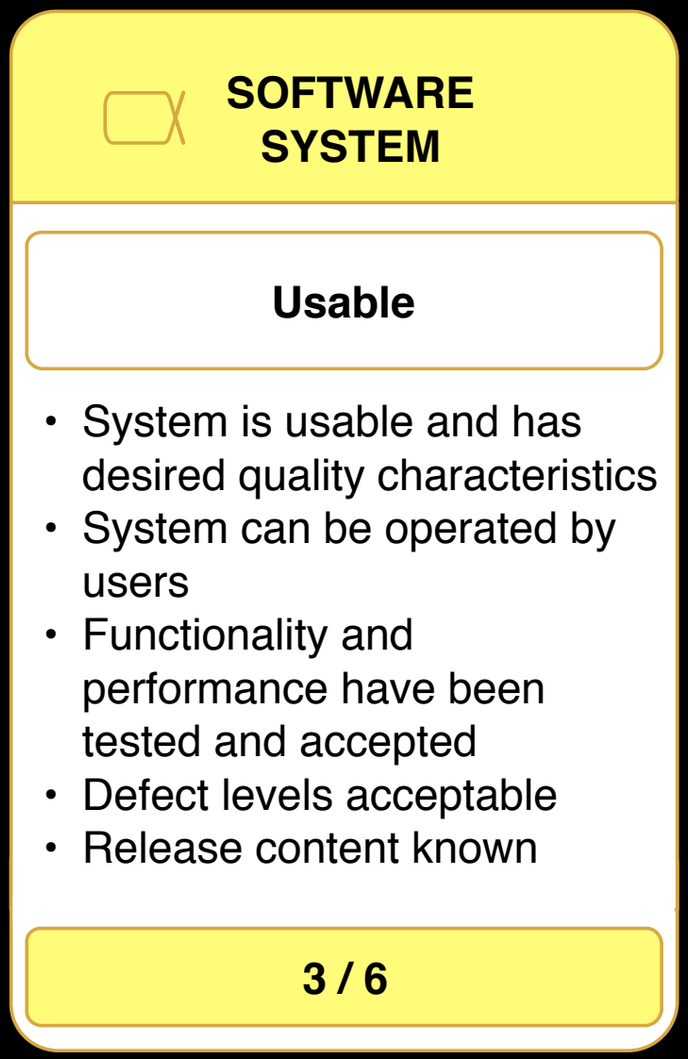
REQUIREMENTS	SOFTWARE SYSTEM	WAY OF WORKING
Addressed	**Usable**	**Working Well**
• Enough requirements are implemented for the system to be acceptable • Users are identified • Initial sponsors are identified	• System is usable and has desired quality characteristics • System can be operated by users • Functionality and performance have been tested and accepted • Defect levels acceptable • Release content known	• Way of working is working well for team • Team members are making progress as planned • Team naturally applies practices without thinking about them • Tools naturally support way of working
5 / 6	**3 / 6**	**5 / 6**

Determining how to achieve the next states



This state reminds us of the need to work with stakeholders to ensure that they are happy the system produced. In our story Smith had to work with Angela to determine which additional requirement items needed to be implemented. This resulted in the following additional task: "Talk to Angela and agree on additional requirement items, fitting in the iteration, to make the system worth being operational".

Determining how to achieve the next states (cont'd)

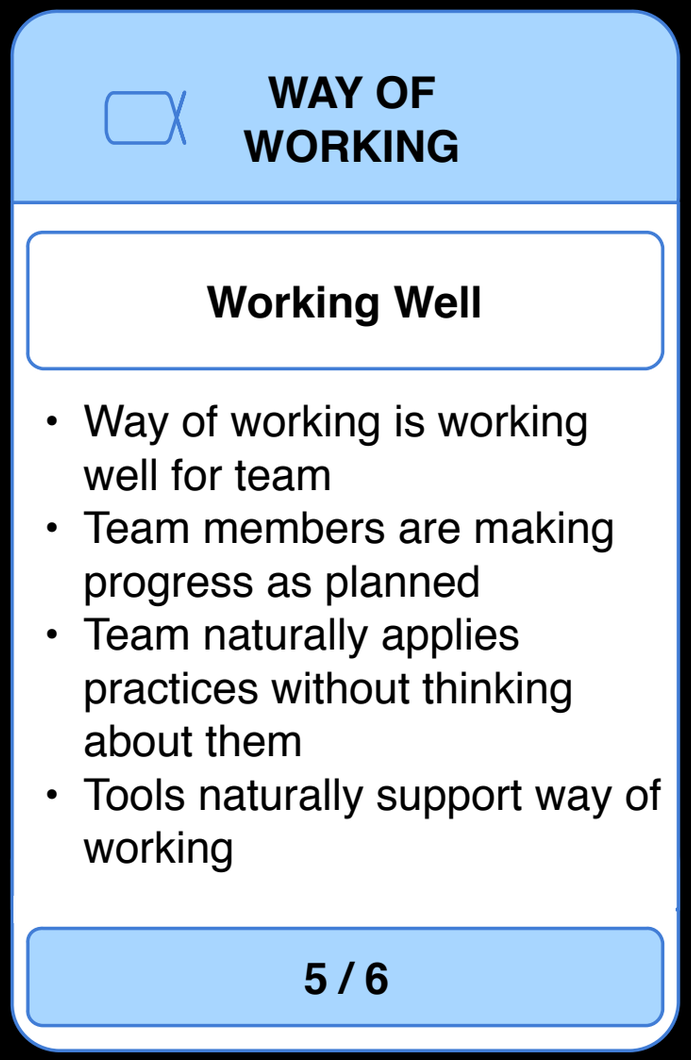


This state reminds us that the software system must be shown to be of sufficient quality and functionality to be useful to the users. So far, Smith's team had been testing within its development environment. Now it had to conduct tests within an acceptance test environment, which they had yet to prepare. This resulted in the following task: Task 2. Prepare test environment.

Smith's team had to bring all requirement items currently demonstrable in the system to completion. By "complete" they meant that each requirement item must be fully tested within the acceptance test environment.

- Task 3. Complete requirement item A: "Browse online and offline".
- Task 4. Complete requirement item B: "Post comment (online and offline)".
- Task 5. Complete requirement item C: "Browse album".

Determining how to achieve the next states (cont'd)



Both Dick and Harriet agreed that they had difficulties in applying automated testing.

They needed help in order to make progress. Tom agreed that he had to spend time teaching them.

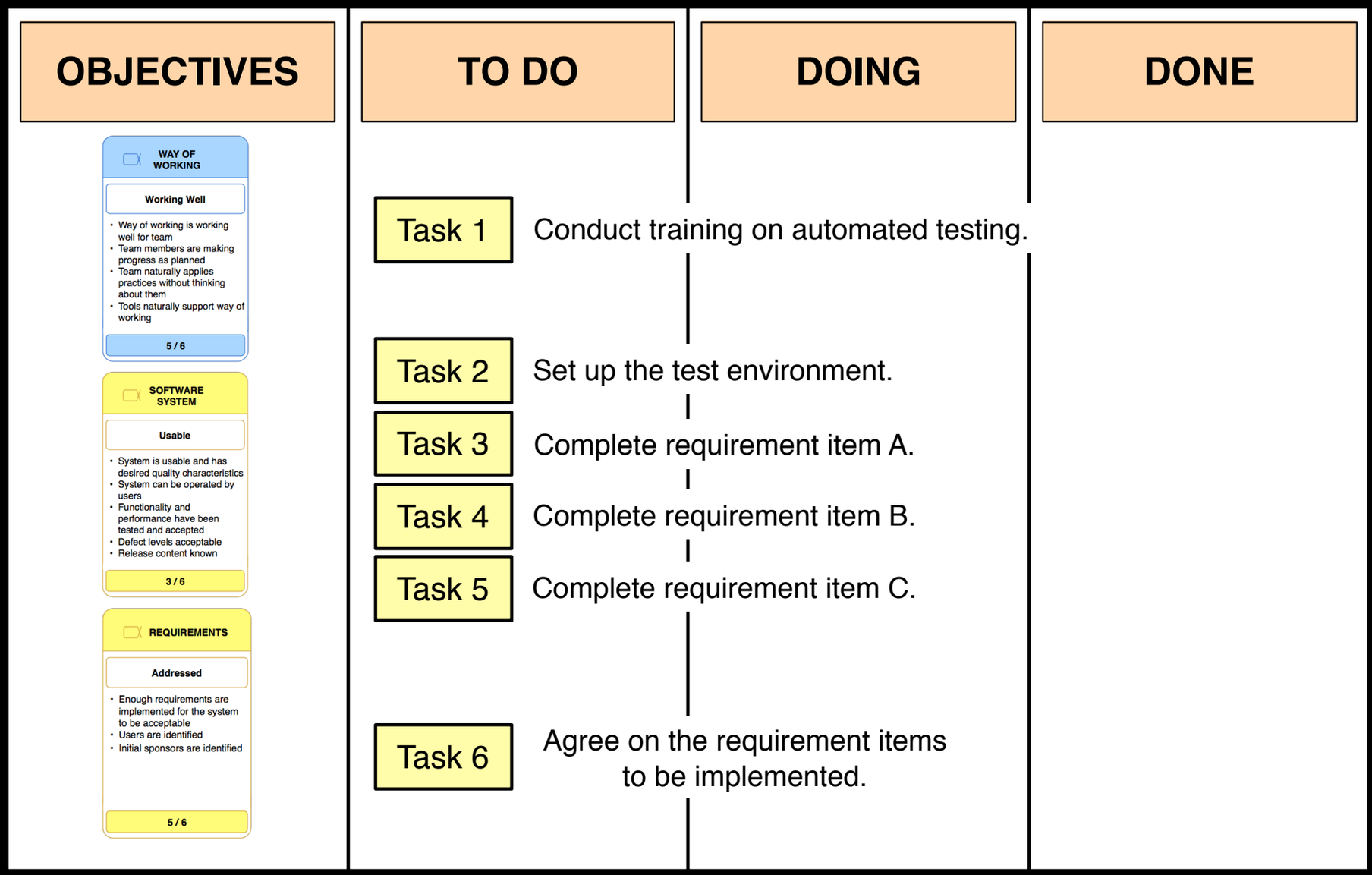
A task was added to the iteration backlog for Tom to conduct training on automated testing for Dick and Harriet.

Why is the kernel useful in planning iterations?

- **Inclusive,** the kernel alphas serve as reminders across the different dimensions and helps create a plan

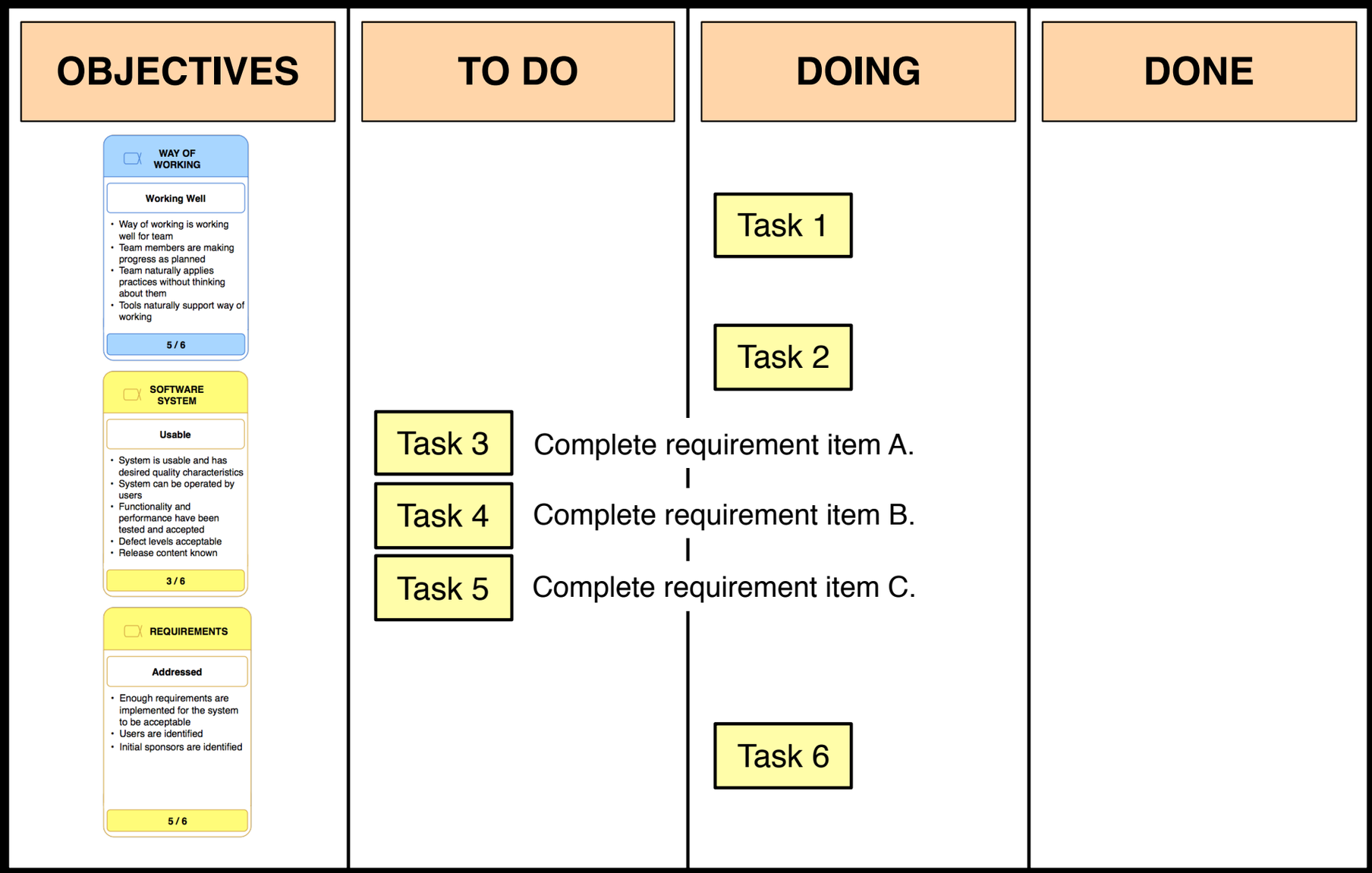
- **Concrete,** the checklists for each alpha state give you hints as to what you need to do in the iteration and to determine progress

Doing the iteration



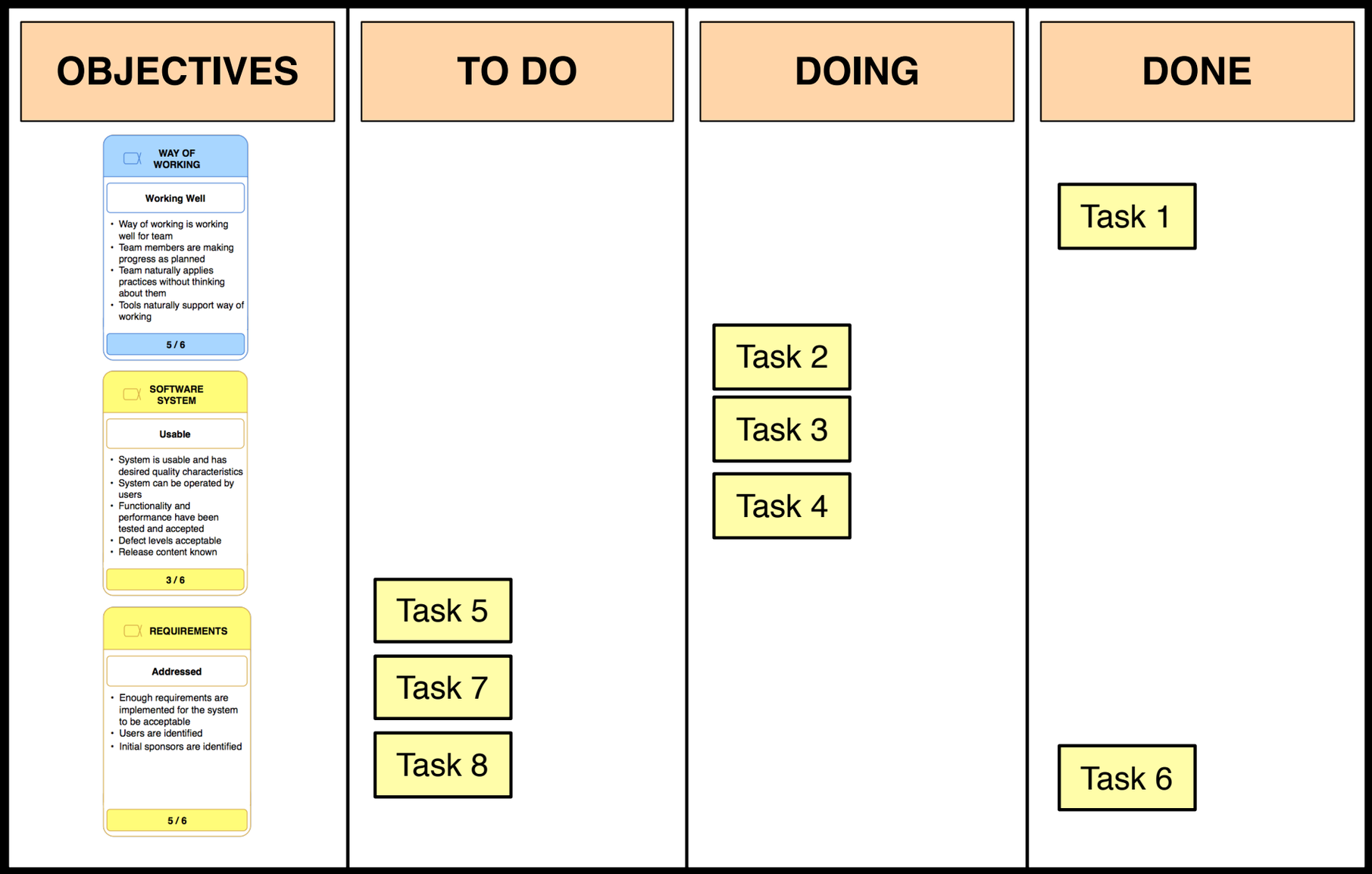
The task board at the beginning of the iteration

Doing the iteration (cont'd)



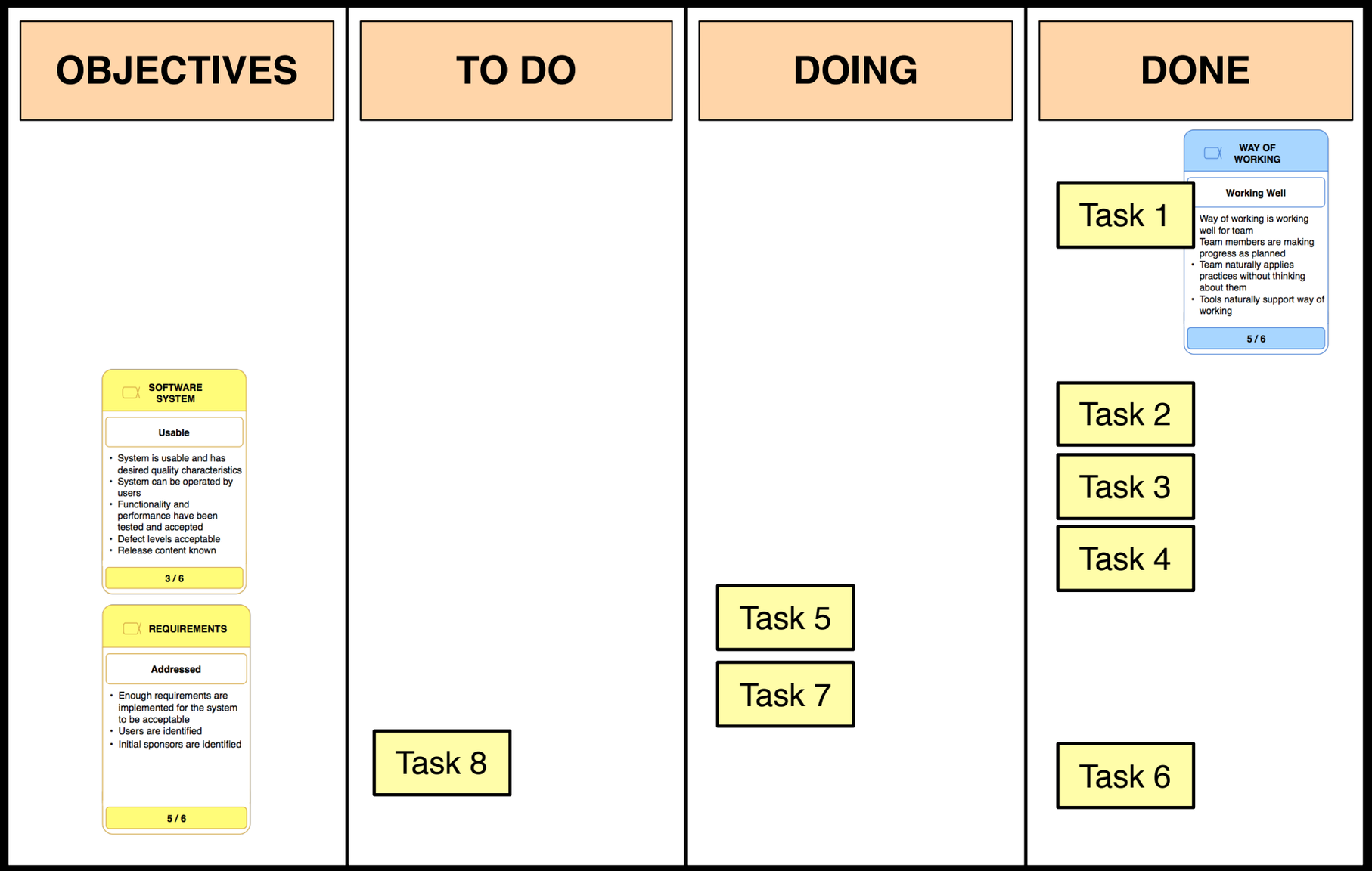
The task board on day 1 of the iteration

Doing the iteration (cont'd)



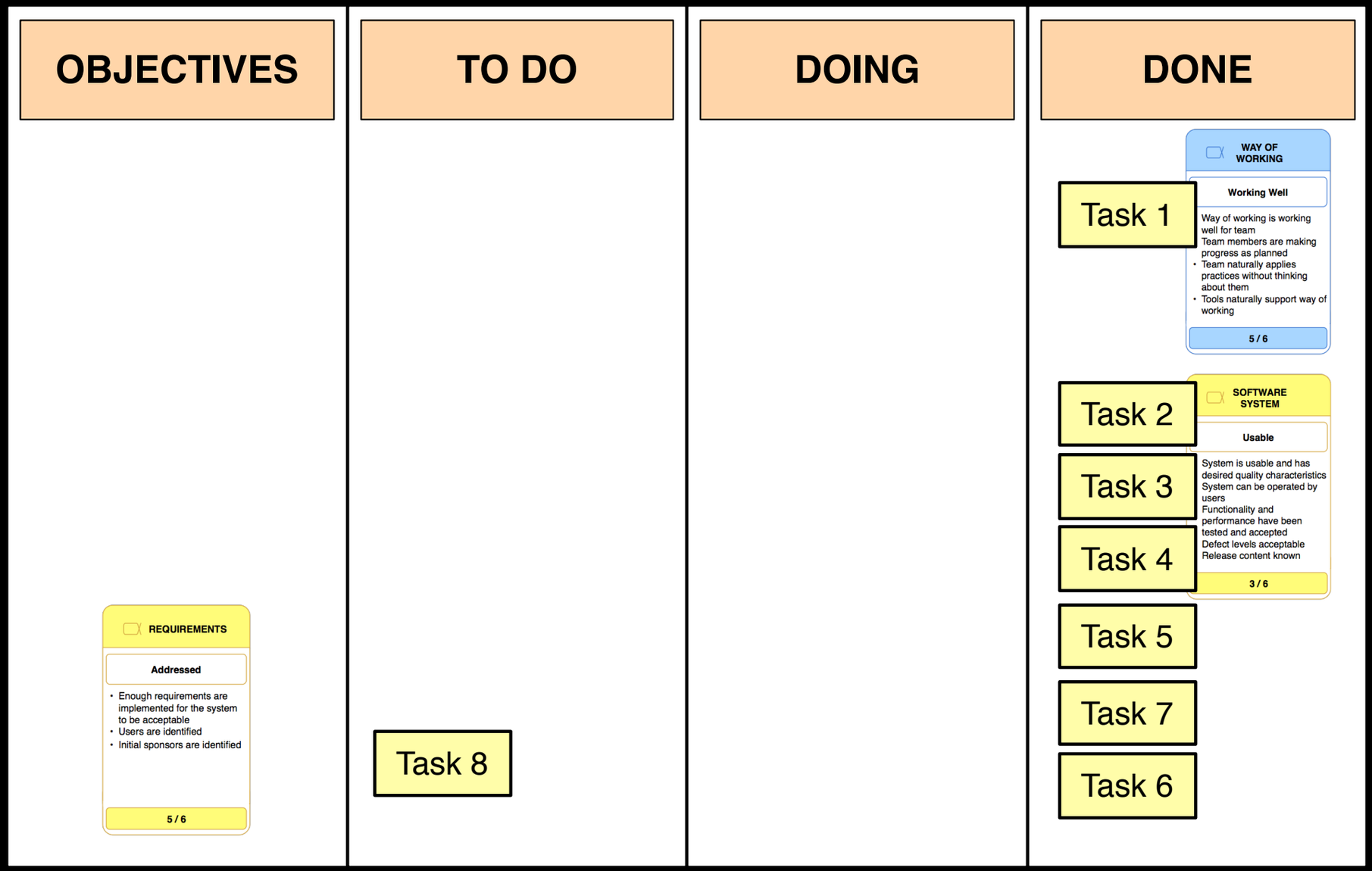
The task board on day 2 of the iteration

Doing the iteration (cont'd)



The task board on day 4 of the iteration

Doing the iteration (cont'd)



The task board on day 6 of the iteration

Why is the kernel useful in doing/checking iterations?

- **Iteration objectives,** the alpha state checklists help you agree on what needs to be done

- **Task prioritization,** alpha states' checklists help determine if new work is something you should prioritize or postpone

- **Reminder for missing tasks,** do we have all the tasks we should be working on now?

Adapting the way of working

- What went well?
- What did not go well
- What can be done better?

Reflection

Dick, "Our application does what it needs to,
but the user experience is not that good.
Downloading is a little slow."

Reflection (cont'd)

Dick, "Our application does what it needs to,
but the user experience is not that good.
Downloading is a little slow."

Reflection should not focus on the product
but rather the way of working

Adapting the way of working (example)

- What went well with our planning, doing, and checking related to the previous alpha states?

- What did not go well with our planning, doing, and checking related to the previous alpha states?

- What can we do better with our planning, doing, and checking related to the previous alpha states?

Reflection (example)

Harriet, "Actually, Smith, for me to do my job better I would like to have better guidance regarding how to work on a requirement item."

Will require a change to the way of working with requirements!

Week 2

- **Monday**: SEMAT
- **Wednesday**: SEMAT continued
- **Sunday**: Submit app vision