

Where we are going!

Ivar Jacobson



The Big Picture

This talk is about the kernel and the kernel language.

It draws on my personal experience.

It suggests a goal we need to find.

It shows it can be found.

Reaching the goal, it will have dramatic impact on the whole software community

- the industry,
- the developers,
- the academics,
- the education,
- the methodologists, etc.

Watts Humphrey:

“This meeting in Zurich is likely to be an historic occasion much like the 1968 NATO session in Garmish.”

Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

What went well and what went wrong

The *perceived* “rise and fall” of RUP

Let's be clear, the “rise and fall” are all about perception

“Good”

- Many proven practices
 - Use-cases (incl test)
 - Iterations
 - Components
 - Architecture
 - Etc.
- Supported UML
 - UML replaced all the hundred modeling languages at the time

“Bad”

- A soup of practices
- Too big
 - People don't read process books
- Hard to extend with agile, CMMI, etc.
- Adoption extremely hard
 - Process savvy
 - Revolutionary
- Gap between what people said they did and what they really did – The Process Gap

On Processes (or Methods and Methodologies)

Some exaggeration <grin>

- Every process tries to be complete
 - As a consequence every successful process will grow until it dies under its own weight
- Every branded process is just a soup of ideas "borrowed" from other processes
 - With some new idea(s)
- Every process usually becomes just shelf-ware
 - Law of Nature: People don't read process books
- The process is out of sync with what the team does...
 - ...and the project – process gap get wider and wider
- The project has to adopt an entire process
 - No-one uses an entire process or limits themselves to practices from one process

No wonder people don't like process

We looked for fundamental changes.

“Bad”

- A soup of practices
- Too big
 - People don't read process books
- Hard to extend with agile, CMMI, etc.
- Adoption extremely hard
 - Process savvy
 - Revolutionary, not evolutionary
- Gap between what people said they did and what they really did – The Process Gap

Fixing what was “Bad”

- Make **practices** first class citizens, and process a composition of practices
- Focus on the **essentials** instead of trying to be complete
- Extensions through practices
- **A new user experience** with focus on developers, not on process engineers.
- **Enact the process**

We redesigned RUP as EssUP

Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

Practices

In the future, an ever present but
invisible process

Process becomes second
nature

The team's way-of-working is
just a composition of
Practices

We need a new
paradigm

Practice is a First Class Citizen

the unit of adoption, planning and execution of process

From the successes
in modern software
development

The Software
Engineering
Camp

Process
Maturity
Camp

Agile
Methods
Camp

Examples:

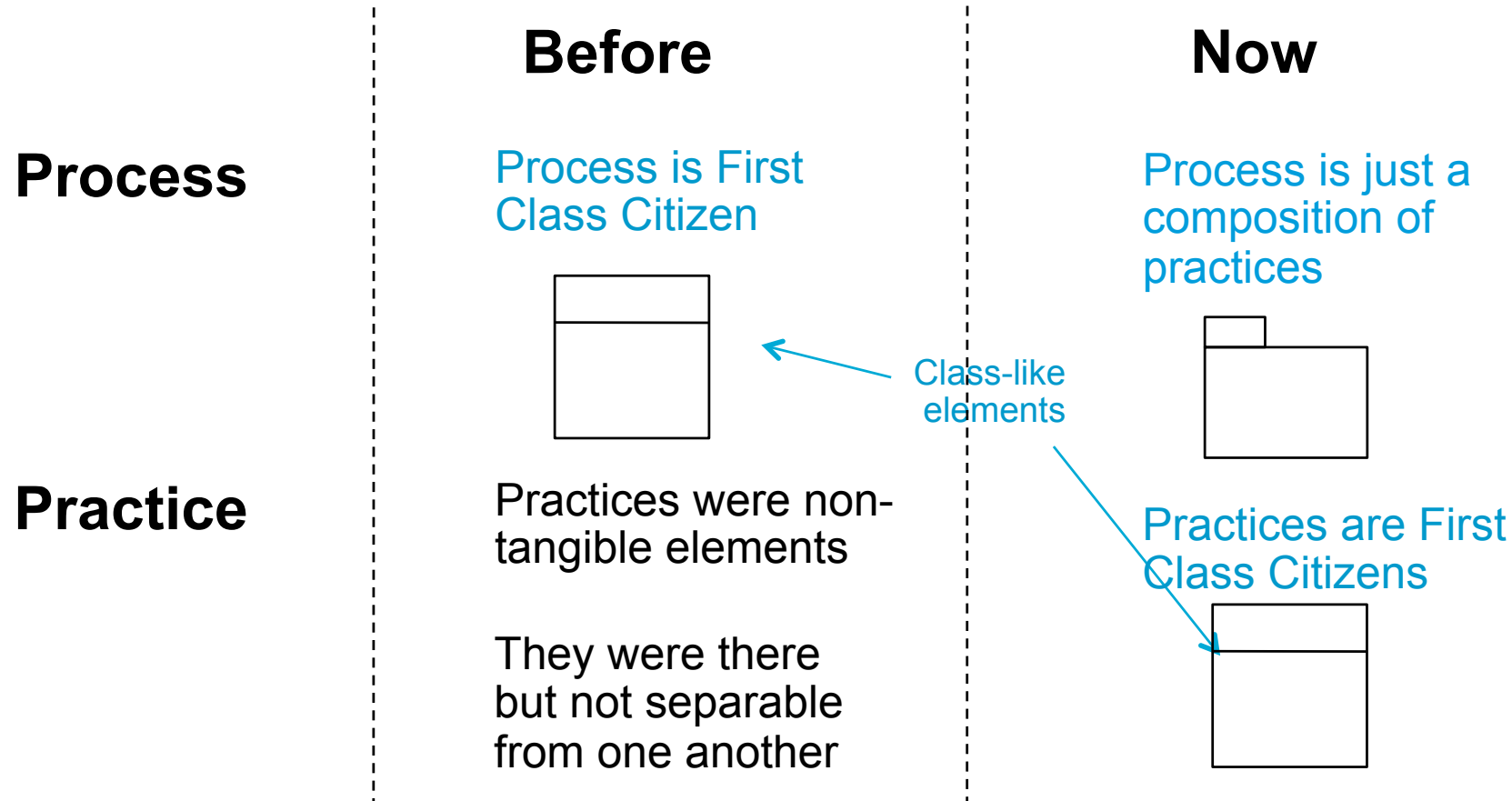
Unified Process

CMMI, Spice

XP, Scrum

The Paradigm Shift: From 'Processes' to 'Practices'

We have always had practices in a loose meaning



- After the paradigm shift you can do all kinds of operations on practices
 - Separate them, compose them, teach them, execute them

We needed a shared definition of “practice”

Pragmatics

- A practice provides a way to **systematically** address a particular aspect of a process. *It is a separate concern of the process.*
- There are three kinds of practices (at the least):
 - **Peer** practices
 - A practice has *a clear beginning and an end* allowing it to be separately applied, examples:
 - Iterative development
 - Use-case driven development
 - Project management à la Scrum
 - **Extension** practices
 - Use cases for SOA
 - **Cross-cutting** practices
 - Team practice incl workshops, self-organizing teams, war room, pair programming, etc.
 - Process improvement for the essentials of CMMI – e.g. metrics.

A Good Practice is good for the team

- Gives a result of observable **value** to the **customer** of the team
 - It is a building block for the team – not necessarily for the process engineers.
 - Not too big – not too small
 - It includes its own **verification**
 - It is that thing that needs to be made **lean**
 - It is that thing for which you want to have **metrics**

Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

Focus on the Essentials

What is Essential?

- It is the key things to do and the key things to produce
- It is about what is important about these things
- It is less than a few percent of what experts know about these things
 - Law of nature: People don't read process books
- It is the placeholders for conversations
 - Law of nature: People figure out the rest themselves
 - Training helps
- It is the base for extensions

Starting with the essentials makes a practice adoptable.

How much do you need in your hands?

Find Actors and Use Cases

Specify the System

Opportunity | Specified System | Backlog

Find actors and use cases to:

- Agree on specified system behavior
- Establish the system boundary
- Scope the system
- Agree on the value the system provides or beyond
- Identify ways of using & testing system

The activity is completed when:

- The Use-Case Model: **Value Established** or beyond
- Use Case Specifications: **Briefly Described** or beyond
- Supplementary Requirements: **Initiated**

The activity contributes to achieving:

- Specified System: **Shared**
- Use-Case Module: **Scoped**

Recommended approaches:

- Use-case modeling workshop
- Structure the use-case model
- Handle changes (to the use-case model)

Essential Unified Process 3.1 | © Ivar Jacobson International, 2005-2007 | Use Case Essentials

Find Actors and Use Cases

Introduction

To effectively shape and scope means get the emerging model someone with the **RIGHT ADVICE** competency level should be involved. The analysis team ensures that there is sufficient C...

Find Actors and Use Cases

Approach

- Generate requirements as abstract use case database

Find Actors and Use Cases

Common mistakes

Doing it by yourself

A common mistake is that one person creates the use-case model on their own. Problem: This results in poor quality, no consensus and limited understanding. Evidently: Make sure that the right people (users, domain experts and other stakeholders) are involved and actively contributing to the creation of the use-case model.

Thinking you know best

A common problem is that the members of the development sub-team think they know what is best for the customer and users. Problem: This is likely to result in limited consensus and reduced team morale. Evidently: Make sure that the members facilitate the creation of the use-case model rather than dictating its content to the others involved.

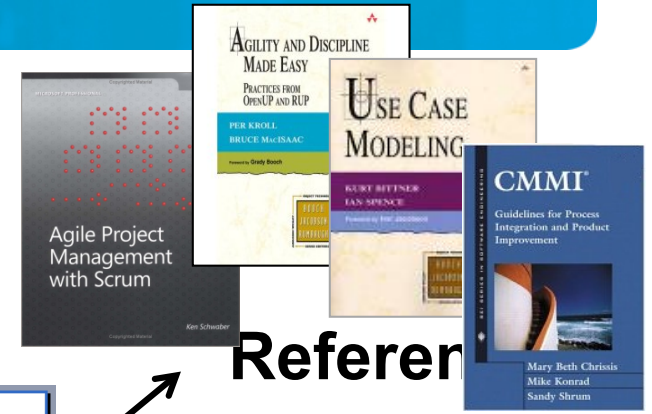
Over-structuring the model

Sometimes the team get carried away with the possibilities of structuring the use-case model and over-use the use-case relationships. Problem: Too much structuring of the model or doing it too early, can make the model difficult to understand and use. Evidently: Don't use the use-case relationships unless absolutely necessary.

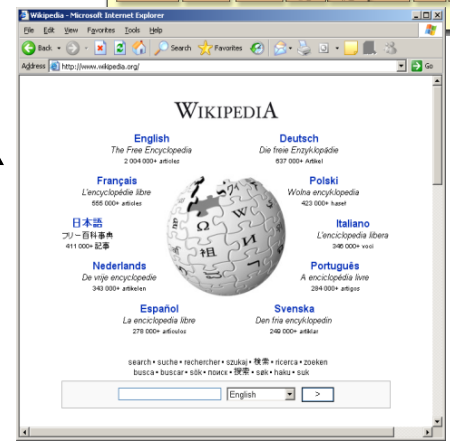
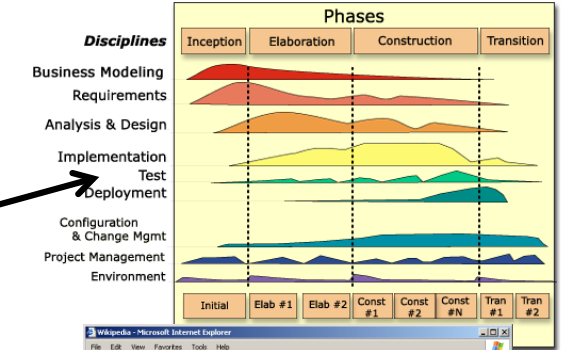
Functional decomposition

Use cases are not functions. A use-case model is based on identifying the value offered by the system and describing the external interactions required to achieve that value. It is not the decomposition of the requirements into a series of reusable function definitions.

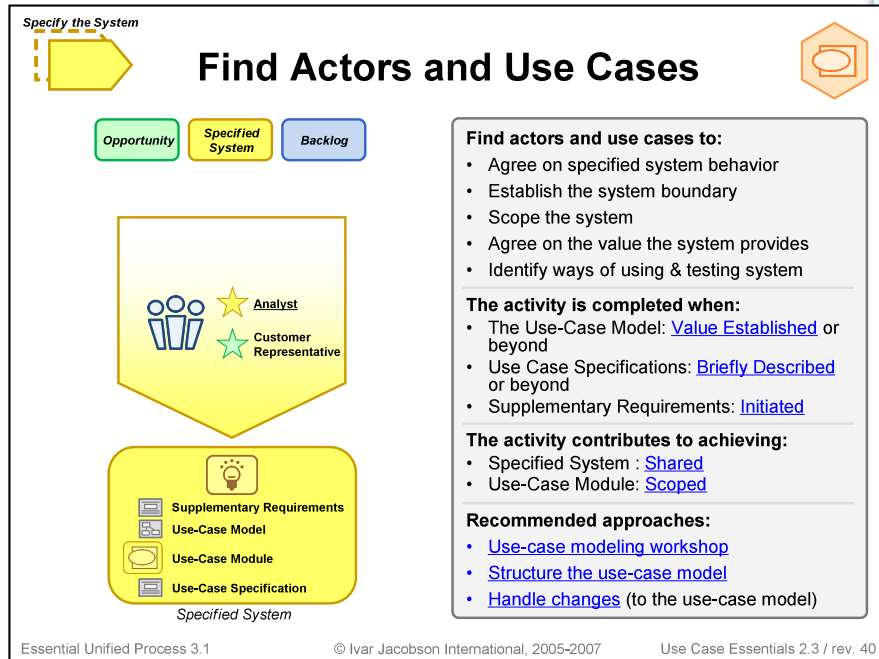
The Essential Unified Process | www.sei.com | © Ivar Jacobson International, 2007 | Page 3 of 3



Reference

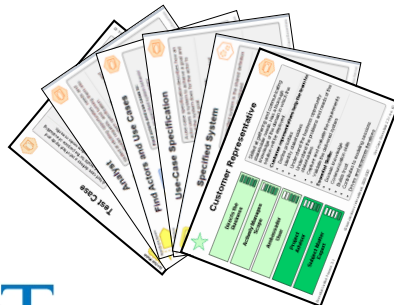


Why Cards?

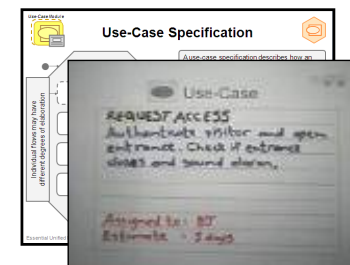


- Cards are tactile
- Cards are simple and visual
- Cards use conversational and personalized style
- Cards are not prescriptive so they get the learner to think more deeply
- Cards get...and keep...the readers attention
- Cards promote agility
- They can be written on to make minor adjustments to the practice on the fly

- A practice is a set of cards



- A team works on a set of instance cards



Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

Practices are enacted

Use Case Module Game Board						
Alpha Title	States					
	Start	Scoped	Specification Agreed	Realized	Implemented	Verified
Withdraw Money						
Transfer Money						

Set Up Your Goals

Specify Use Case Module
Use Case Essentials

Select and Prioritize Use Cases
Use Case Essentials

Find Actors and Use Cases
Use Case Essentials

Describe the use-case module to:

- Specify the use case
- Detail the needed behavior and define test characteristics (i.e. non-functional) of software towards it.

When: **Specification Agreed**, **Bulleted Outline**, **Chosen** or beyond. **Key Identified**.

This activity is completed when:

- Use-Case Model: **Value Established** or beyond
- Supplementary Requirements: **Initiated**, **Described** or beyond

This activity contributes to:

- Requirements are: **Shared**
- Use-Case Modules: **Scoped**

Recommended approaches:

- Use-case modeling workshop
- Structure the use-case model
- Handle changes to the use-case model

Supplementary Requirements
Use Case Essentials

Use Case Specification
Use Case Essentials

Use Case Model
Use Case Essentials

The purpose of a use-case model is to explain, bound and scope a system by providing a complete picture of its actors and use cases.

The use-case model:

- Allows teams to agree on the required functionality and system characteristics
- Clearly establishes the boundary and scope of a system
- Enables agile requirements management.

Essential Contents

- 1..n Actor
- 1..n Use Case
- 1..N Associations and relationships
- 1 An explanation of the model as a whole

References To

- 1..N Use-Case Modules

Get Help To Reach Your Goals

Things to do

Things to produce



Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result

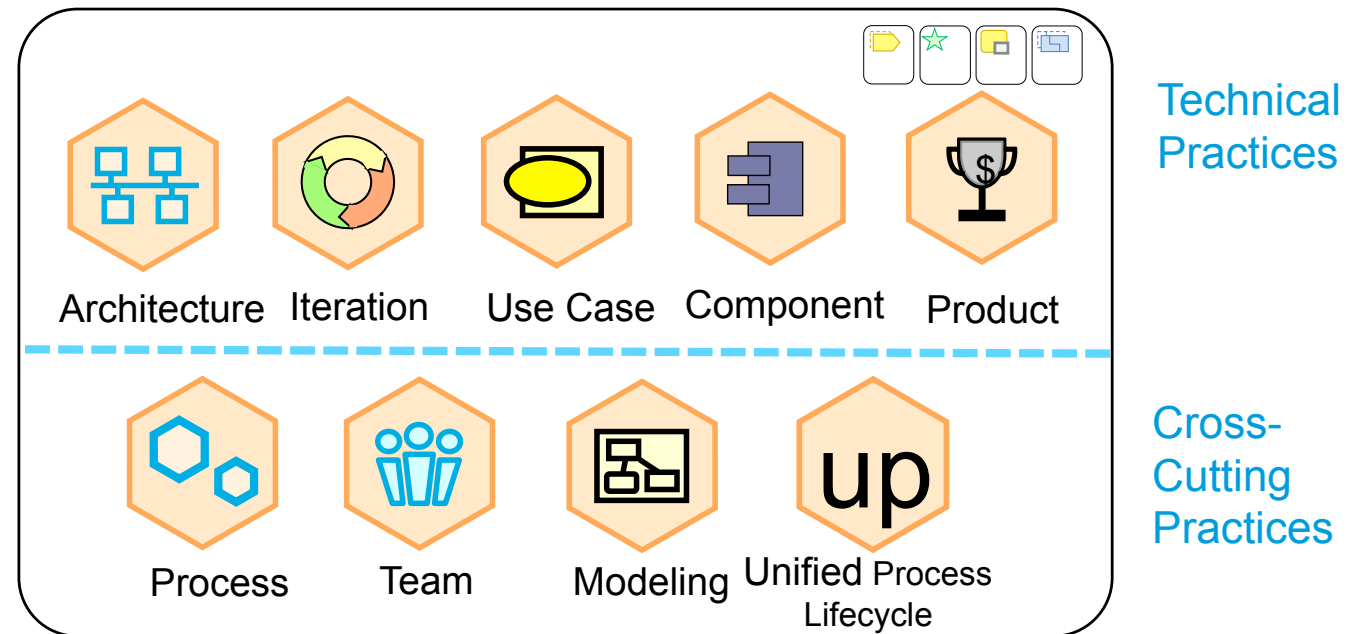
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

Thus we fixed what didn't work

Fixing what was “Bad”

- Make **practices** first class citizens
- Focus on the **essentials**
- Extensions through practices
- **A new user experience** with focus on developers
- **Enact the process** to close the gap

Essential Unified Process



Great, but now more became evident!

Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

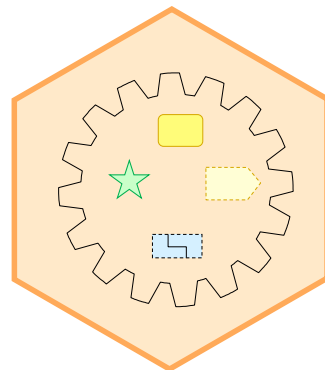
Hypothesis harvested from the fixing-the-problem work

- All methods comprise of a set of things that are always there - documented or not.
- We called this set the Kernel.
- Every method can then be described as a set of composed practices using the kernel.

There is a kernel!
Many different methods can be built out of
this same kernel.

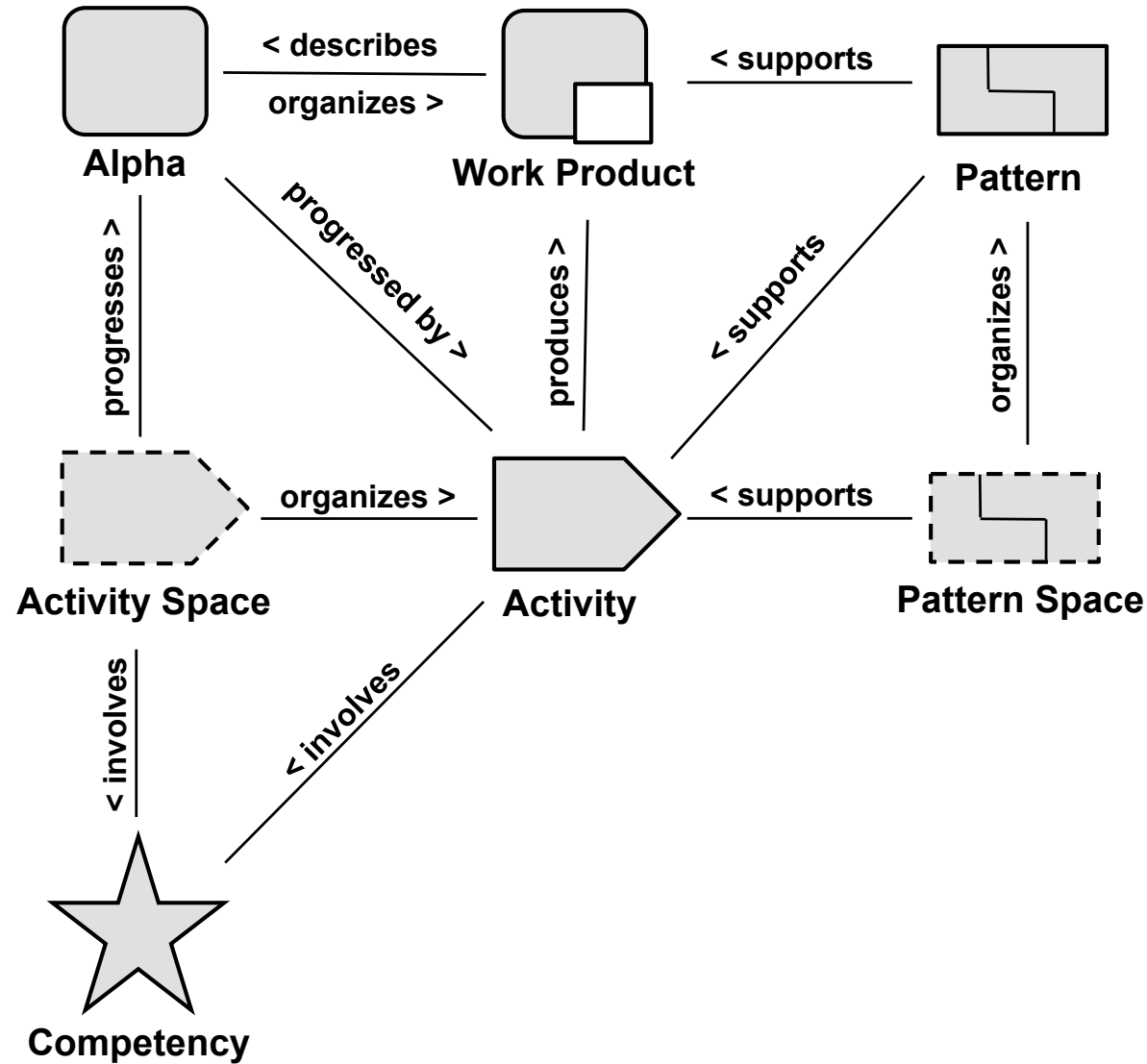
To verify the hypothesis we started all over

- We called our initiative EssWork (moving beyond EssUP)
- The Kernel we harvested is very small, extracted from a large number of methods
- It contains empty slots for things that every process have
 - Slots for
 - Competencies, such as analyst, developer, tester
 - Things to work with, such as backlog, implementation, executable system
 - Things to do, such as implement the system, test the system
- The Kernel is practice and of course method agnostic.



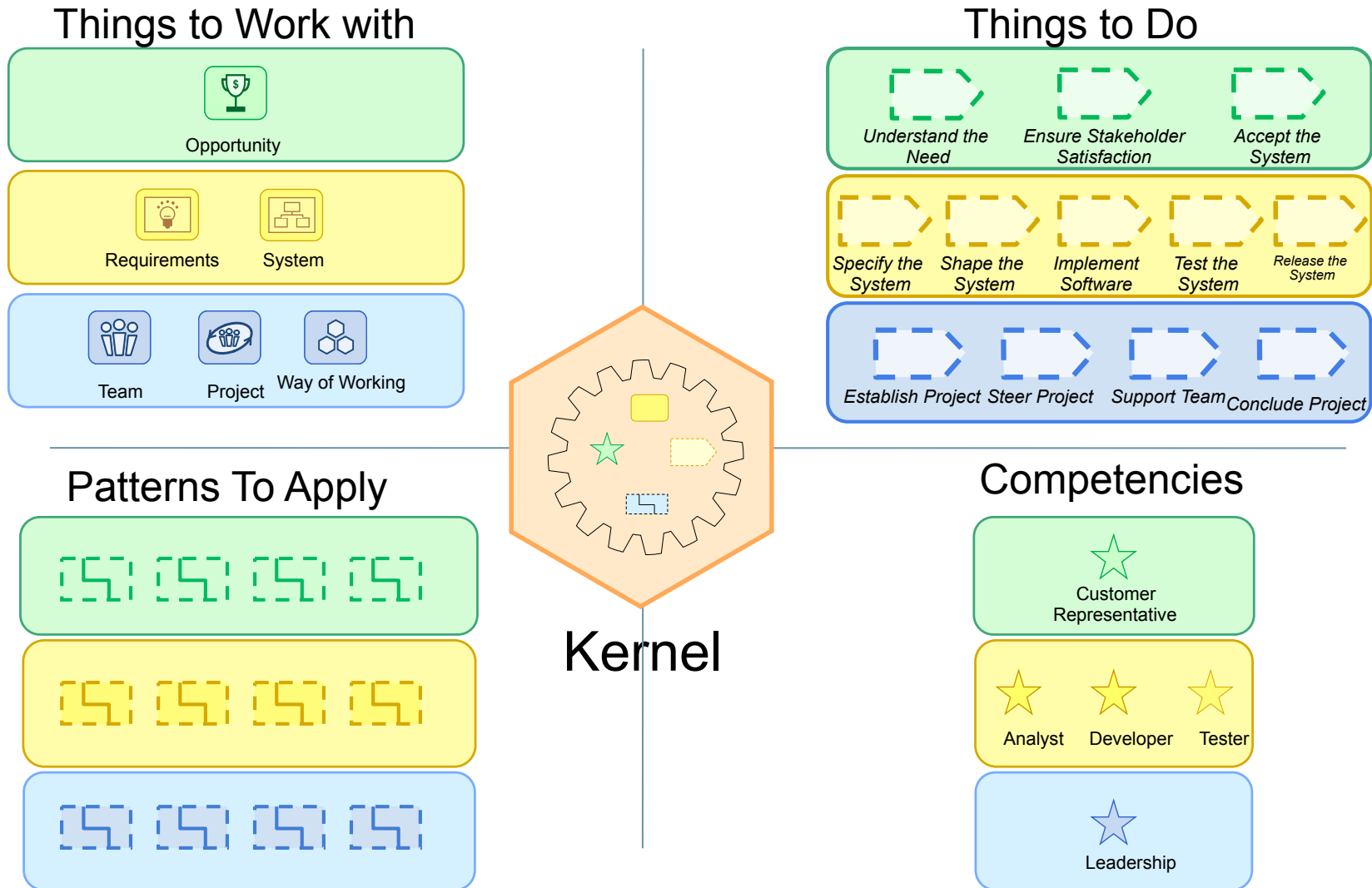
Kernel

The Kernel includes a Meta-Model - an implied language

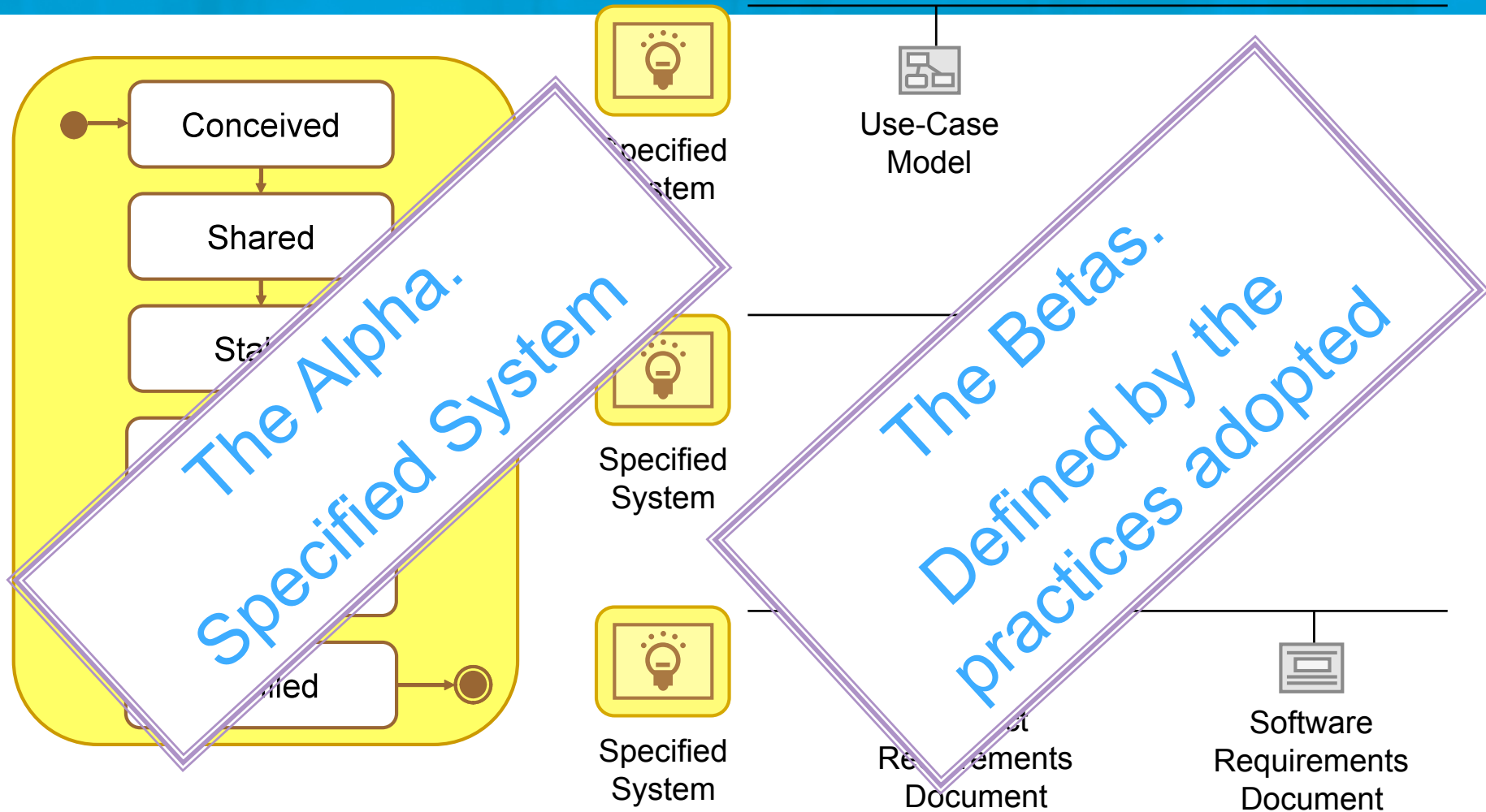


The EssWork Kernel

- contains empty slots for things that every process have



Practices put the meat (*Betas*) on the bones (*Alphas*)



For example there are many ways to specify the system.

Comparing Alphas and Work Products

Alphas:

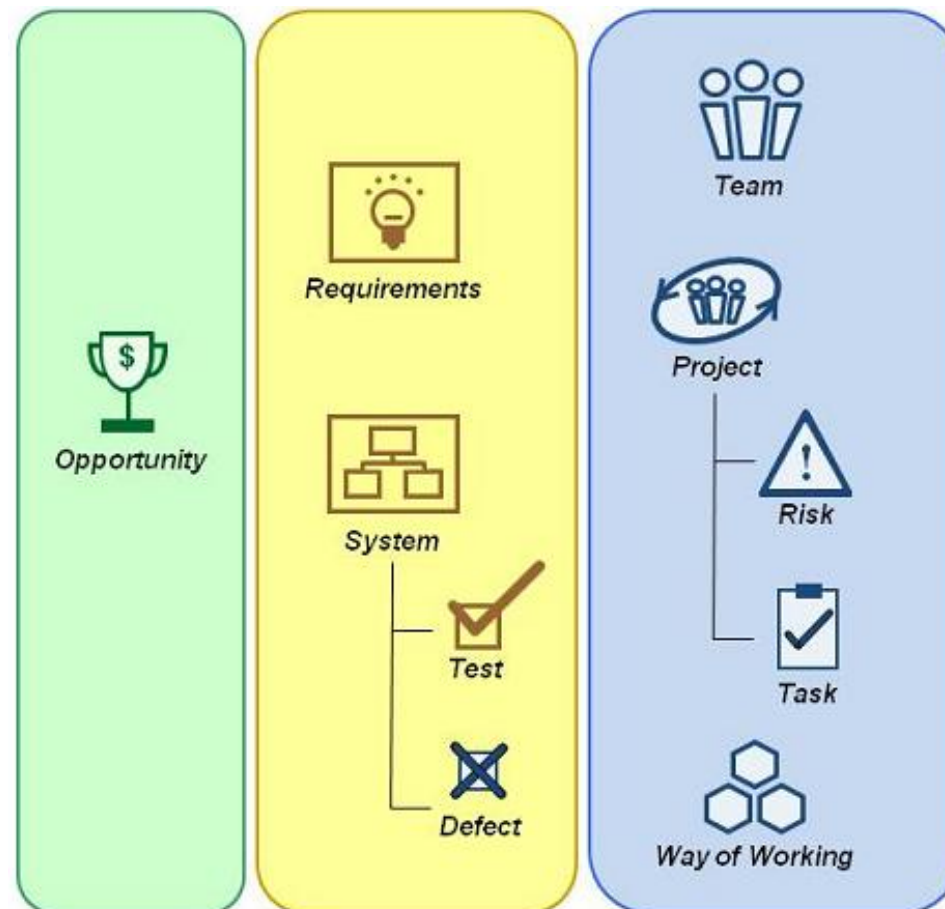
- The most important things that all software projects have whether they exist
- Intangible
- The things whose progress we want to understand, monitor, direct and control
- Alphas have progress states
- State progression means progression towards release

Work products:

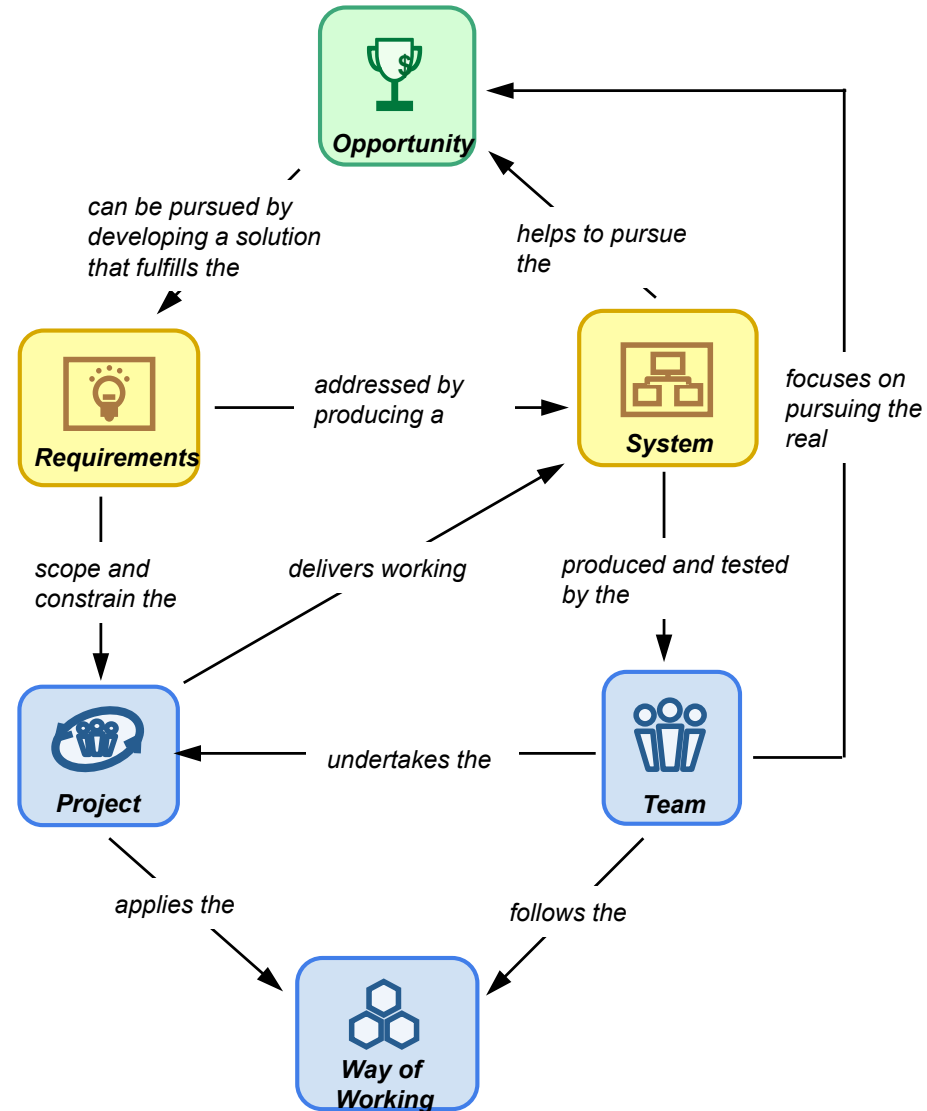
- Used to record information **about** alphas
- Used to understand and assess the alphas
- Can be physical documents, electronic files, models, databases, .xml
- State progression generally represents more information or detail

Things to Work with: Alphas and Work Products

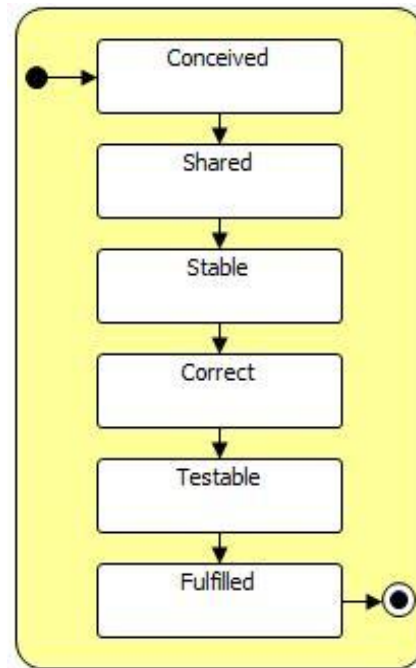
These are the alphas:



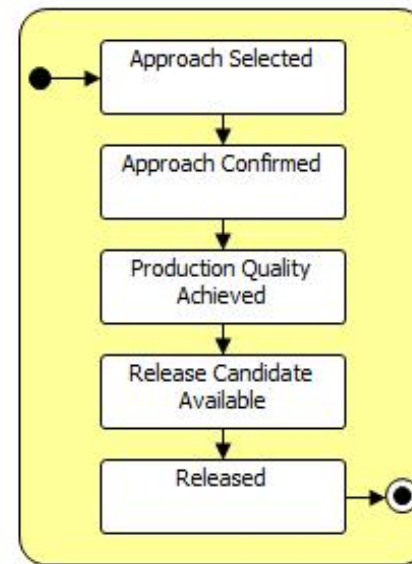
Alpha Relationships



Alpha States

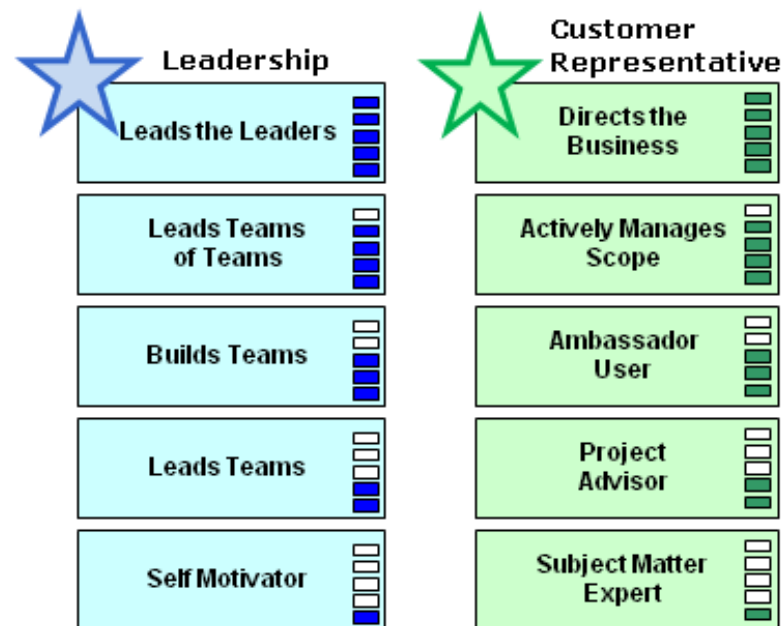
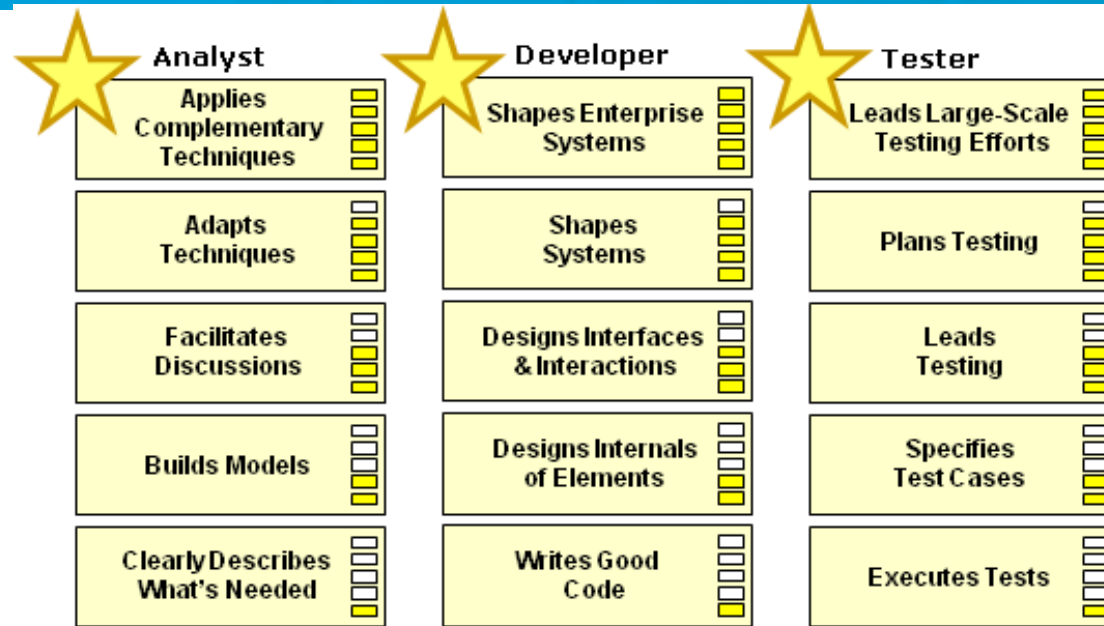


Requirements states

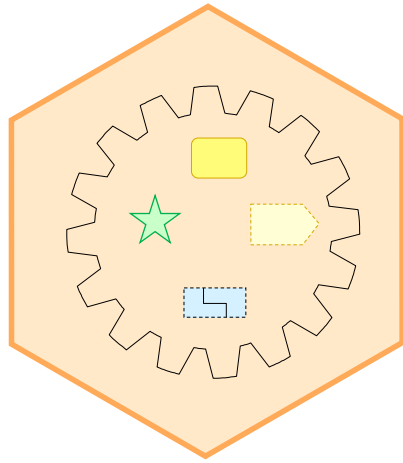


System states

Competency Levels

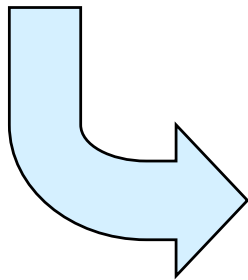


Using the kernel

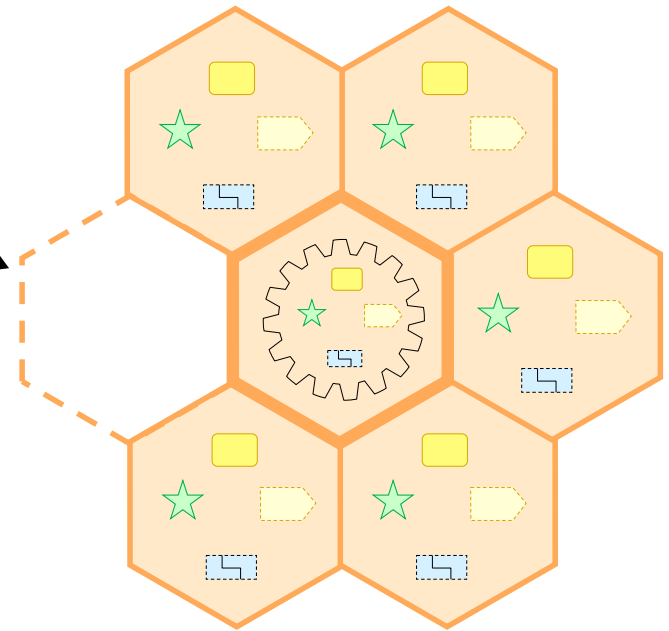
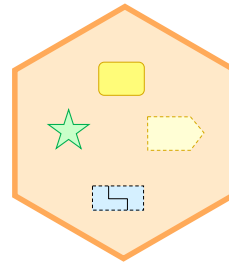


Kernel

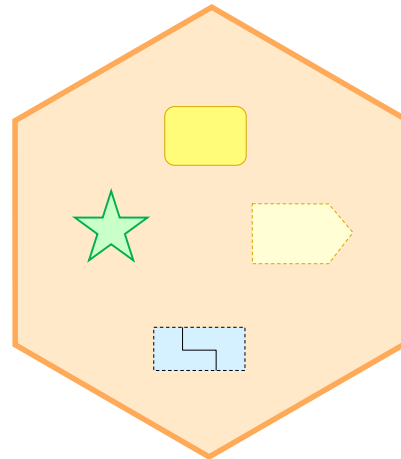
The kernel defines an “empty process”



Practices “slot” into the common kernel.

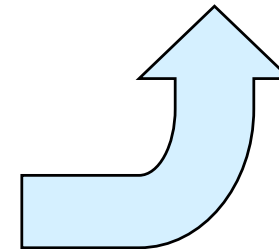


Way of Working

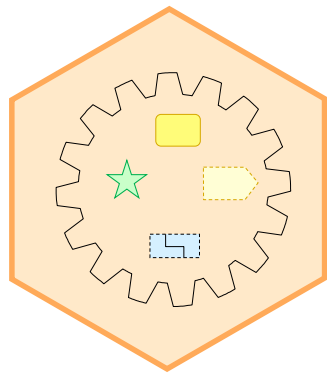


Practice

Each practice contains practice-specifics to add to the kernel.

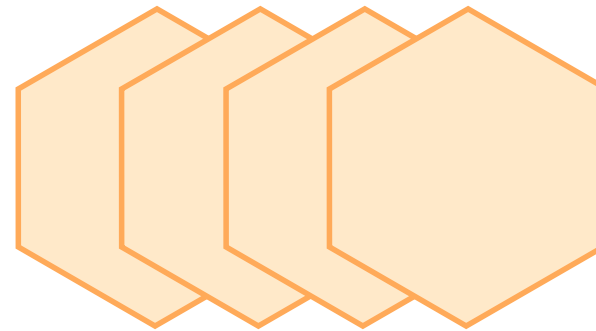


Change starts by harvesting your best practices from your own method



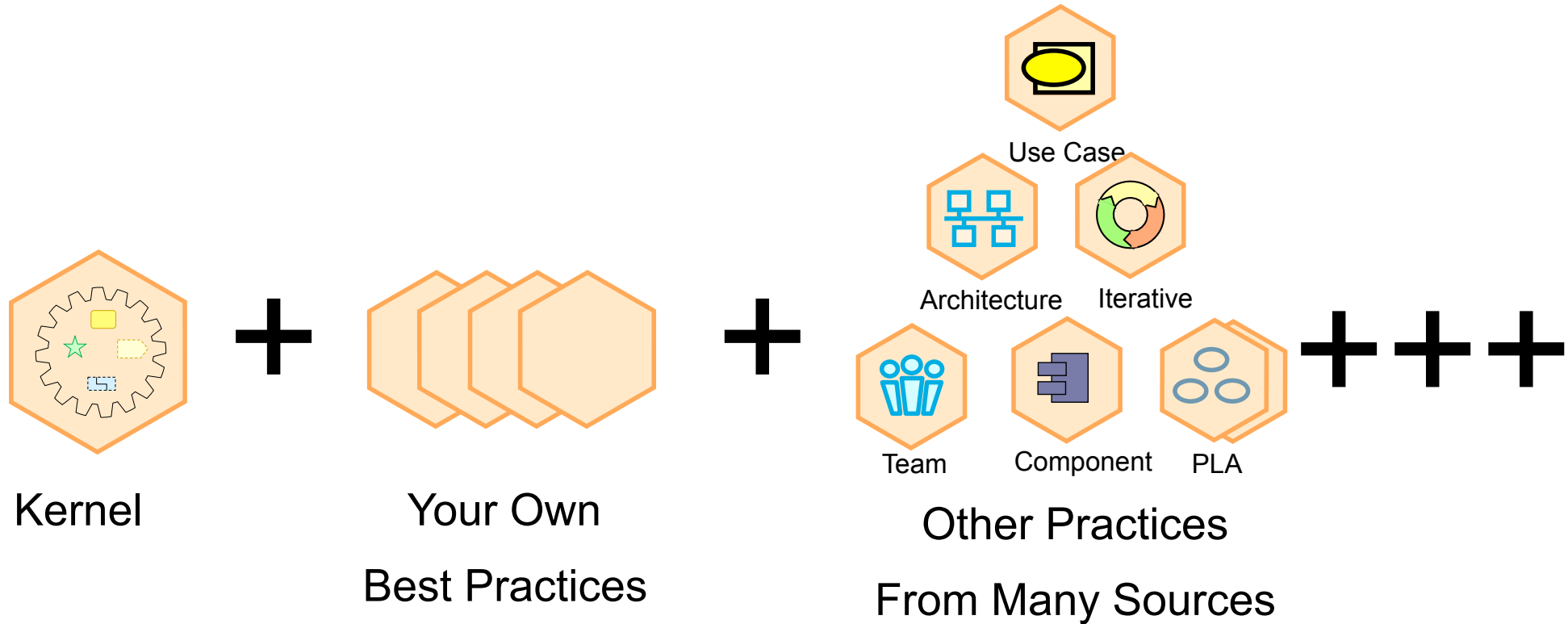
Kernel

+



Your Own
Best Practices

Improve your method by adding other, proven practices



OK, there is a kernel!
Maybe there are many?
But none is widely-accepted!
That needs to be changed!

Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

CASE FOR ACTION 2nd part

- 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
 - Addresses both technology and people issues
 - Are supported by industry, academia, researchers and users
 - Support extension in the face of changing requirements and technology

The Kernel \approx The Kernel Language + The Universals

The Envisioned Kernel

Level

3

Methods

↓ Composed of
⋯ Defined in terms of

2

Practices

Patterns

1

Universals Kernel language

The kernel

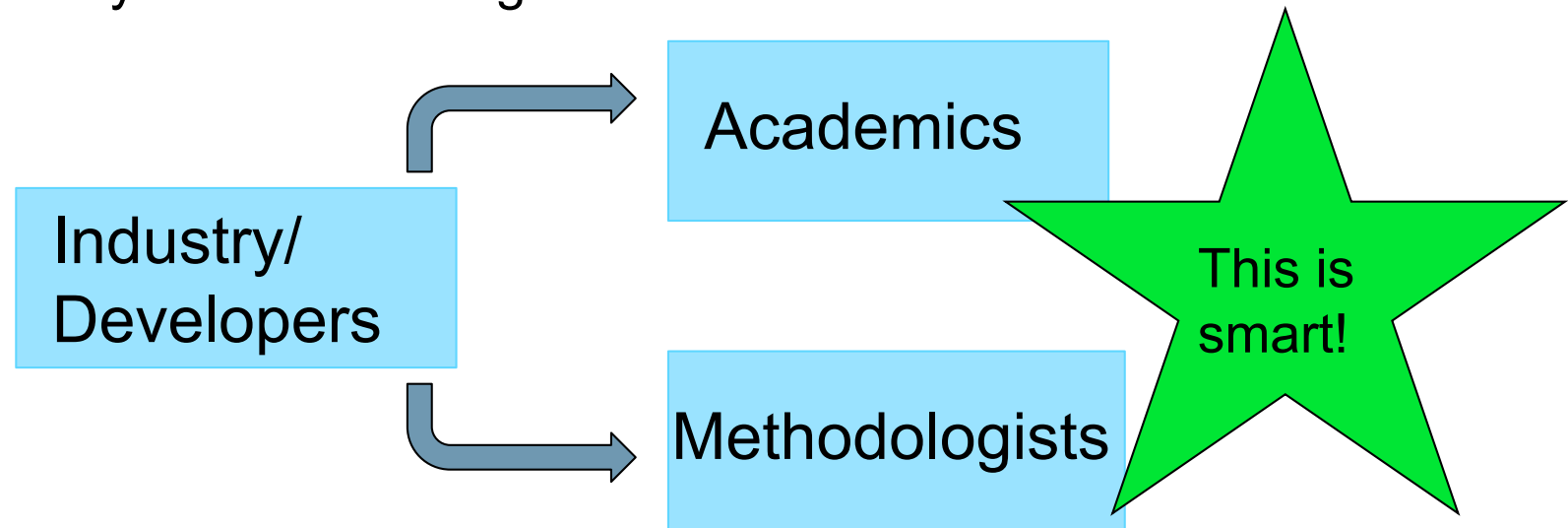
Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

A recipe for success

Our work needs to be

- driven from the demands of the industry/developer community, and
- enabled and formulated by the research community, and
- popularized by the methodologists.



We need a theoretical basis that is widely shared and supported, one that crosses the boundaries between the different software development camps.

Some challenges addressed by SEMAT

Industry

Big companies have many processes.

Challenges:

- Reuse practices
- Reuse training
- “Reuse” of people
- Evolutionary improvement is hard

Developers

Want to become experts. Challenges:

- Their skills are not easily transferable to a new product.
- Their career path follows a zig-zag track from hype to hype.

Academics

Asked to educate and research. Challenges:

- The Gap between research and industry
- No widely accepted theory
- Teaching instances of methods doesn't create generalists

Methodologists

Every method is a soup of practices. Challenges:

- Have to reinvent the wheel

SEMAT can have significant impact on the software community.



Agenda

- On what went well and what went wrong
- Addressing what went wrong
 1. Practices
 2. A new user experience
 3. Practices are not dead, they are enacted
 4. Result
- There must be a kernel
- The Semat kernel: track 3 and 4
- If successful what impact can we expect?
- Wrap up

Final Words



Questions

Thank You

ivar@ivarjacobson.com

The Universals

Kernel properties

- Concise.
- Scalable.
- Extensible.
- Measurable.
- Formally specified.
- Broad practice coverage.
- Broad lifecycle coverage.
- Broad technology coverage.

The Universals

Criteria for inclusion

- Universal
- Significant
- Relevant
- Defined precisely
- Actionable
- Assessable
- Comprehensive.

- Let's now start to talk about the **Universals** which belongs to track 3: Which are the universal alphas? The very root has n top alphas. In our case (EssWork) they are:- Opportunity, (which is an intangible but onto which we can attach a business case work product, a budget, and lots of other stuff)- Requirement (which are what you call Intent which I like). Here you can attach reqt spec, use case model, but all these are practice specific- System. Here you can attach design model, use case realizations, code, deployment model, ...all are practice specific- Project. There is always a project. Here you can attach project plan, iteration plan, backlog...practice specific stuff- Team. There is always a team. Here you can have sub-alphas such as team members etc.- Way of working. Another word for method/process, whatever. Here you can attach descriptions describing your way of working. In EssWork this is done by attaching a number of practice descriptions. All these are top alphas. Sub-alphas are always practice-specific. For instance, if you

- Some questions I have got: What is Guidance? I think it is a work product attached to the alpha Way-of-working? Tool. We probably need a new language construct ToolHuman operator. We have an alpha called Team and it has sub-alphas Team_member. If this is not enough we may have to add a new language construct WorkerAutomatic operator. Could be Worker with the attribute Automatic.Language. Is this a new language construct, or a Tool? Program. This is a work product attached to the System alpha or to sub-alphas of the System alpha.

The Kernel Language

- **The kernel language contains constructs that we need to define in track 4, such as :- Method/methodology/process or as I prefer to call them: Way-of-working- Practice - Pattern and KindofPattern- Alpha and sub-alpha- Work product- Competency- Activity and KindofActivity**