Software – “Ways or Working” and Practices over Process

Kim Horn

Version 1.2 **1 August 2021**

Agenda

- Practice Vs Process
- SEMAT / ESSENCE
  - Kernal Core Elements
  - Alpha States
  - Backlogs
- Krushen
- SPEM / EPF
  - Concepts
  - Practices
- APG & GSA
- End
- Issues with Process in IT

Process for Process Sake

Process is not Enough to effectively describe a Way of Working

- Just as, or more important are:
  - Practices
  - Guidance
  - Principles
  - Capability Patterns
- Focus on Practices not Process;
- Two well defined bodies of work (and Tools) help us with this non-process material:
  - **SEMAT/ESSENSE Tool: EssWorkBench**
  - **SPEM & EPF Tool: Eclipse Process Framework & Example OpenUP**

This presentation looks at these 2 approaches with an emphasis on Practices over Processes.

Way of Working Background Influences

Approach	What ?	Why ?
TOGAF	The Open Group Architecture Framework – Open Group	• Defacto Standard EA Framework. • Numerous Certified practitioners.
SPEM	Software Process Engineering Model - OMG	• Well accepted Model of Software Engineering • We have library material in SPEM from APG. • Supported by Enterprise Architect Tool.
EPF	Eclipse Process Framework	• Open Source tool for building and documenting Software Process and Knowledge. • Implements SPEM. • We have EPF Tools. • We have TOGAF implemented in EPF, supported by APG. • We have Solution, Governance add-ons from APG.
OpenUP	Open Unified Process	• Pragmatic Lean Unified Process(UP). • We have OpenUP in EPF.
EUP	Enterprise Unified Process	• Extends the UP with EA concerns. Shows a way to link EA Frameworks and SDLCs
SEMAT	Software Engineering Method and Theory	• Initiative to find universal standard for software engineering.
ESSENCE	SEMAT core process design language	• Application of the Standard with focus on core practices to provide a way of working
EssWork	Essential Work	• SEMAT and ESSENSE Tools, we have the workbench

APG = Armstrong Process Group - TOGAF and EPF contributors

Practices Vs Process

Issues with Processes

Processes lead to paradoxes and contradictory behaviour, including*:

- The problem of Denied Commonality
- The problem of Creating a Complete Process
- The problem of Adopting a Complete Process
- The problem of One Size Fits All Process
- The problem of Out-Of-Sync Processes
- The problem of Acquiring Process Knowledge
- The problem of Stupid Processes
- The problem of Organisation processes do not work together.
- The problem of No process for building process.
- The problem of Scalability and Agility

"Taken all together these problems are making it harder and harder for companies and teams to establish an effective way of working. The processes are becoming a barrier to change rather than an enabler."

Ivar Jacobsen

* Each detailed at end of pack

Process Vs Practise

Process	Practice
The way tasks are organised	The way tasks are done
Routine	Spontaneous, Judgemental
Orchestrated	Improvised
Assumes Predictable Environment	Responds to a changing unpredictable environment
Relies on explicit knowledge	Relies on tacit knowledge
Linear	Network or Web like

Seely and Duguid, “Balancing Act: how to capture knowledge without killing it.” 2000 HBR

Practices

A “practice” provides a way to systematically and verifiably address a particular aspect of a problem.

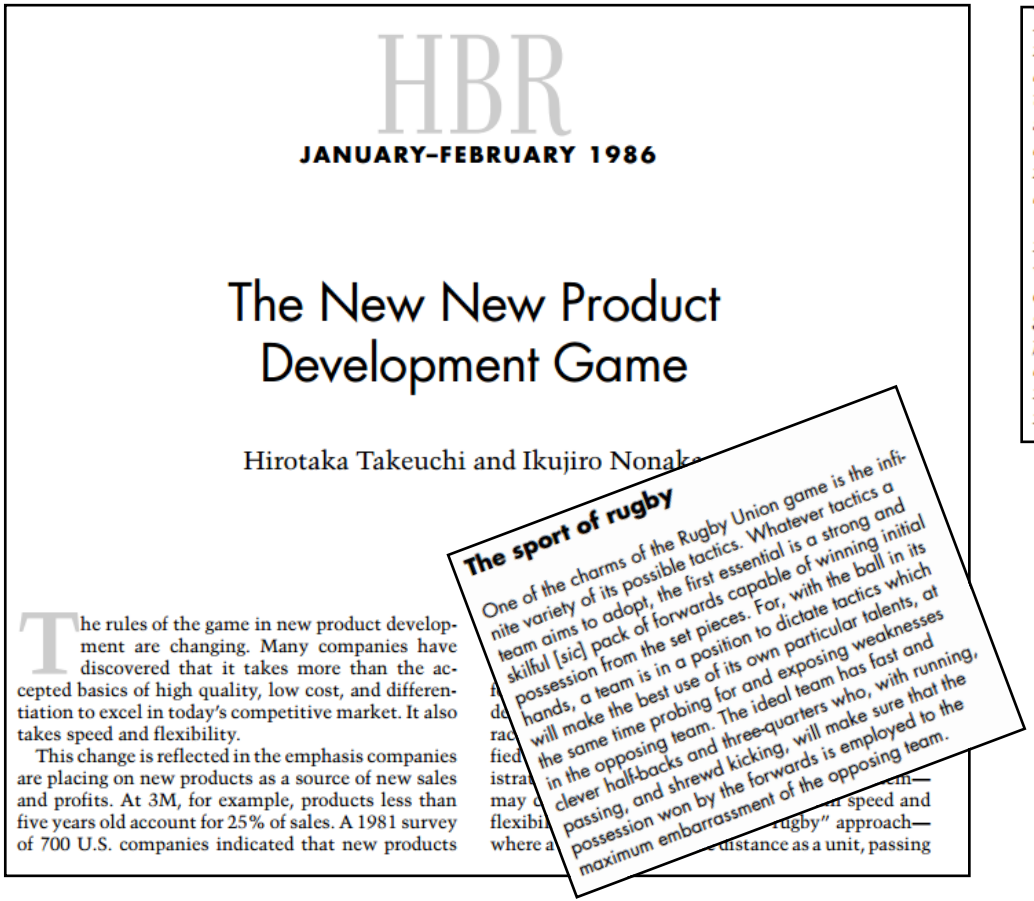
- A practice **does not attempt to address the entire problem**. Rather, a practice attacks a particular aspect of the problem.
- A practice is **systematic** in that someone can articulate it - it is not a black art.
- A practice has a **clear beginning and end**, and tells a complete story in usable chunks.
- A practice **includes its own verification**, providing it with a clear goal and a way of measuring its success in achieving that goal. Without verification, the practice is not complete.

Because of these qualities, **practices can be developed, learned and adopted separately**, and that can be used in conjunction with other practices to create easily understood and coherent ways-of-working.

Practices

- A practice provides the guidance to characterize the problem, the strategy to solve the problem, and instructions to verify that the problem has indeed been addressed.
- It also describes what supporting evidence, if any, is needed and how to make the strategy work in real life.
- Put simply, practices describe what to produce, how to produce it, and the competencies needed to perform the practice.
- Practices also provide pre-defined patterns to tailor and tune their use.
- Practices are independent and decoupled with process.

Scrum as an example of an Agnostic Independent practise



HBR

JANUARY–FEBRUARY 1986

The New New Product Development Game

Hirotaka Takeuchi and Ikujiro Nonak

The rules of the game in new product development are changing. Many companies have discovered that it takes more than the accepted basics of high quality, low cost, and differentiation to excel in today's competitive market. It also takes speed and flexibility.

This change is reflected in the emphasis companies are placing on new products as a source of new sales and profits. At 3M, for example, products less than five years old account for 25% of sales. A 1981 survey of 700 U.S. companies indicated that new products

The sport of rugby

One of the charms of the Rugby Union game is the infinite variety of its possible tactics. Whatever tactics a team aims to adopt, the first essential is a strong and skilful [sic] pack of forwards capable of winning initial possession from the set pieces. For, with the ball in its hands, a team is in a position to dictate tactics which will make the best use of its own particular talents, at the same time probing for and exposing weaknesses in the opposing team. The ideal team has fast and clever half-backs and three-quarters who, with running, passing, and shrewd kicking, will make sure that the possession won by the forwards is employed to the maximum embarrassment of the opposing team.

In today's fast-paced, fiercely competitive world of commercial new product development, speed and flexibility are essential. Companies are increasingly realizing that the old, sequential approach to developing new products simply won't get the job done. Instead, companies in Japan and the United States are using a holistic method—as in rugby, the ball gets passed within the team as it moves as a unit up the field.

This holistic approach has six characteristics: built-in instability, self-organizing project teams, overlapping development phases, "multilearning," subtle control, and organizational transfer of learning. The six pieces fit together like a jigsaw puzzle, forming a fast flexible process for new product development. Just as important, the new approach can act as a change agent: it is a vehicle for introducing creative, market-driven ideas and processes into an old, riged organization.

- Cannon AE-1 Camera (1976)
- Fuji-Xerox FX-3500 Copier (1978)
- Honda City Car (1981)
- NEC PC 8000 Computer (1979)
- **and recently Software…**

Process Entanglement – dependencies

"**Every process can be considered a collection of typically tangled and tightly coupled practices.** Once existing practices are separated, methodologists can focus on capturing best-practices in a reusable and extensible format without repeating or replacing existing practices.

Treating processes as collections of practices fundamentally changes the role of the processes. They just become a short-hand way of referring to a known set of mutually supportive practices, and their adoption acts as a useful starting point or goal for projects trying to change their way-of-working."

Where is the real knowledge ?

'In a practice, the "well defined steps" constitute more of a toolkit to be used by the practitioner; in a process the steps are more rigidly followed. In a practice, the intelligence remains almost entirely with the practitioner; designers move as much intelligence as possible into processes. Bob Lewis

The proportion of business practices that can be formally codified in process form is really only the tip of the iceberg, and that the vast majority of ‘‘knowledge’’ encompassed in a successful practice is uncodified and held tacitly in the minds of the staff performing the task.

Seely Brown (2001)

IT Catalyst Definition

A ***process*** is a series of well-defined steps, which, if faithfully followed, will deliver repeatable, predictable results. Processes extract as much knowledge and skill as possible from the human beings who participate in them, embedding them in the process steps. Processes are well-suited to mass production.

A ***practice,*** in contrast, is a different way to organize work. While still a series of steps, they are loosely defined -- the specifics are left to the practitioners who make them work, based on their knowledge, skill, experience, and judgment. Practices are well-suited to the creation of unusual or unique items, dealing with ambiguous situations, and especially in competitive arenas, where predictability is a great way to lose.

Processes and practices aren't opposites. They're the endpoints of a continuum of possible ways to organize work ... to design ***business functions***. What matters is recognizing what each situation calls for, rather than assuming the right answer is always a well-defined process. In many cases faithfully following the steps isn’t enough.

SEMAT / ESSENCE

Where are we with software development 'Processes' ?

Everyone of us knows how to develop our own software, but as a community we have no widely accepted common ground, there are many 'processes'.

Goal 1: Is it possible to 'distil' an Essence of commonality, a Kernel ?
Goal 2: is there a better way to do things ?

Provide a Better User Experience

SEMAT and ESSENCE provide a new way of looking at 'SDLC', and a set of tools for implementing practices.

- Most process material is presented in a boring way.
- It's a Law of Nature: people don't read process manuals.
- Process Web sites don't get used, hard to find what you need.

A Better way:

- Start with a small set of material and add, rather than starting with a complex set and trying to simplify by removing;
- Use simple card metaphor;
- Actually use physical cards:
  - can be manipulated, ordered, compared, drawn on and annotated.
- Use simple check lists, as surgeons and pilots do;
- Lean: Provide only just enough information to apply the practice.

Essence of a SDLC – Way Of Working

A standard **Kernel** provides a baseline starting point – a "map" of the software development endeavour.

Key language concepts: **Alpha**, **Activity Space**, **Work Product** and **Activity**

Alpha = Abstract-Level Progress Health Attribute

Alphas are subjects in a software endeavour whose evolution we want to understand, monitor, direct and control.

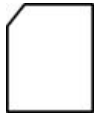
Alphas exist regardless of their concrete representation. E.g. requirements exist even if not documented.

Practices add details and provide specific guidance on particular aspects of the software development

Kernel

Practice


Alpha


Work Product


Activity Space


Activity

Four key things are all that's required

1. We are **concerned** about: the Customer, the Solution and the Endeavour.

2. There are **things to work with,** that we want to understand, monitor (progress and health), direct and control.

3. There are **things to do.**

4. We Need **Competency to do stuff.**

That's it.

3 Key Concerns

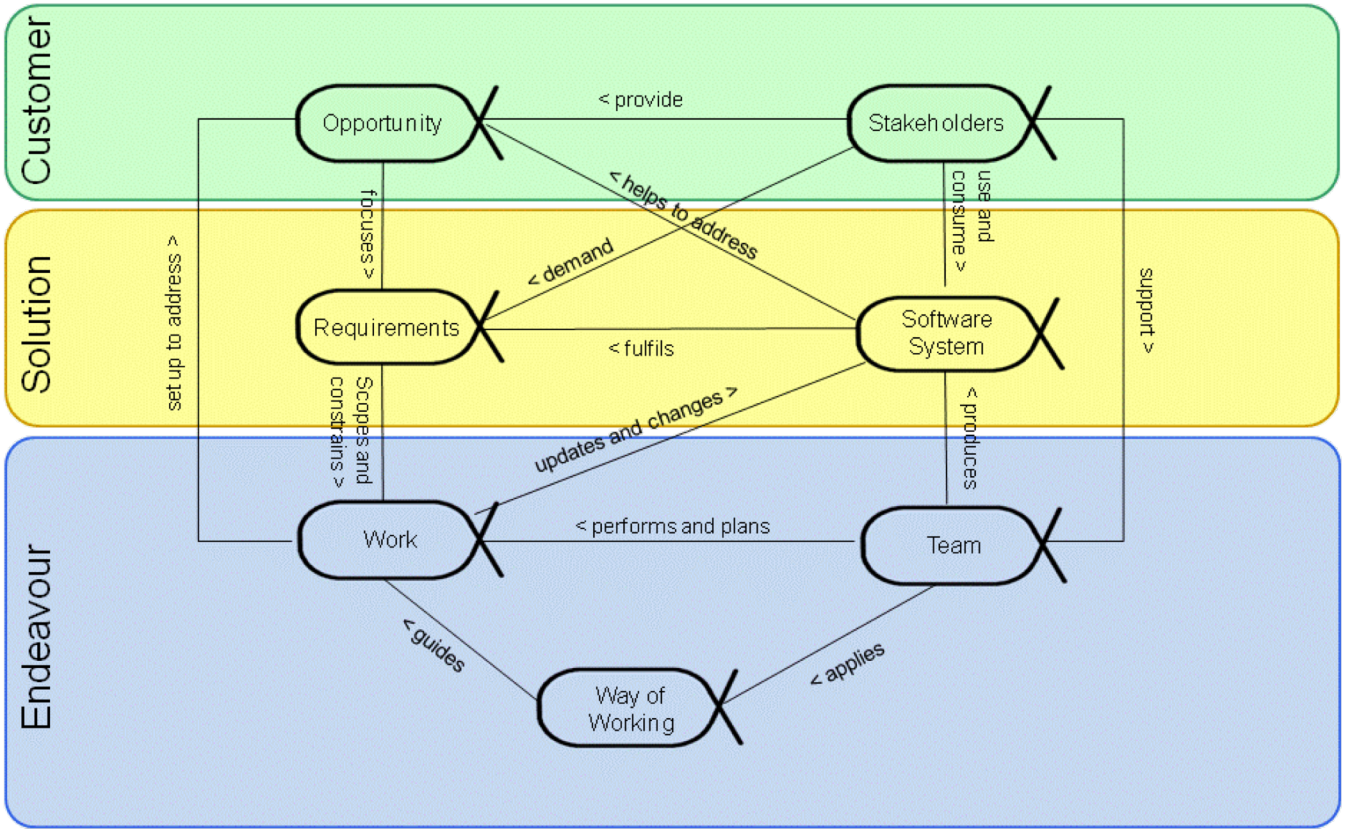
Undertaking this Endeavour to create a Solution with the Customer

Customer

Solution

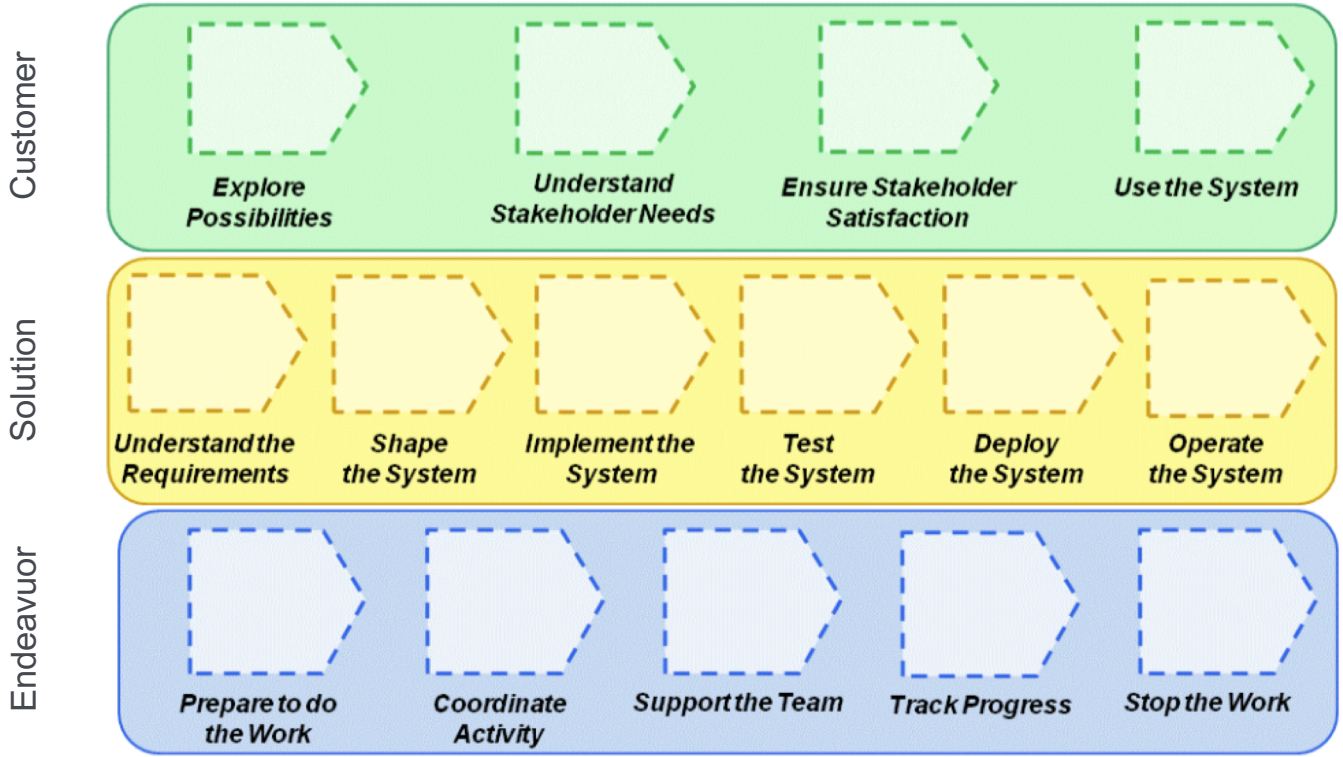
Endeavour

Core Elements – things to work with – Alpha's



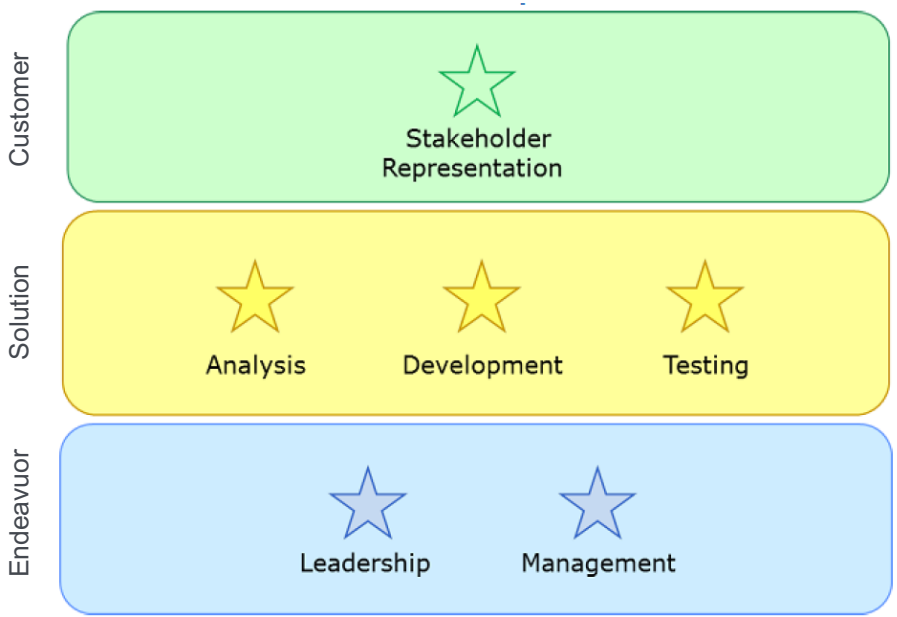
Core Elements – Activity Spaces: placeholders for Activities

Things to do within the concerns

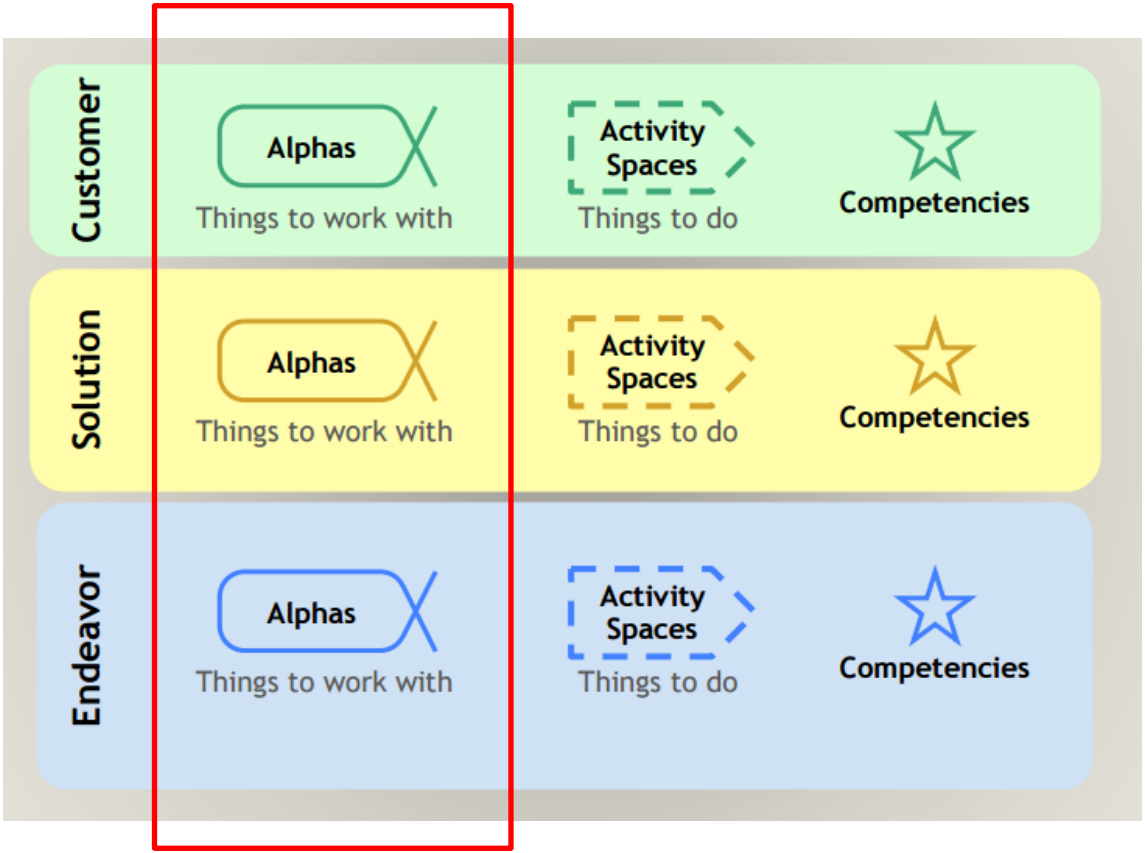


Note: there is no left to right implied order, this is not a process description.

3 Core Competencies



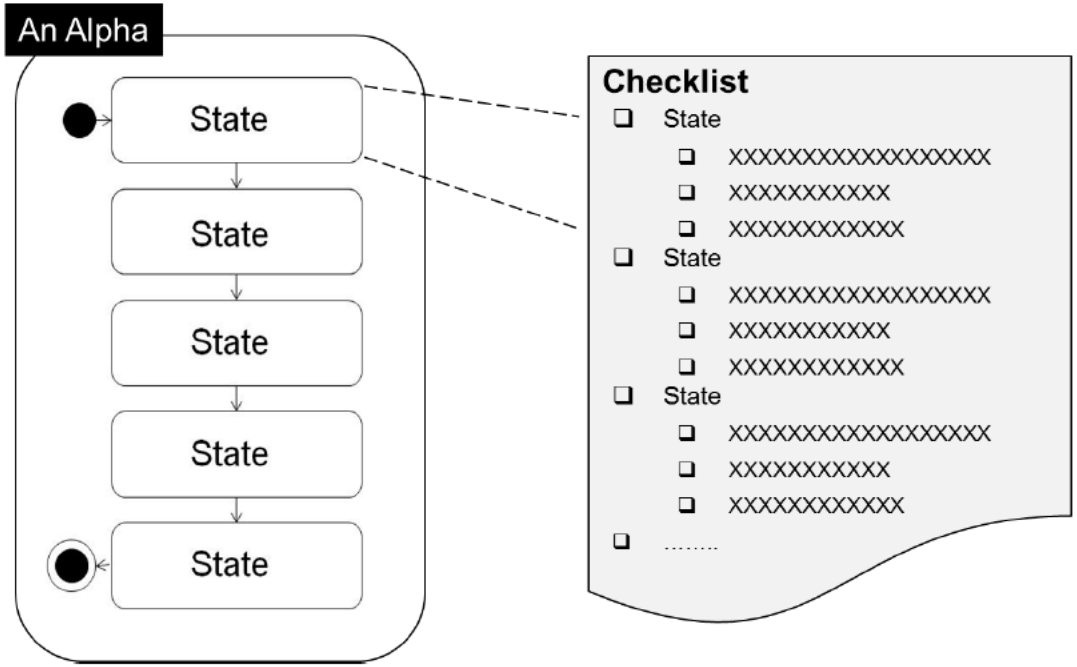
Essence Kernel



Just using Alphas provides significant value

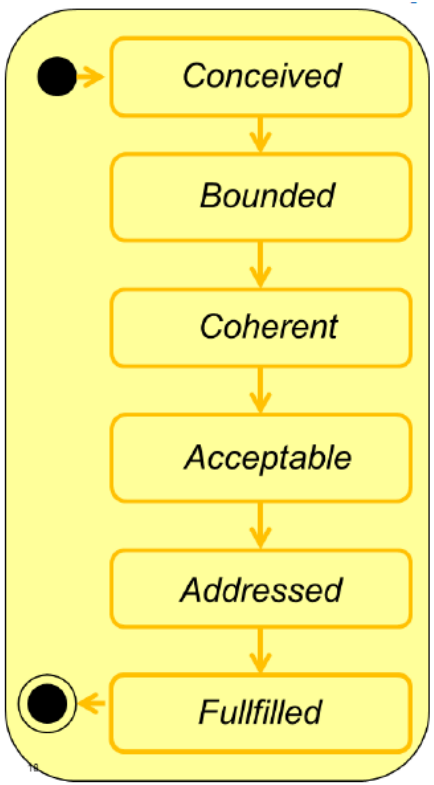
Structure of an ALPHA

An Alpha has State and is simply managed with Check Lists



An Alpha does not tell you how to get from State to State

Example - Requirements States



The need for a new system has been agreed.

The purpose and theme of the new system are clear.

The requirements provide a coherent description of the essential characteristics of the new system.

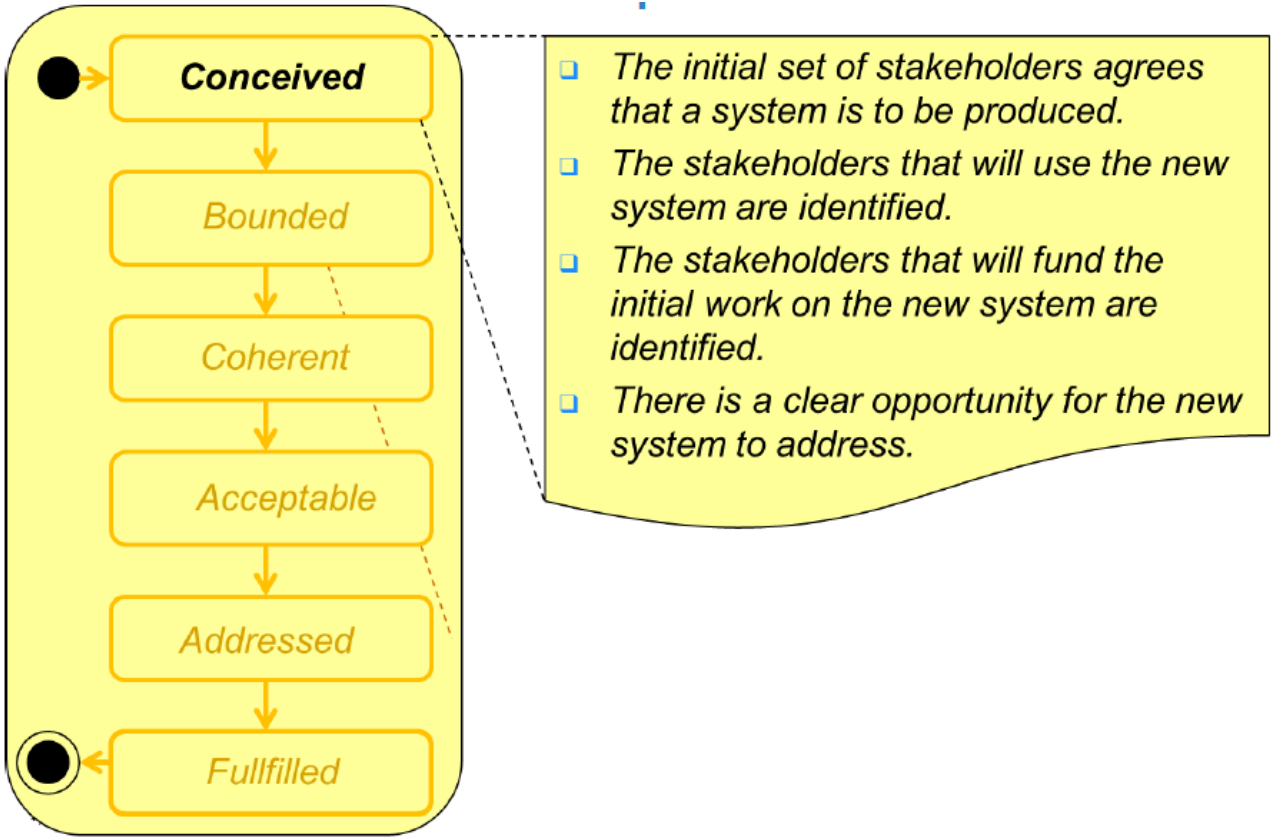
The requirements describe a system that is acceptable to the stakeholders.

Enough of the requirements have been addressed to satisfy the need for a new system in a way that is acceptable to the stakeholders.

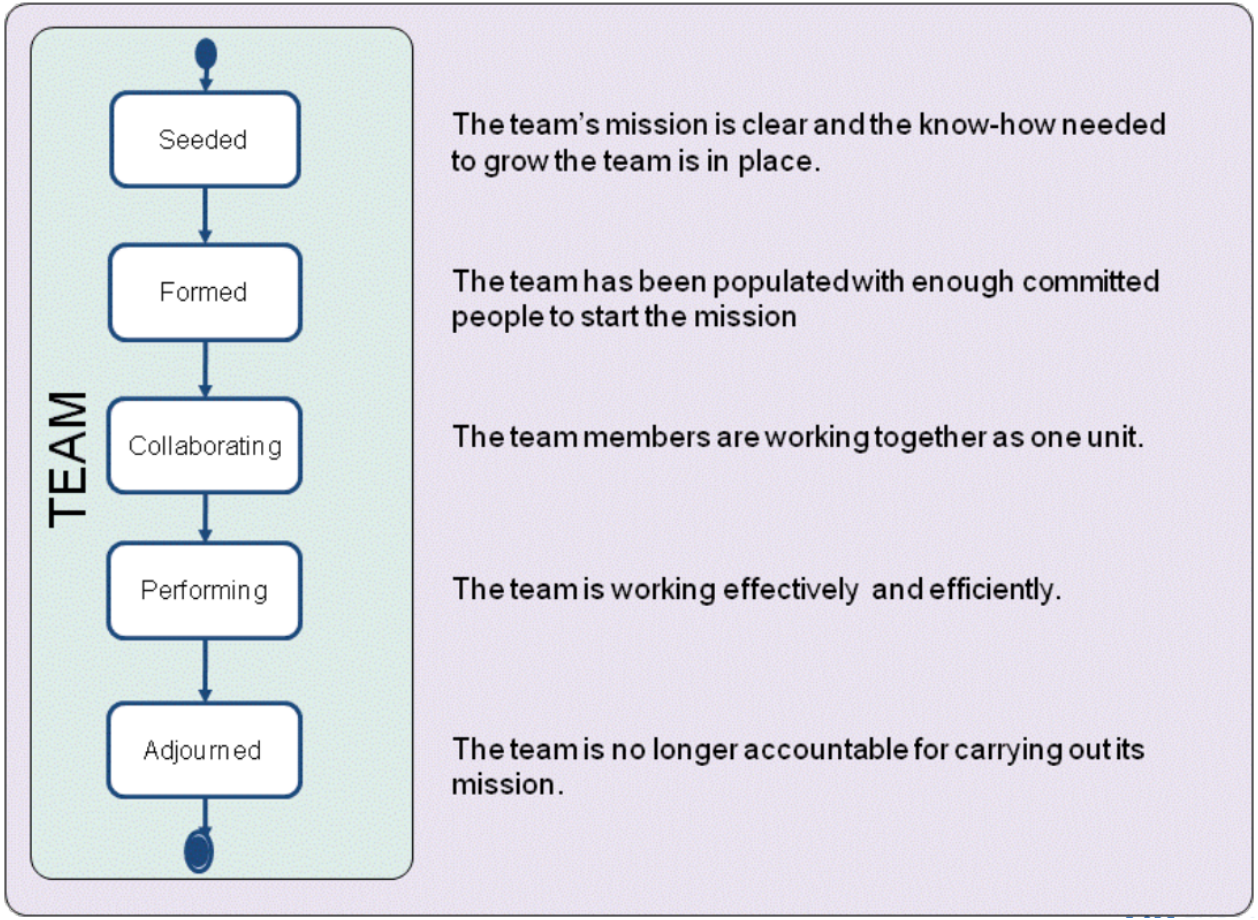
The requirements have been addressed to fully satisfy the need for a new system.

Note: This is not suggesting a waterfall approach to requirements, indeed could describe an iteration.

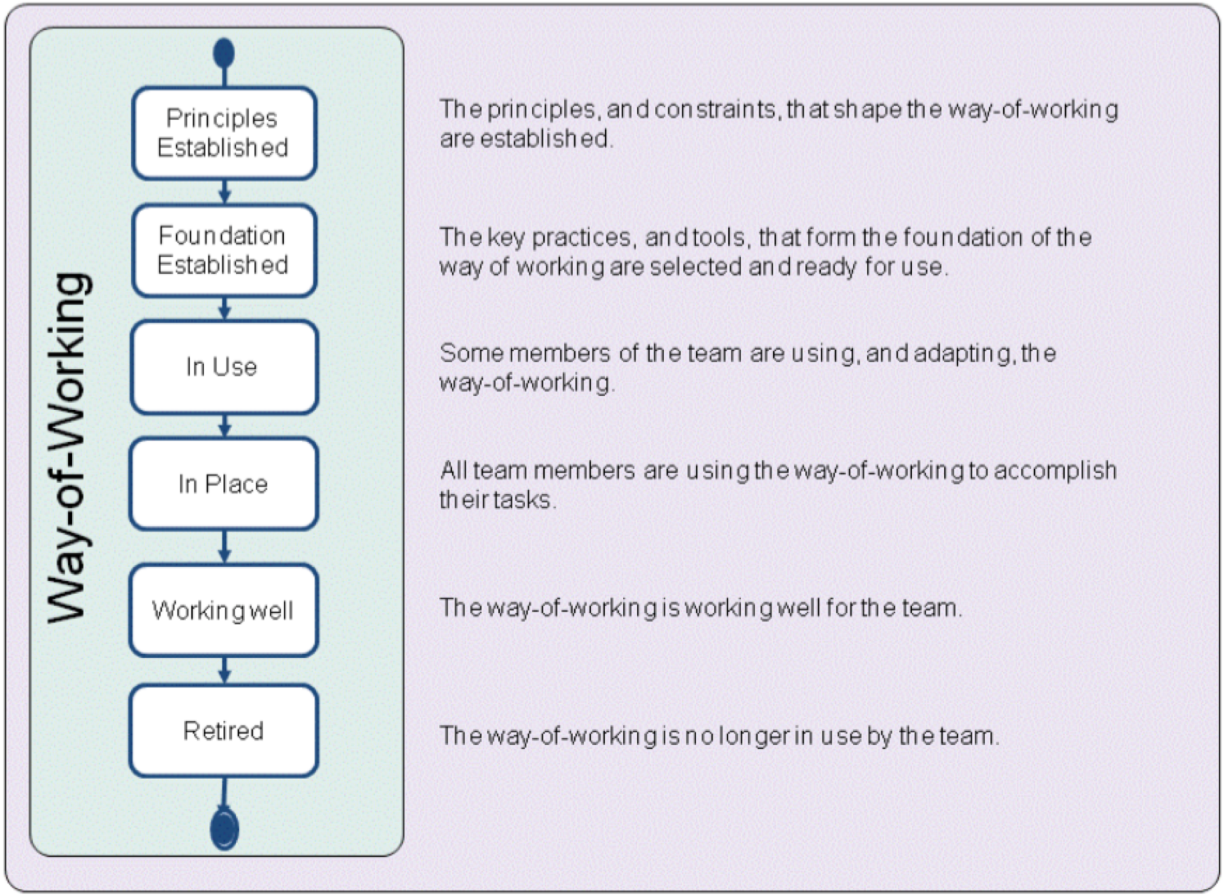
Each State has a CheckList



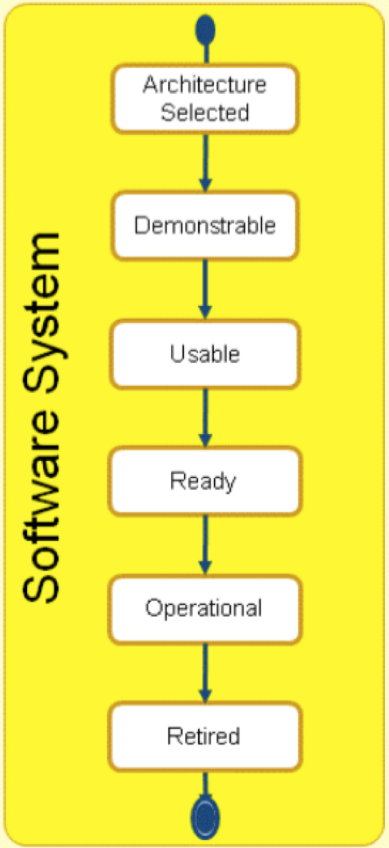
Example - Team



Example - Way Of Working



Example - Software System - States



An architecture has been selected that addresses the key technical risks and any applicable organizational constraints.

An executable version of the system is available that demonstrates the architecture is fit for purpose and supports functional and non-functional testing.

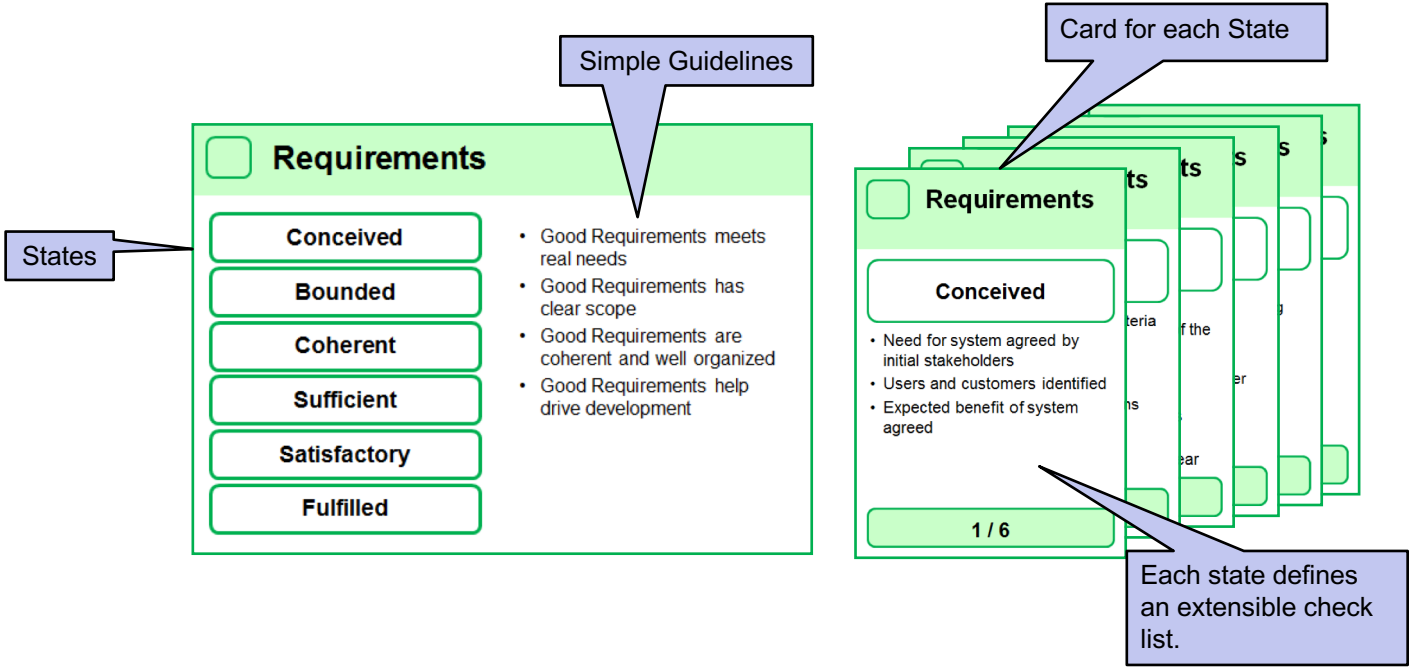
The system is usable and demonstrates all of the quality characteristics required of an operational system.

The system (as a whole) has been accepted for deployment in a live environment.

The system is in use in a live environment.

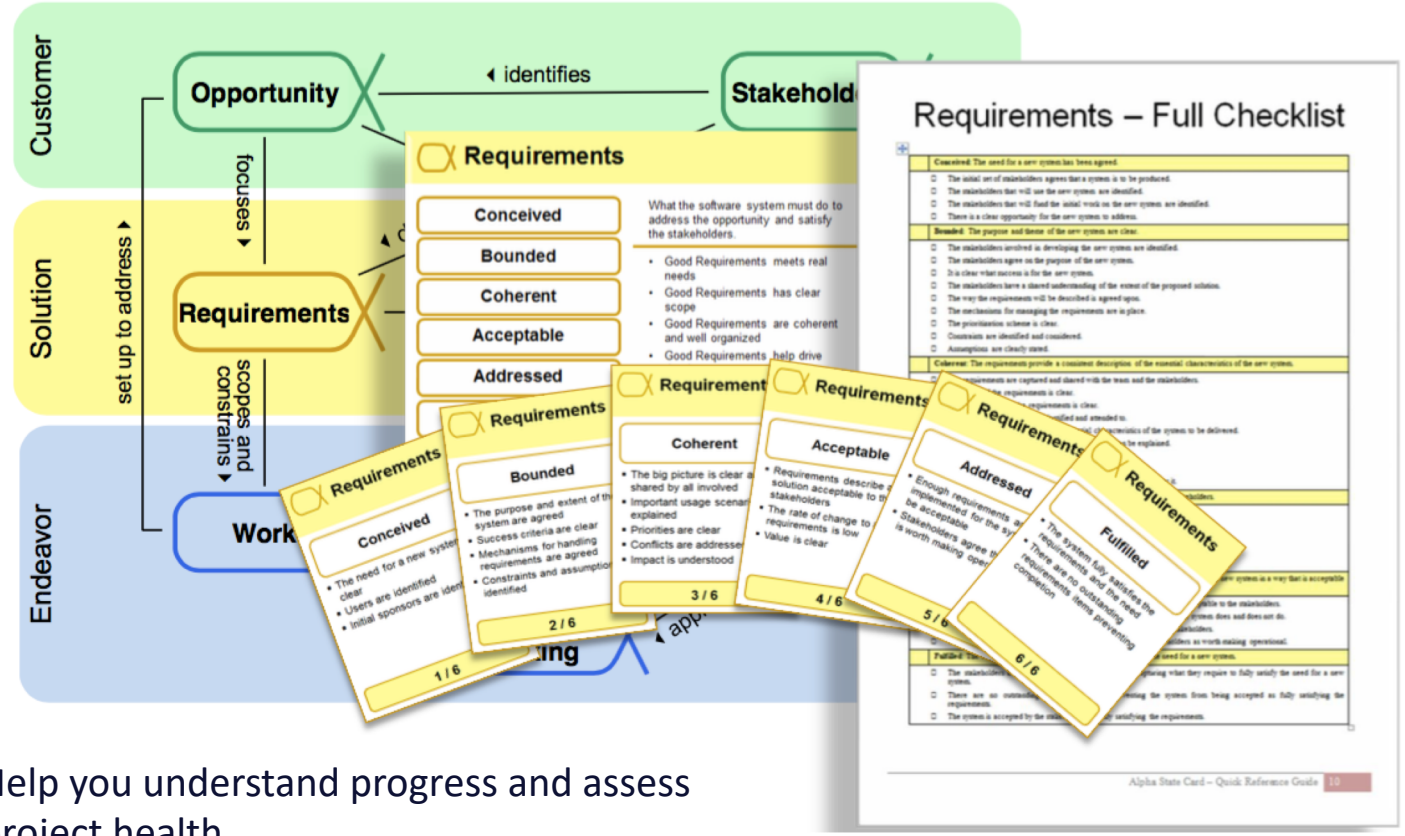
The system is no longer supported.

Cards and Checklist are Extensible



- Each Card (Alpha) is independent, so adding check list items, or cards, does not result in entanglement and dependencies.
- Cards are created and managed with ESSWorkBench Tool

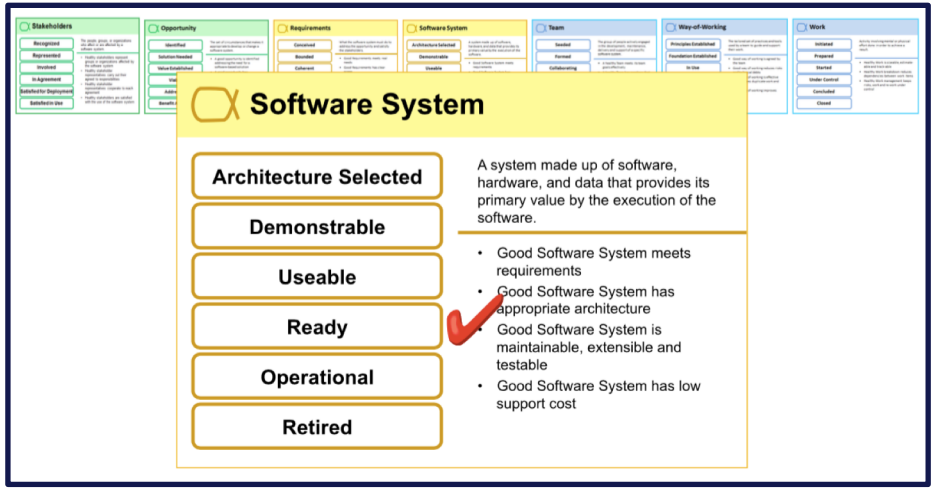
Kernel is Practical – Cards and Checklists



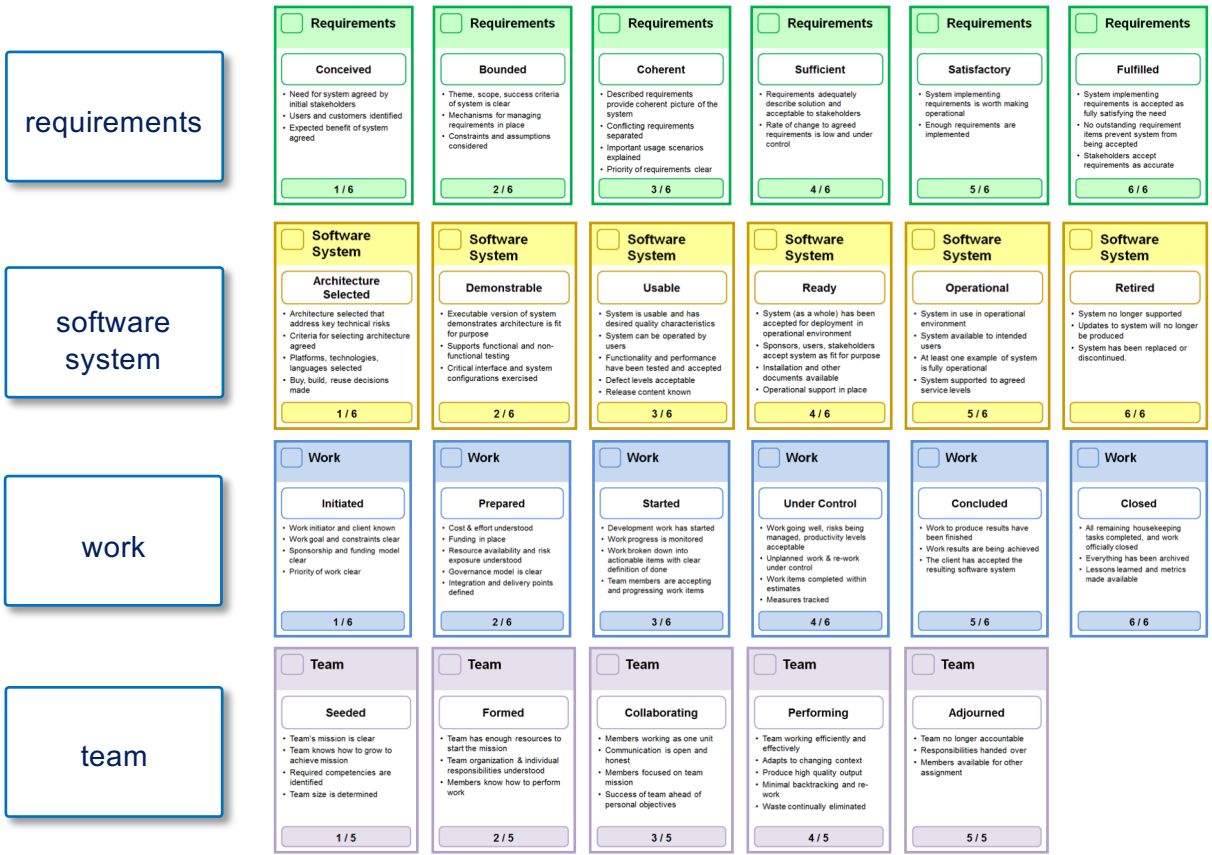
Help you understand progress and assess project health.

How do you use the cards ?

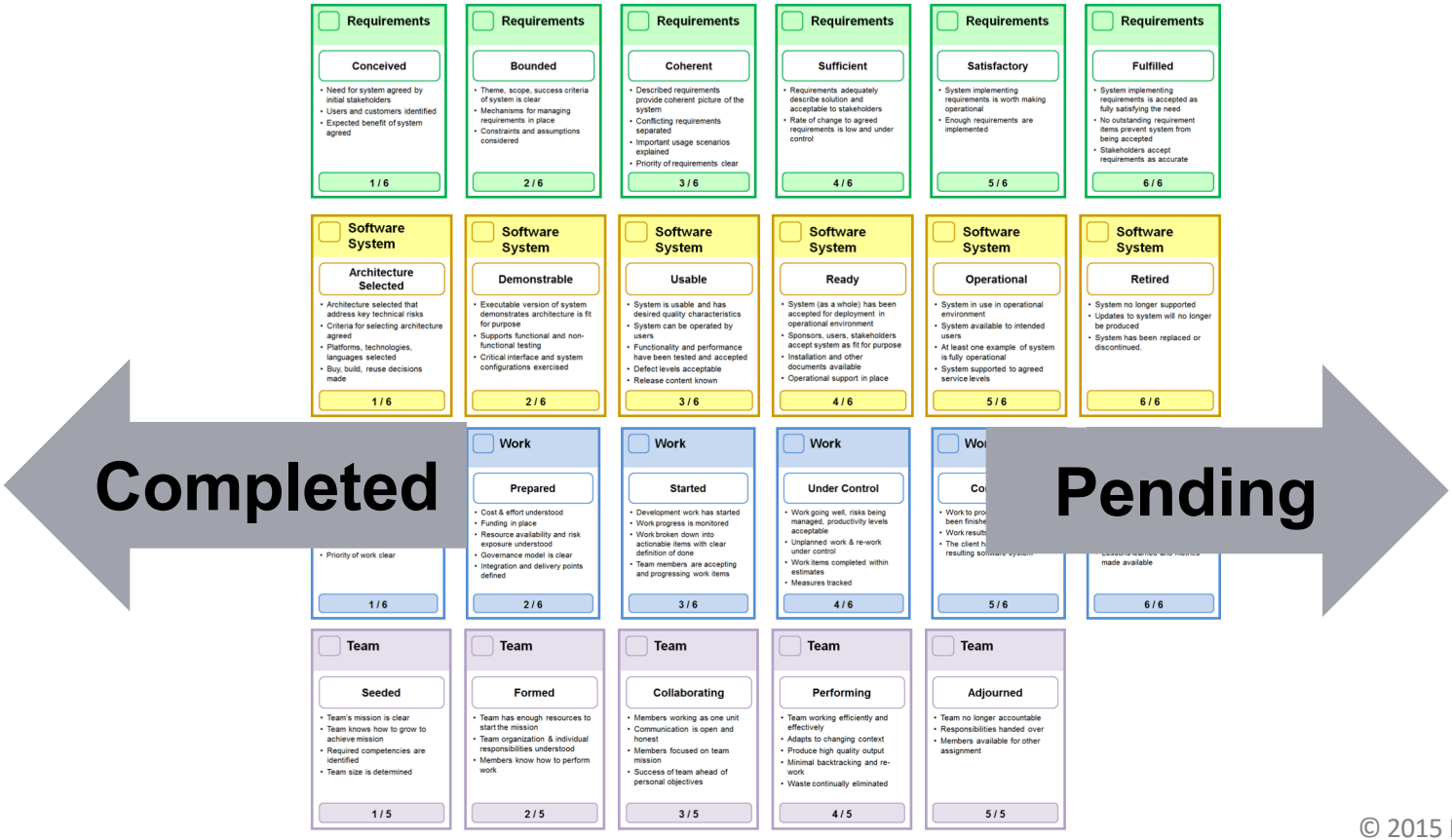
- Help work out what's Next.
- Where do we think we are up to :
  - Lay Cards Out
  - Play Poker
- Focus on Pain Points – find the root cause of challenges
- Retro's, Reviews



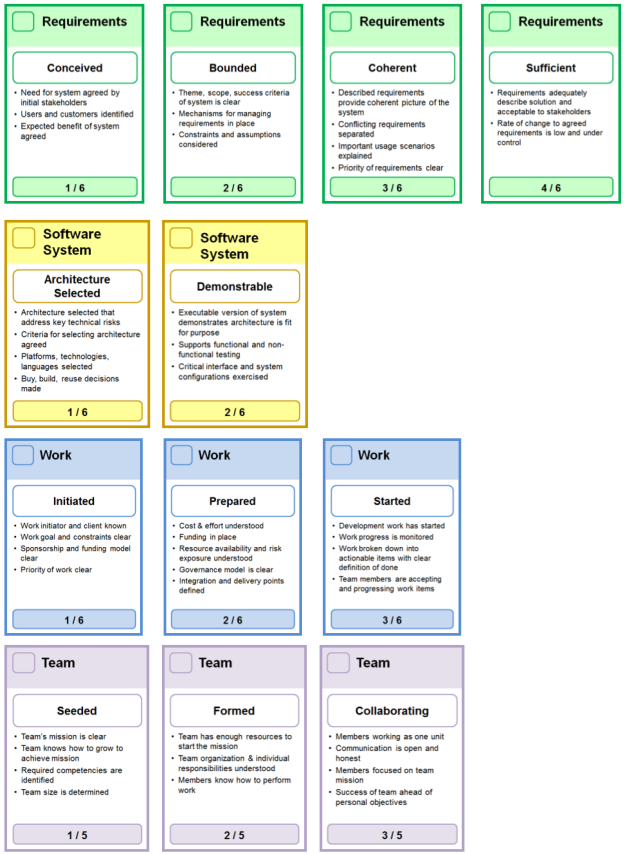
What is our Current State ?

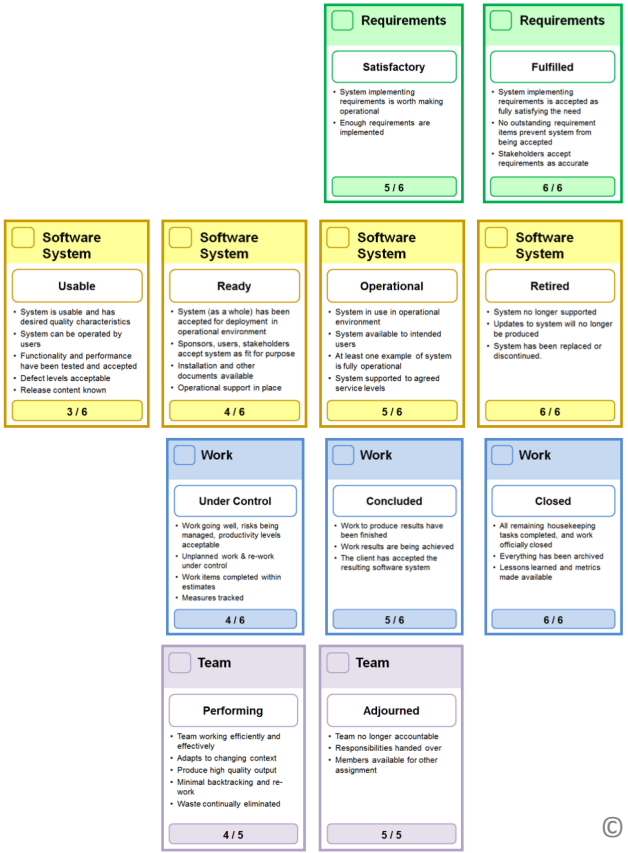


What is Completed; What is pending ?

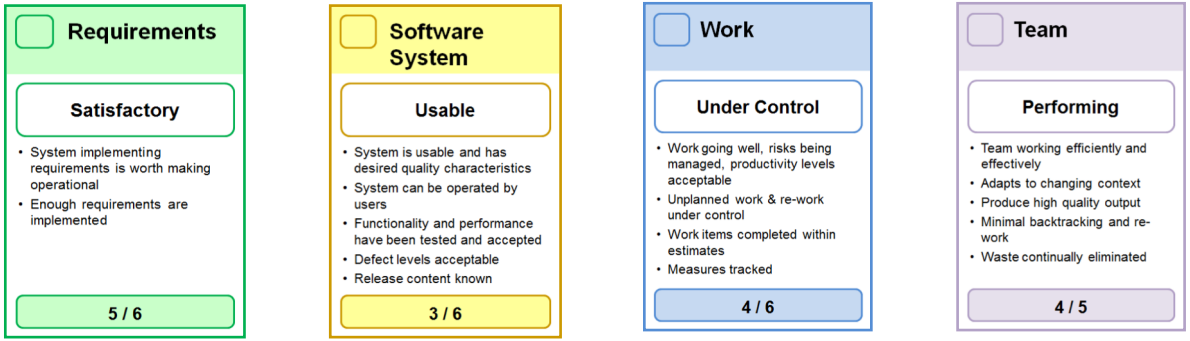


We are Here – What's next ?



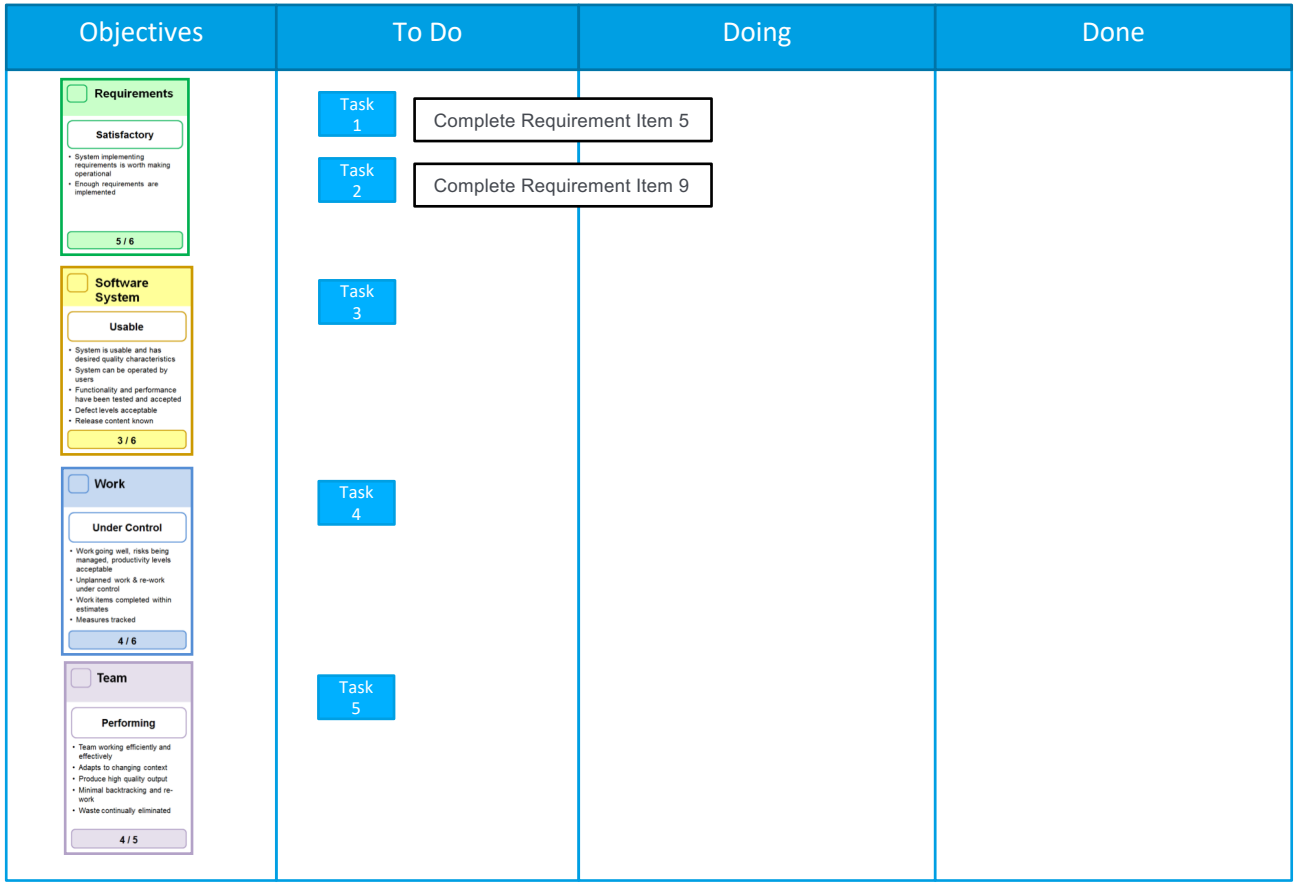


How to achieve Next State ?

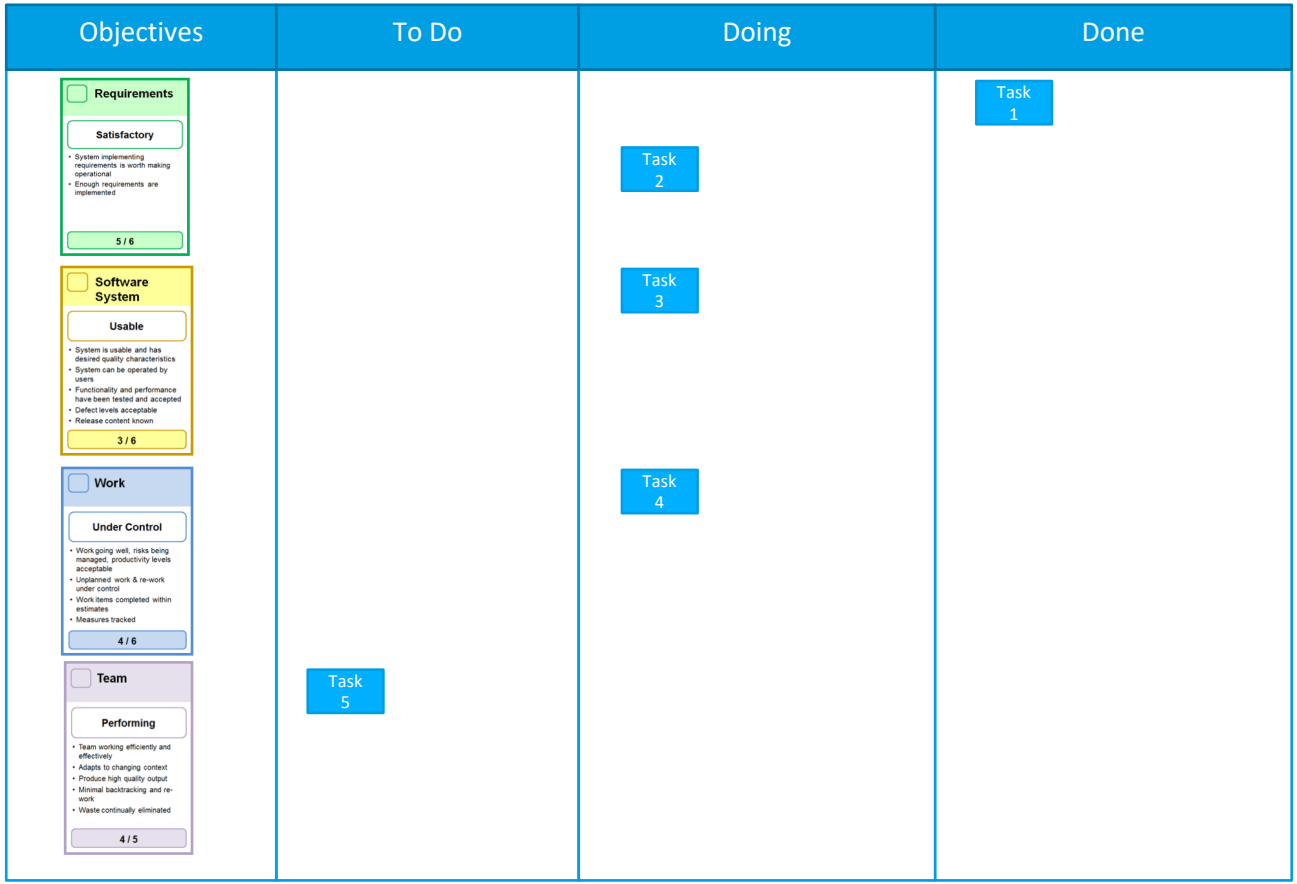


These are our Objectives for next Iteration. For each, we can define the tasks required to complete or progress the Alpha State.

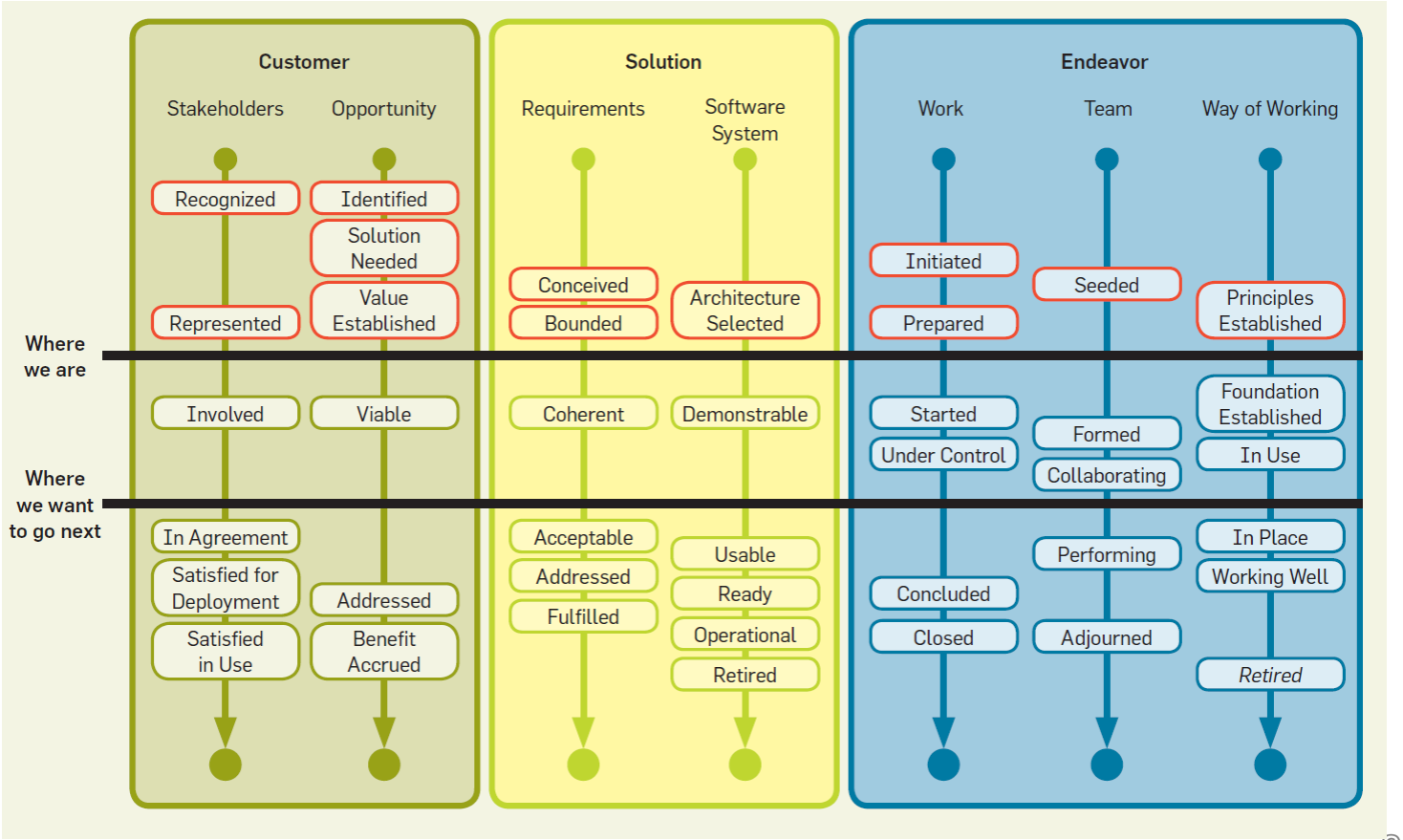
Task Backlog - Start



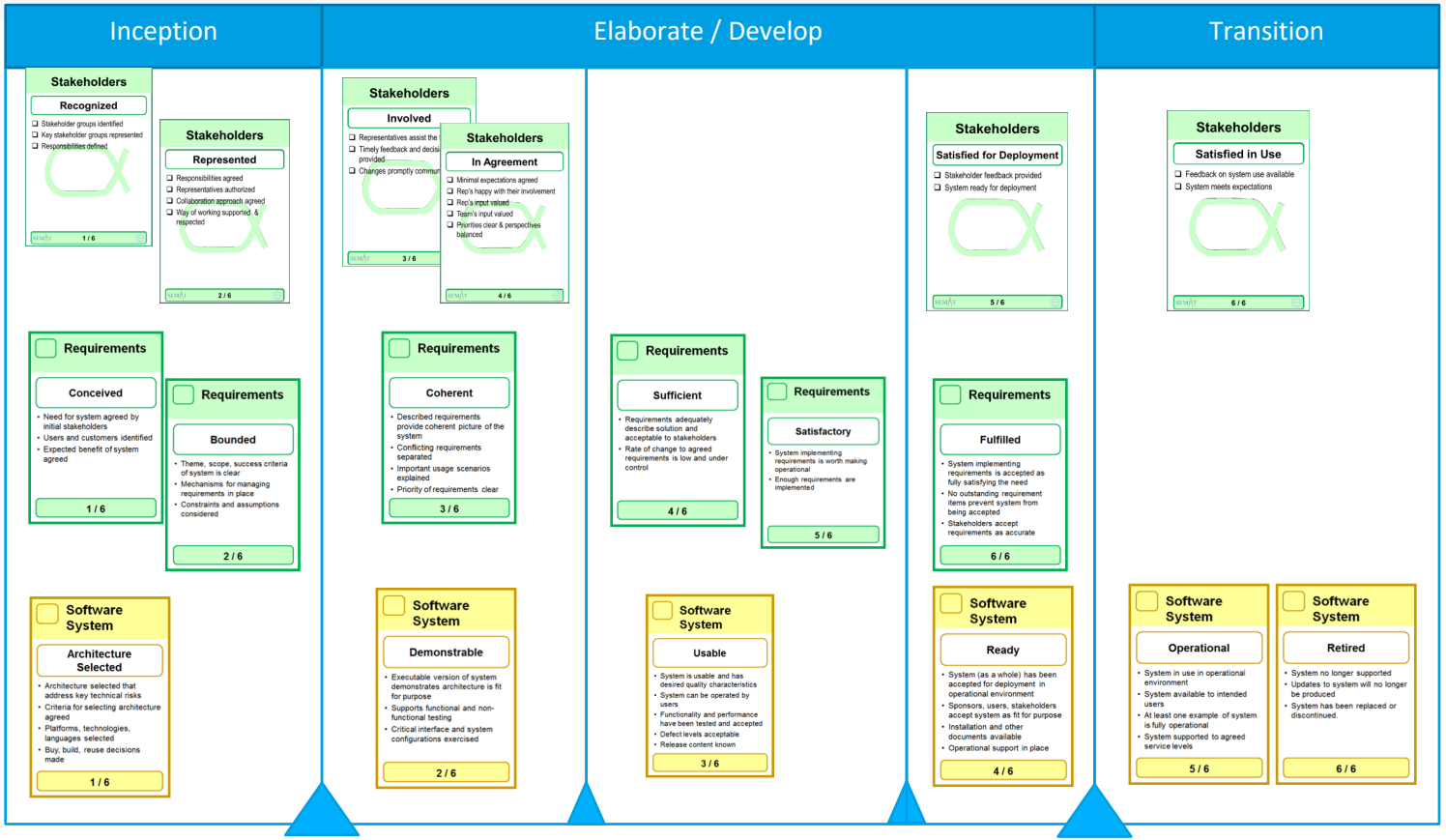
Task Backlog – In Progress



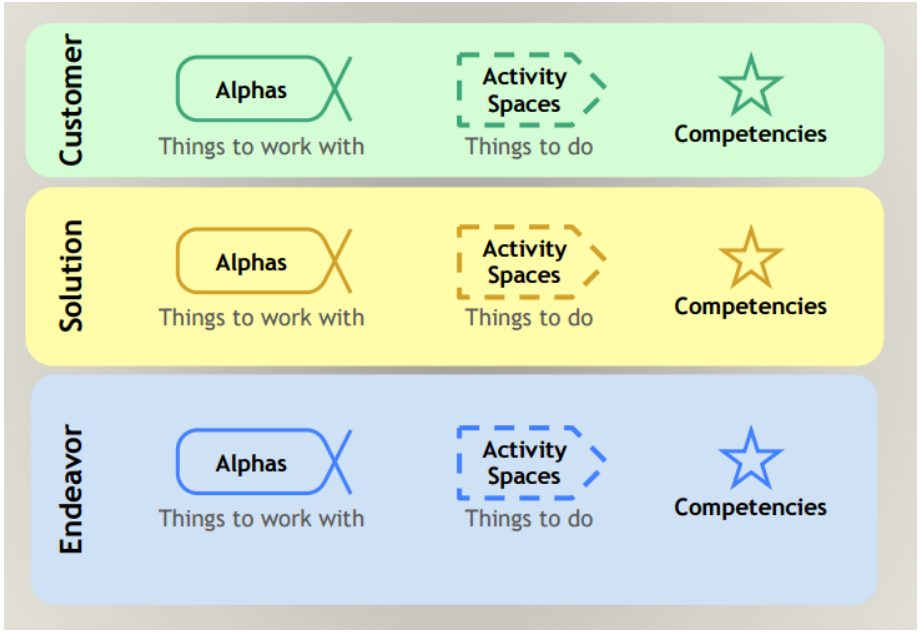
Process Agnostic - Where are we – Where Next ?



What about Governance Gates / Phases – another with milestones

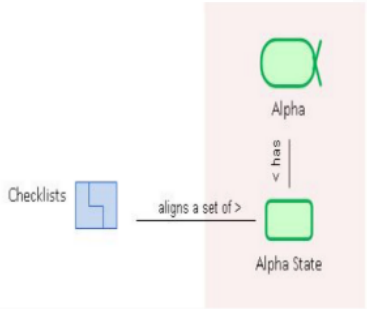


Just looking at Alphas provides many benefits

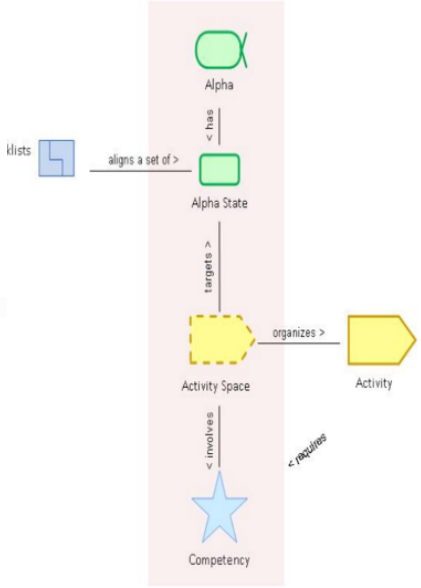


Types of ESENSE

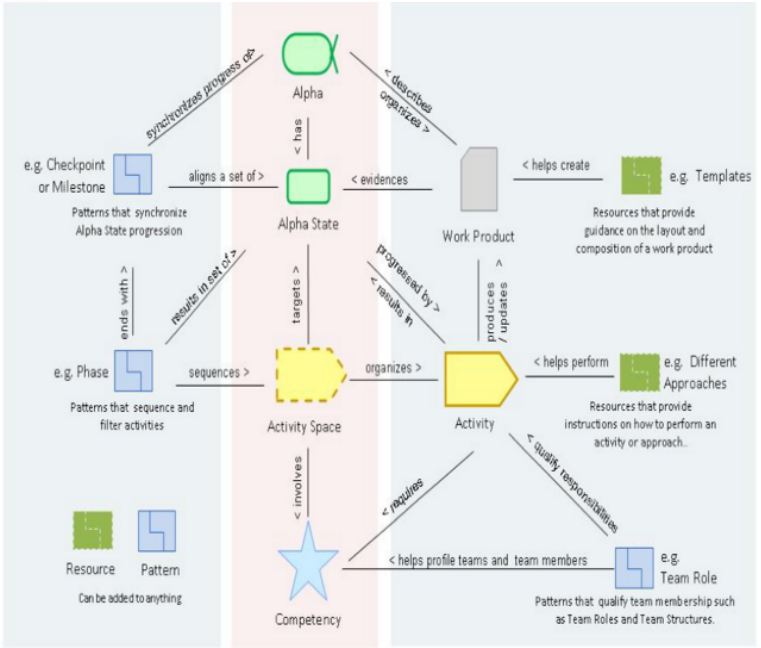
Lite



Kernel

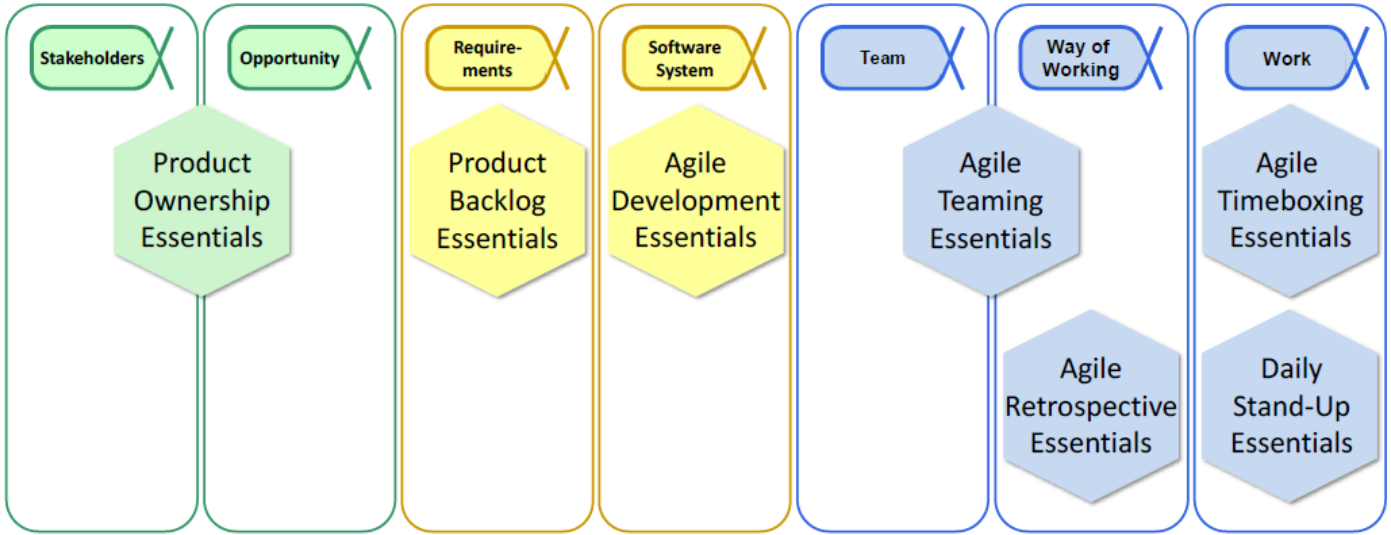


Plus



Extending the Kernel with Practices

Adding Practices on top of the Kernel creates a specific Method.
For example, an Agile method with 7 practices.

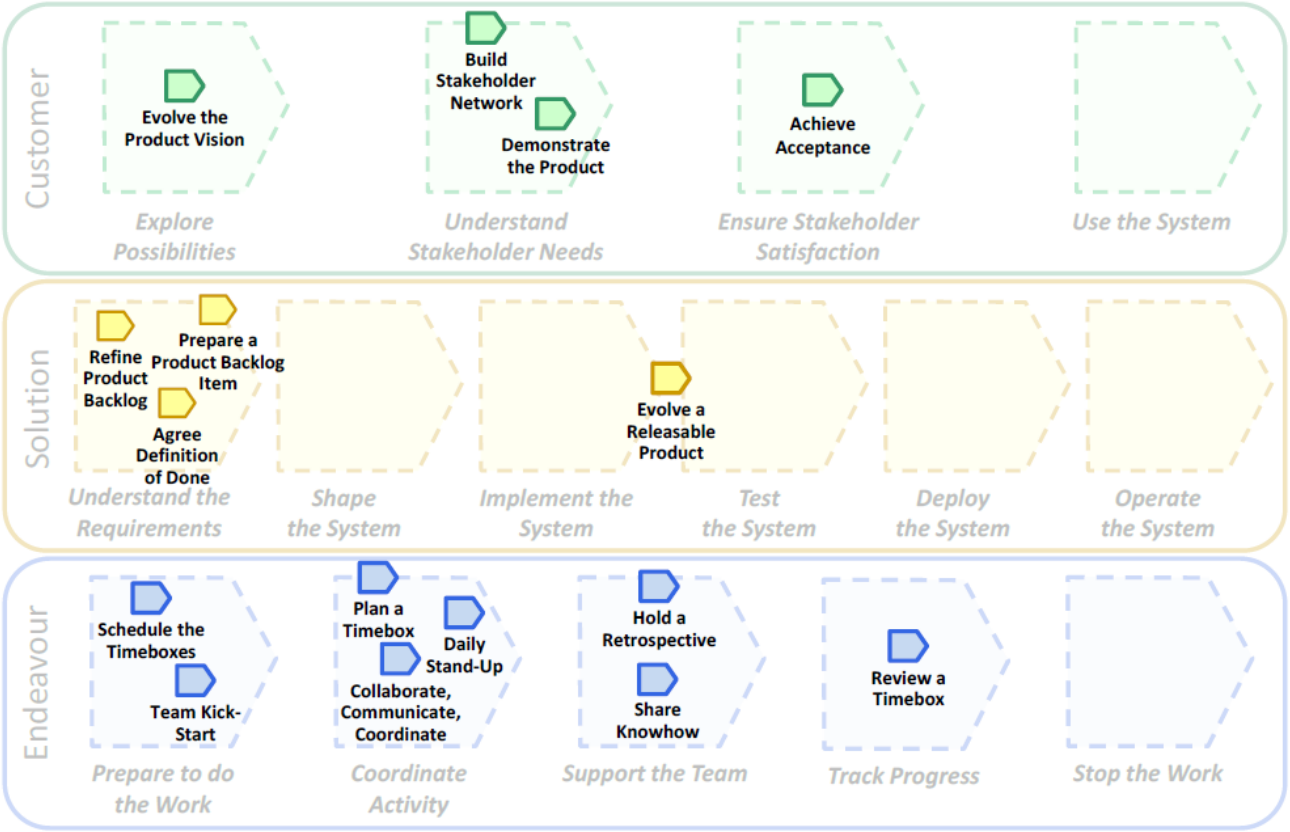


Practices can involve multiple Alphas.

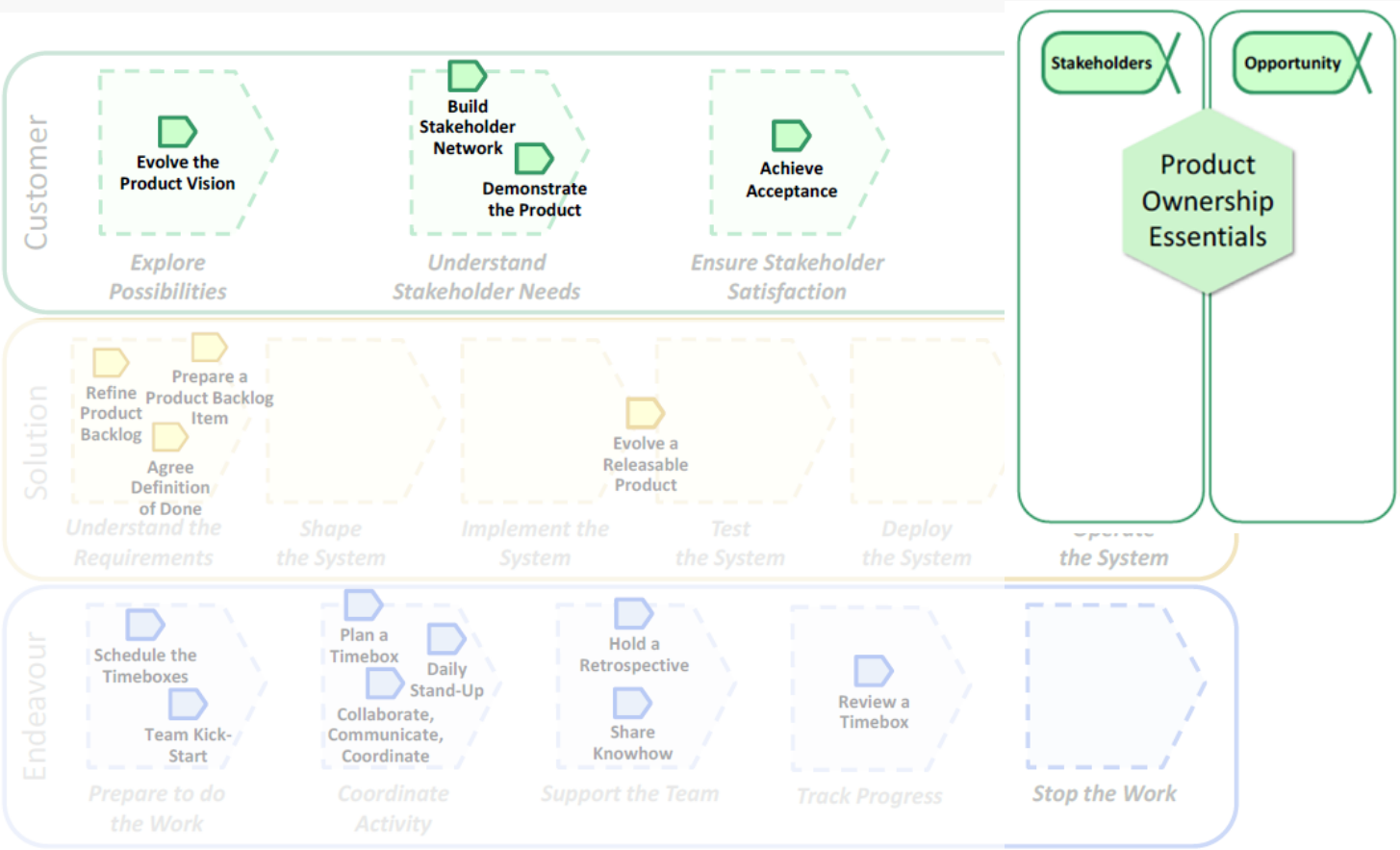
Agile Essentials Practises

- **Product Ownership Essentials** - own, evolve and communicate the product vision and guide the evolution of the product to achieve the vision
- **Product Backlog Essentials** - capture what the users of a software system want it to do as a priority-ranked list of independently buildable items
- **Agile Teaming Essentials** - a self-organizing team maximizes its performance by using a highly collaborative teaming approach
- **Daily Stand-up Essentials** - use a short, daily, whole-team meeting to reaffirm delivery focus, assess progress, agree work plans and action the removal of impediments to progress
- **Agile Development Essentials** - add value to a product by incrementally extending it while ensuring it remains usable, releasable and maintainable
- **Agile Retrospective Essentials** - make incremental improvements to the way of working through regularly repeated retrospectives
- **Agile Timeboxing Essentials** - progress the work as a series of focused timeboxes, and assess and re-plan the work at the end of each timebox.

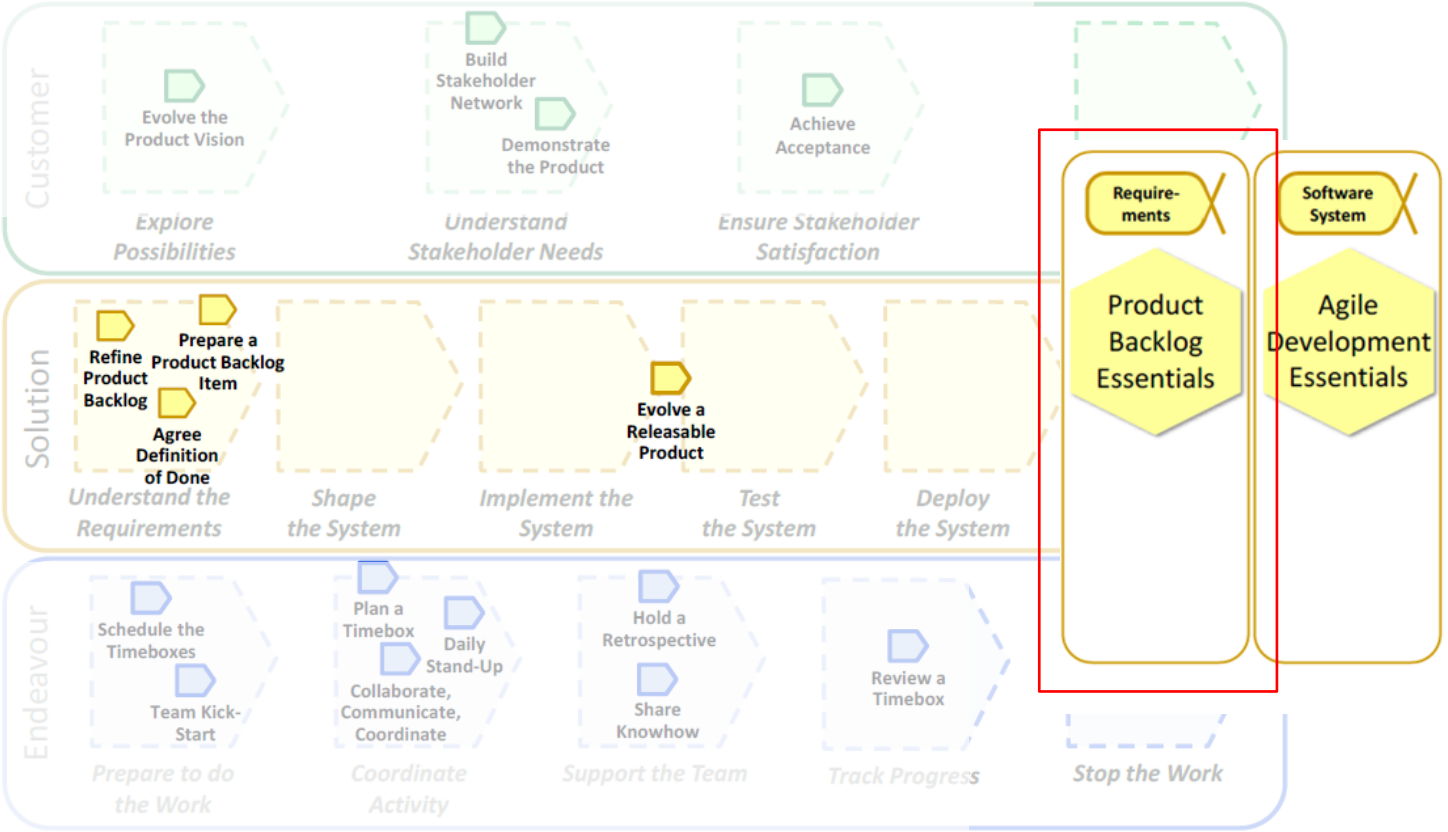
Agile Essentials – Activities on their Kernel Spaces



Customer Related Practise, Alphas and Activities



Solution Related Practise, Alphas and Activities



Example “Product Backlog Essentials” Practise

Activities – the things we do

- **Refine Product Backlog:** Get and keep the Product Backlog visible, up-to-date and in good working order, with high priority items agreed and well understood.
- **Agree Definition of Done:** Agree the quality criteria that will be used to determine whether any change to the product is fully and correctly implemented.
- **Prepare a Product Backlog Item:** Ensure that the Product Backlog Item is ready for development and that it is clear how it will be tested.

Alphas – the essential elements that we progress

- **Product Backlog Item:** Something to build into the product to enhance its value.

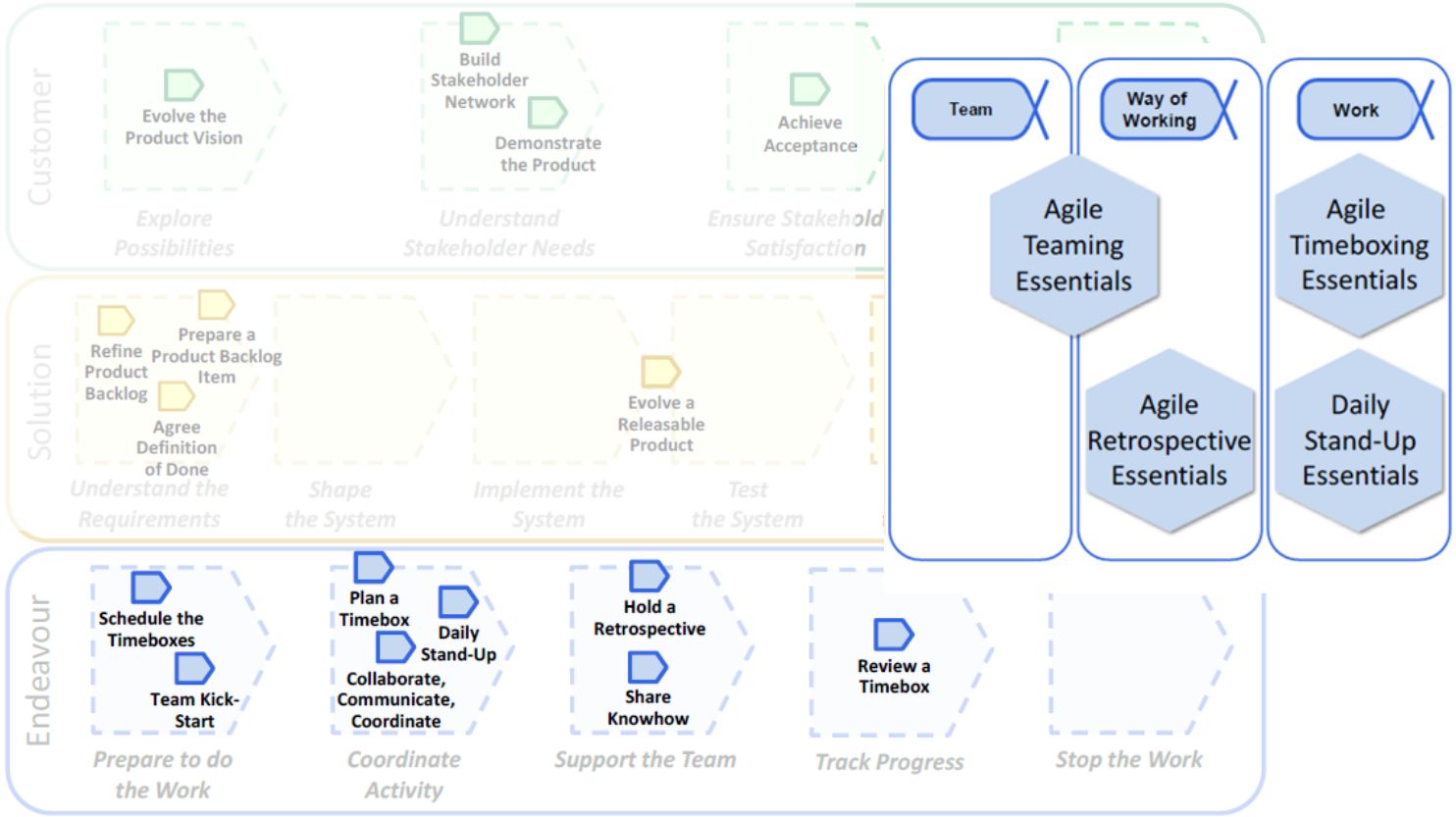
Work Products – the concrete things that we work with

- **Product Backlog:** An ordered list of things to build into the product to enhance its value.
- **Definition of Done:** The quality criteria that will be used to determine whether each Product Backlog Item is fully and correctly implemented.
- **Test Case:** Defines test inputs and expected results to evaluate whether a Product Backlog Item is fully and correctly implemented.

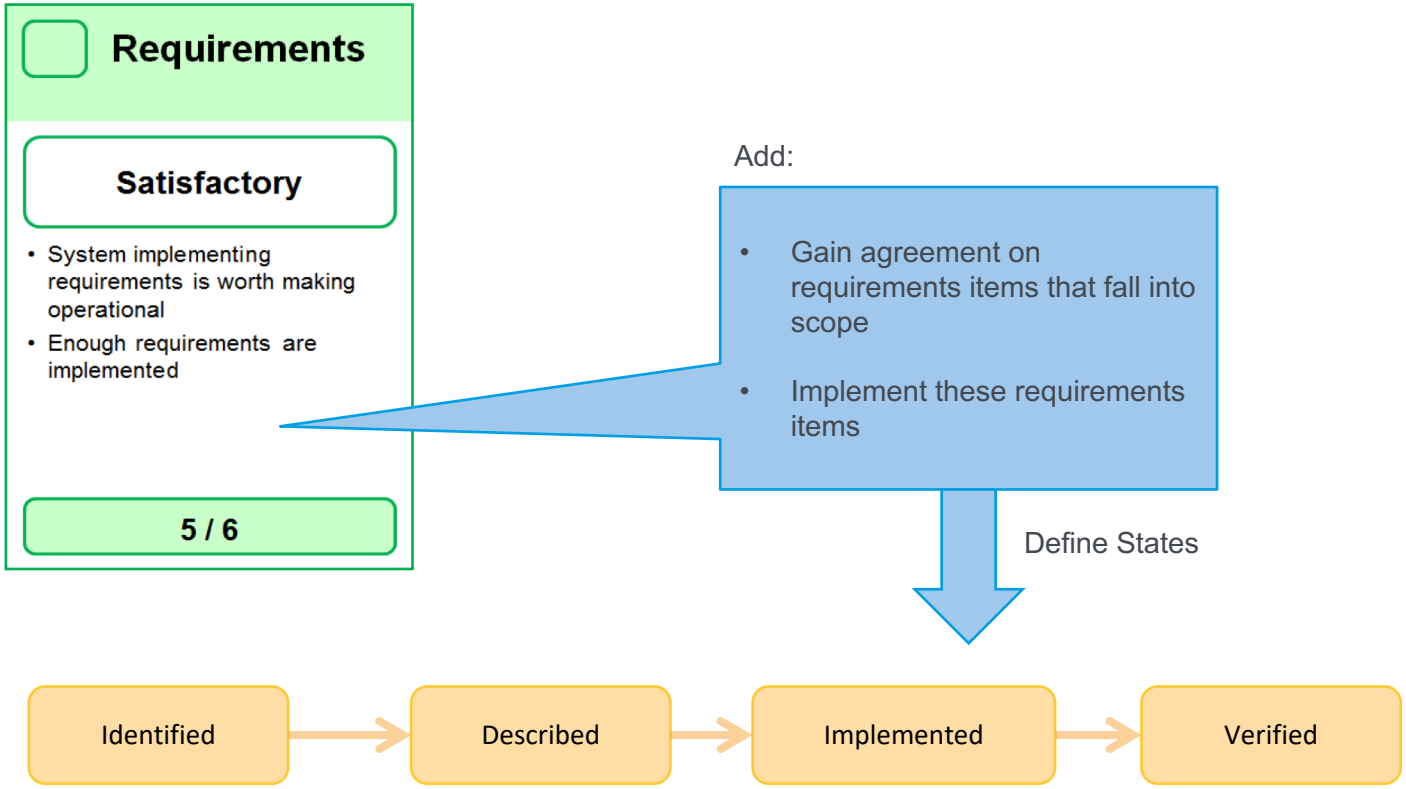
Patterns - supporting practice guidance

- **INVEST:** An acronym of the quality criteria for a Product Backlog Item to be ready for development: Independent; Negotiable; Valuable; Estimable; Small; Testable.
- **Relative Estimating:** Effort to get Product Backlog Items done is estimated not in absolute units of time, but relative to each other.

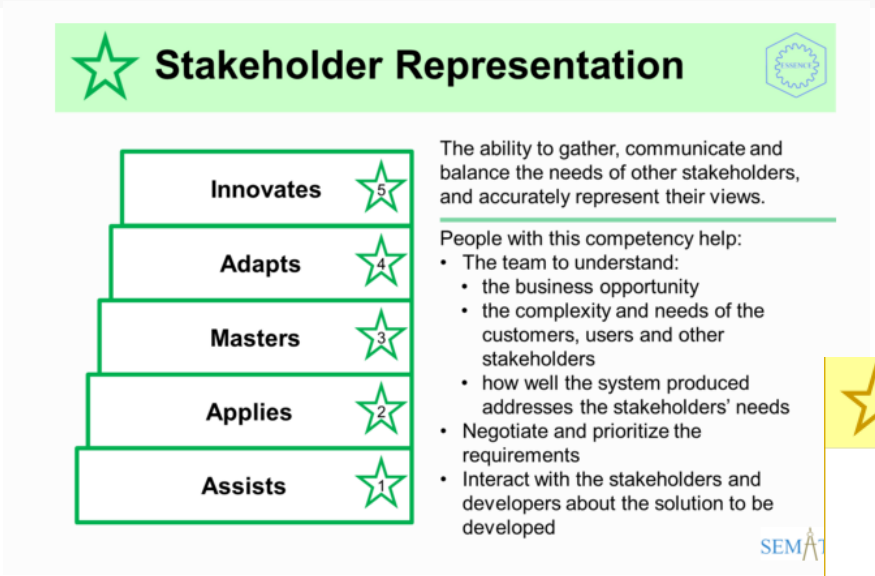
Endeavour Related Practise, Alphas and Activities

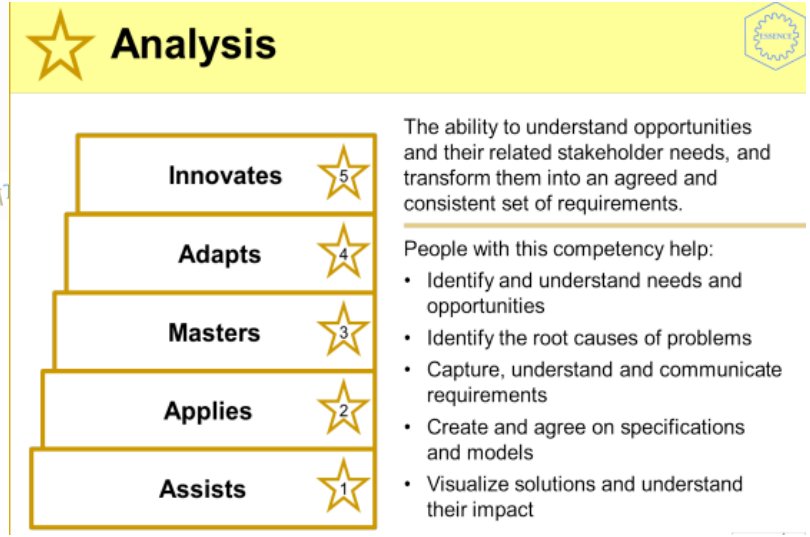


Extend Requirements Card – when need to track items

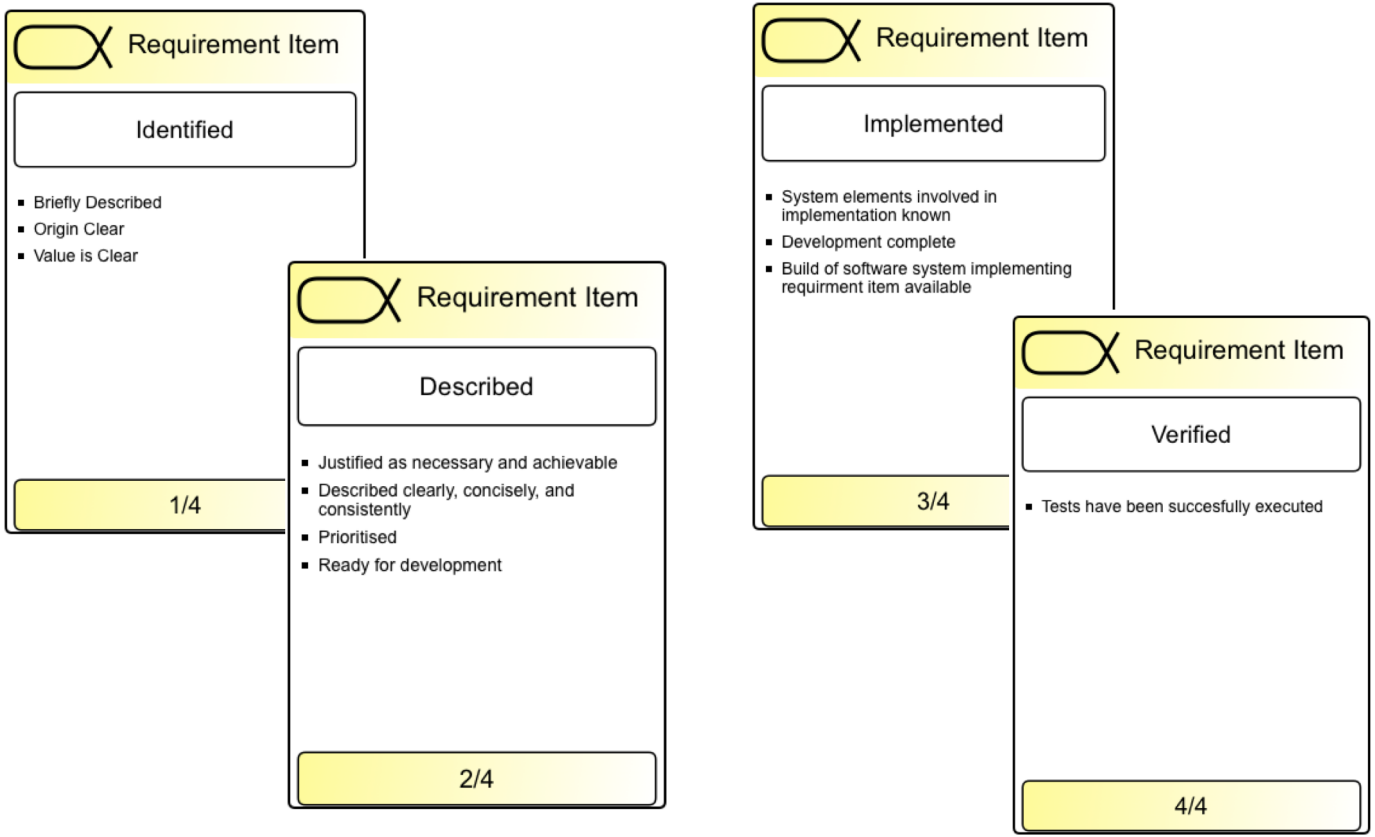


Competency Cards

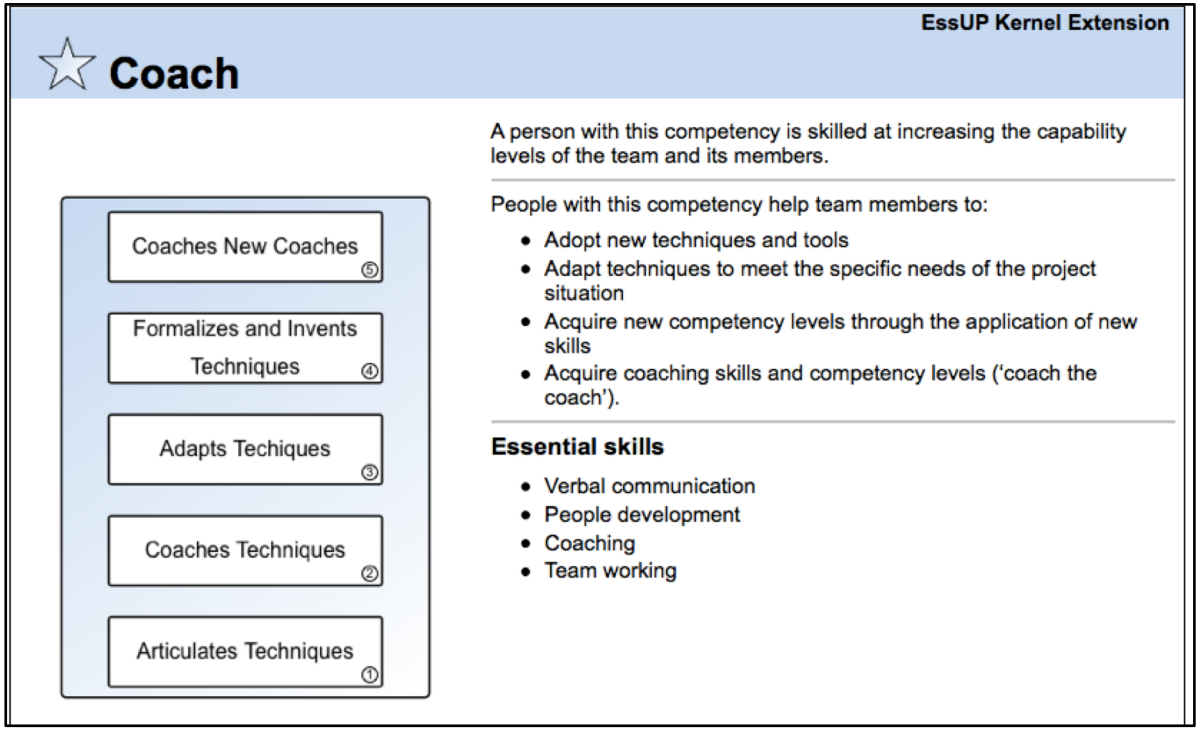




Define each State for Requirements Item

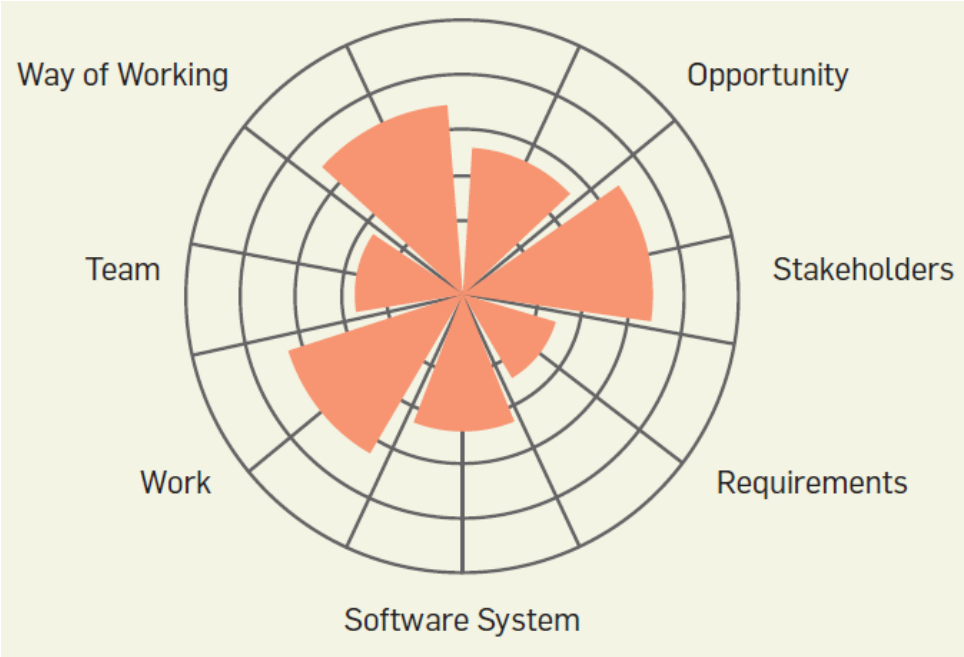


Adding a new Competency



Conclusion

- Process Agnostic
- Start Simple
- New User Experience
  - Simple Card / Checklist Approach.
- Many ways to apply:
  - Review and understand progress.
  - Health Check.
  - Focus and Resolve Pain points.
  - Team Reflection.
- Build as much method as you need:
  - Experience of Team
  - Governance,
  - Formality
- Practise Based
  - Practises are extensible
  - Mix and Match to make own Method
- Can be applied E2E , Infrastructure etc..



Tools:

- Essence Practice WorkBench
- iPhone App: State Explorer

The Death of Agile – 2014, Dave Thomas

He noted that ‘Agile’ is an adjective not a noun.

He said:

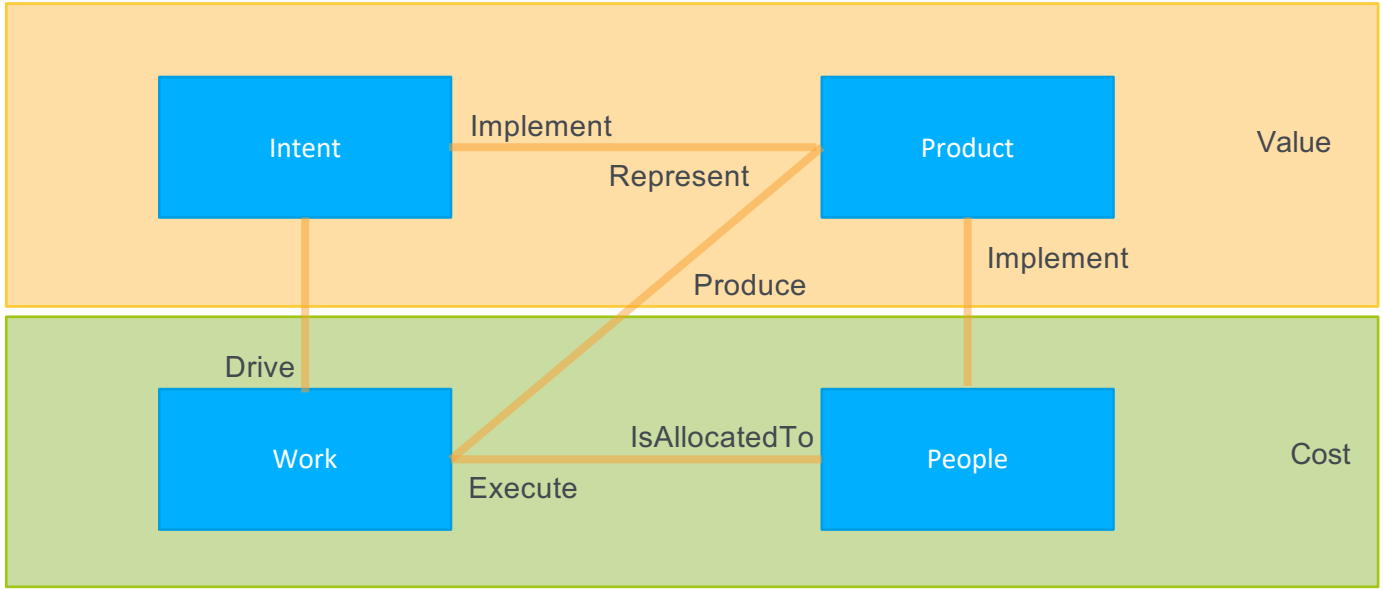
“Here is how to do something in an agile fashion:

- Find out where you are
- Take a small step towards your goal
- Adjust your understanding based on what you learned
- Repeat”

This is the Essence Approach too, find out where you are with basic Kernel, and then take small steps…….

KRUSHEN

4 Concepts in Software Development – a Model



Each has 3 Attributes:

- Time
- Quality
- Risk & Uncertainty

4 Concepts

- **Intent** – what the project is trying to achieve, the intentions of the stakeholders, the vision, requirements, specifications, constraints, tests. Change requests modify the intent.
- **Product** – the outcome of the project, what has been achieved. The software and other artefacts e.g., installers, manuals, training.
- **Work** – The discrepancy between Intent and Product drives a project, the process, activities, tasks and steps.
- **People** – systems development is a human intensive activity

- Maps closely to the Core Elements of SEMAT

SPEM / EPF

SPEM and EPF

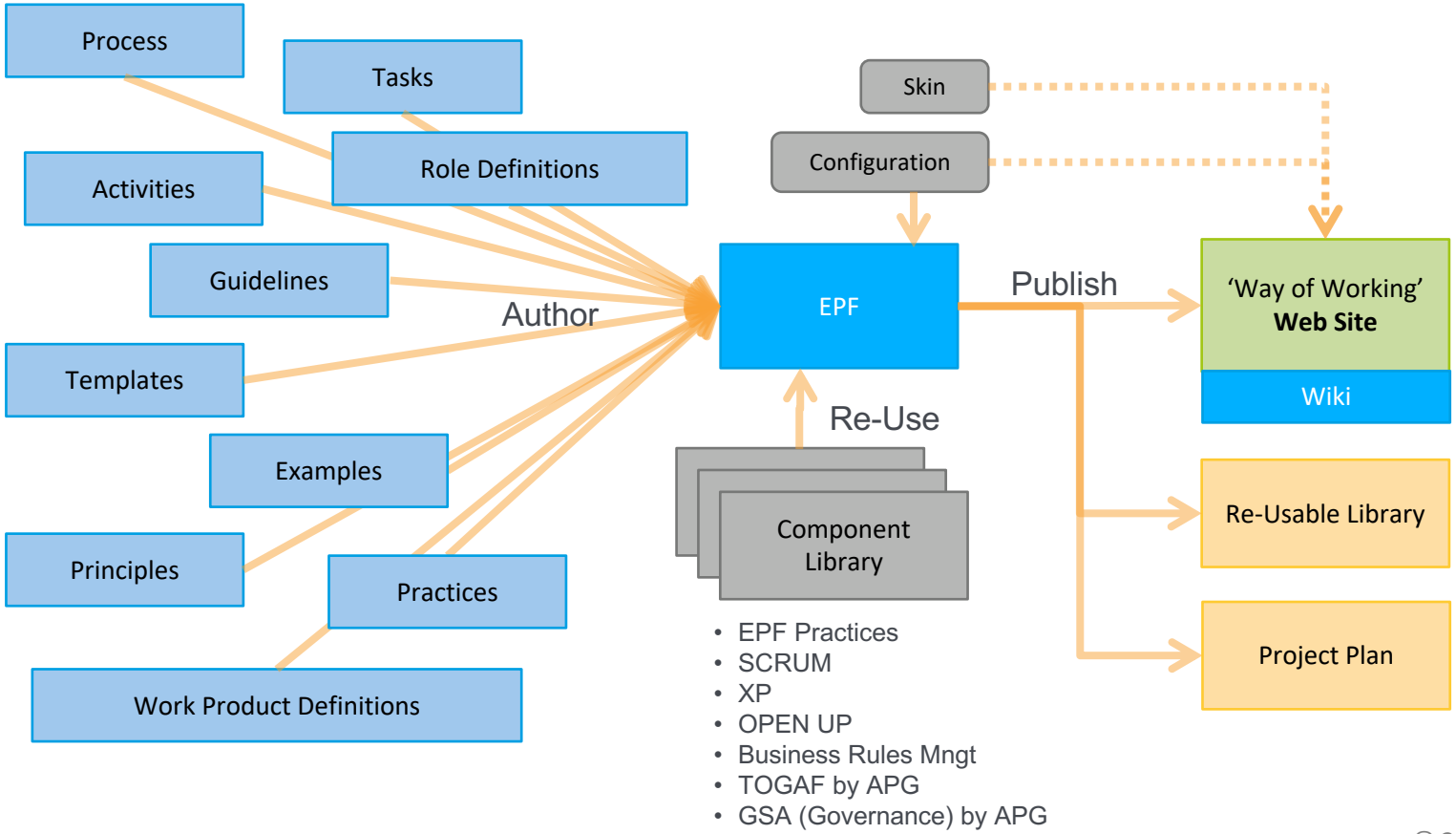
- **SPEM is a Software Process Engineering Model and Framework**
  - The core idea of SPEM is that a development process is a collaboration of multiple process elements to achieve a specific project goal.
  - It is an OMG Standard

- **EPF Composer is a process modelling tool platform, knowledge management tool**, and extensible conceptual framework based on SPEM for authoring, maintaining and customising software development processes.
  - Method content is defined separately from its use in processes
  - Helps author and manage guidance, principles and practices.

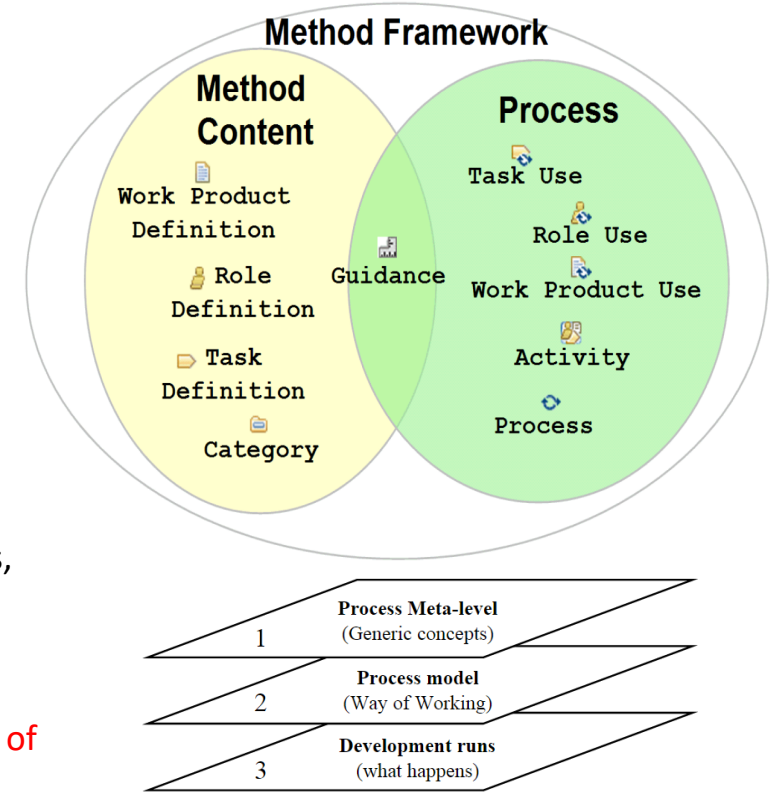
- For us SPEM provides a base model for our way of working, linking both SDLC and Enterprise (e.g. TOGAF is available in SPEM in EPF)

- SPEM also provides the preferred language and taxonomy for our way of working terminology.

EPF Process Composer – Manages Method's Content and Knowledge



SPEM (OMG) Framework – Meta Model



Method Content
(Who, What, Why, How)

- Highly re-useable information content.
- Definition of Roles, Tasks, Work Products and associated relationships.
- Guidance.
- Categories.
- No timing or sequencing of tasks.

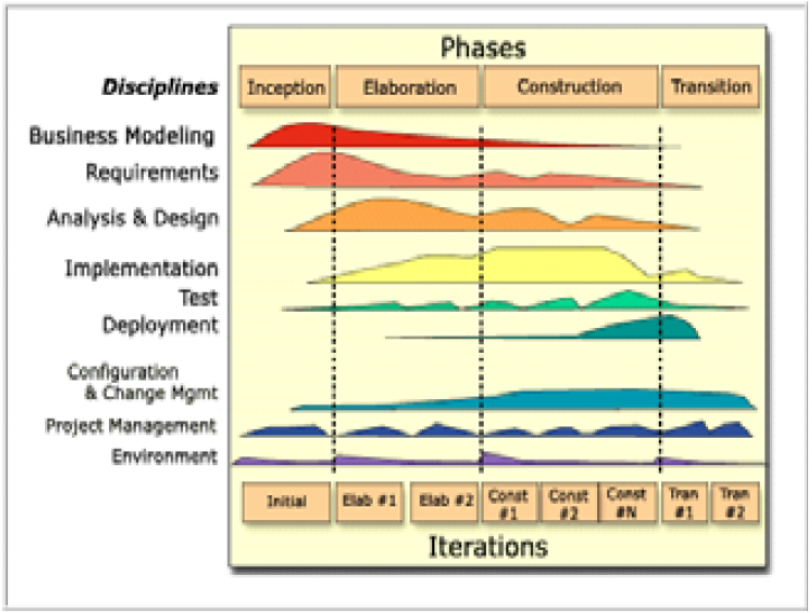
Process (When)

- When Tasks, Roles, Work Products are Used.
- End-End Sequence of Phases, Iterations, Activities and Milestones that define the development lifecycle.
- Define when Tasks are performed via Activity Diagrams and/or Work Breakdown Structures

Content Vs Process – OpenUp Example

Disciplines

Method Content



Process

Phases

Concepts

Lifecycle Structural

- **Phases:** coarse grained planning, major milestones of project
- **Disciplines:** a collection of activities that are related to a major "area of concern"
- **Iterations:** fine grained planning that delivers customer value
- **Increments.**

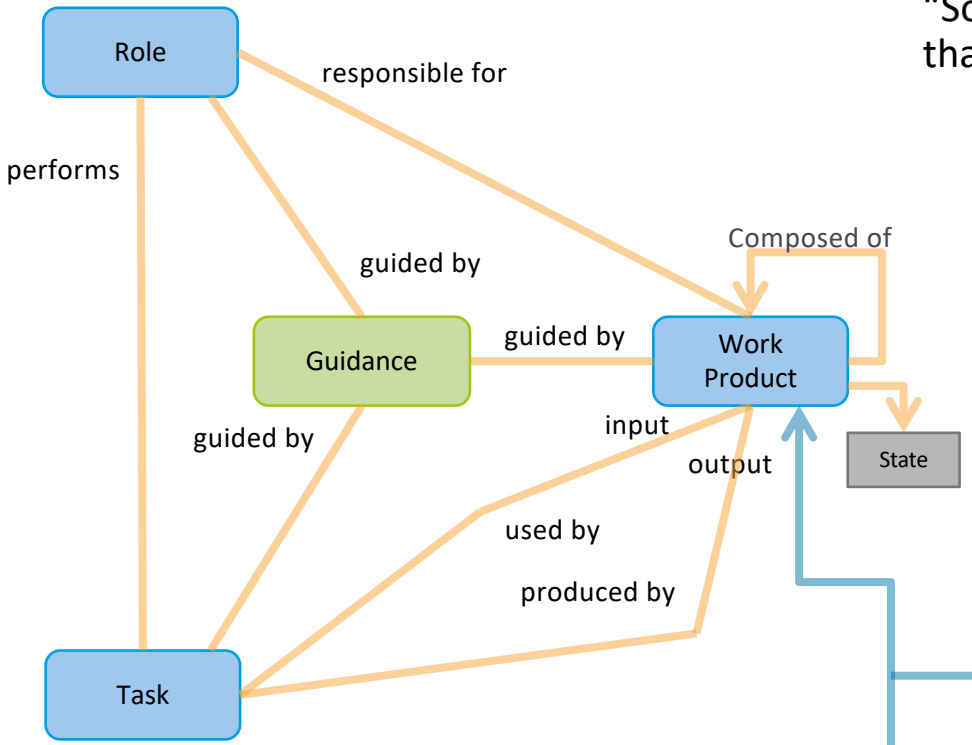
Basic

- **Work Products** what is produced: artefacts, outcomes and deliverables
- **Tasks:** how to perform the work
- **Roles:** who performs the work and for whom
- **Process:** when work gets done, defines workflows and WBS
- **Guidance:** templates, checklists, examples, guidelines etc...

Organising

- **Practices**: a documented approach to solving problems – built from above
- **Capability Patterns:** re-usable chunks of workflow
- **Principles:** the foundation for interpreting roles and work products, and for performing tasks

Work Products, Tasks and Guidance



"Somebody does something that changes something"

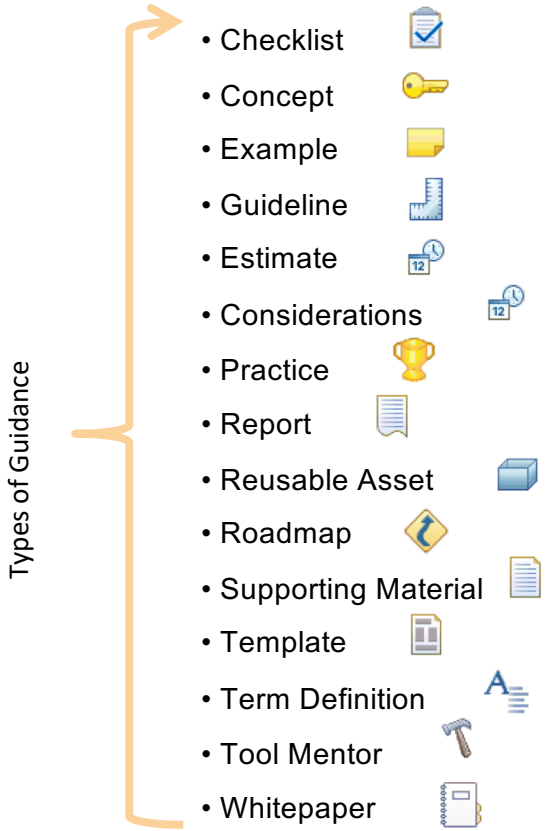
- Roles responsible for work products
  - Each work product responsibility of single role
- Roles perform tasks
  - Each task only performed by single role
- Work products used as inputs to tasks and outputs from tasks
- Roles, Work Products and Tasks are guided by Guidance
- 3 Types of Work Products
- Work Products have State

Artefacts | Deliverables | Outcomes

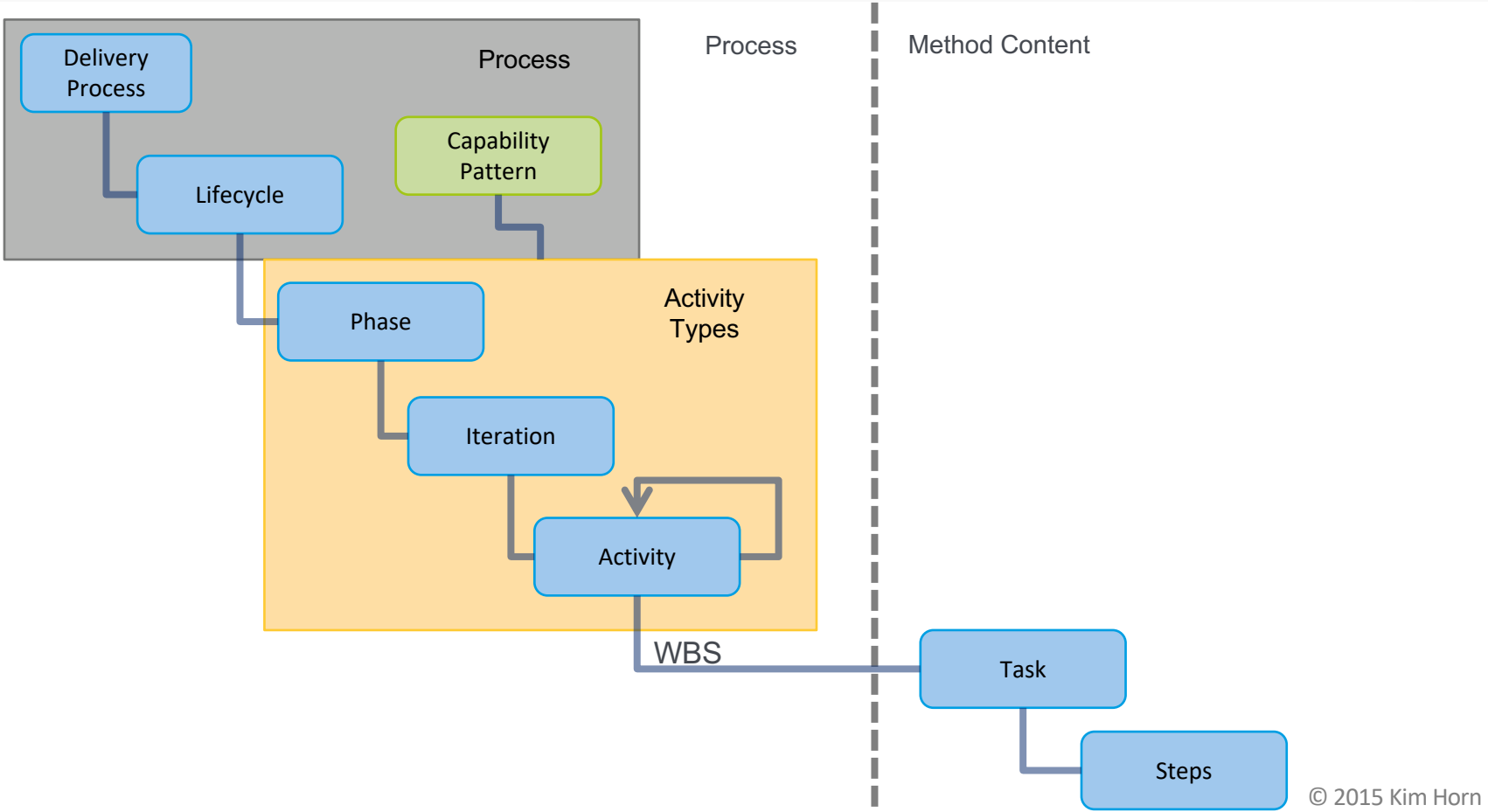
Concept - Guidance

General term referring to all types of material that provide additional detail.

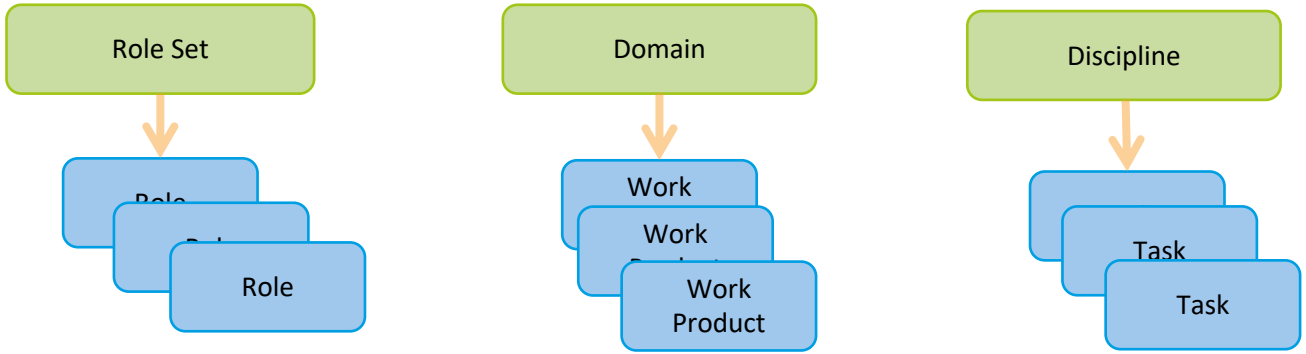
- Guidance may be associated with Roles, Tasks, and Work Products.
- Different types of Guidance depending upon purpose.
- Use Guidance for detailed methodology and supporting information. This will simplify tailoring.
  - Tasks should tell you “what” needs to be done.
  - Guidelines provide detailed “how to”.



Process to Method Hierarchy Model



Structuring and Organising



Domain - Primary categorization mechanism for organizing work products that have an affinity to each other based on resources, timing, relationships or general subject area.

Discipline - Primary categorization mechanism for organizing tasks that define a major 'area of concern' and/or cooperation of work effort.

Definitions

- **Delivery Process -** A delivery process is a special process describing a complete and integrated approach for performing a specific project type. It provides a complete end-to-end lifecycle (for it's scope) and can be used as a reference for running projects with similar characteristics.

- **Capability Pattern** - A special type of process used to define a stereotypical way of performing work related to a particular subject. Capability Patterns are often used as course grained building blocks to assemble delivery processes.

- **Phase -** A specialized type of activity that represents a significant period in a project normally ending with a decision checkpoint, major milestones, or a set of deliverables. Phases typically have well defined objectives and provide the basis for how the project work will be structured.

OpenUP Core Principles

OpenUP Principle	Agile Manifesto Statement
Collaborate to align interests and share understanding	Individuals and interactions over process and tools
Balance competing priorities to maximize stakeholder value	Customer collaboration over contract negotiation
Focus on the architecture early to minimize risks and organize development	Working software over comprehensive documentation
Evolve to continuously obtain feedback and improve	Responding to change over following a plan

OpenUP Practices

Management*

- Iterative Development
- Risk-Value Lifecycle
- Release Planning
- Whole Team
- Team Change Management

*From EPF Agile Kernel

Technical

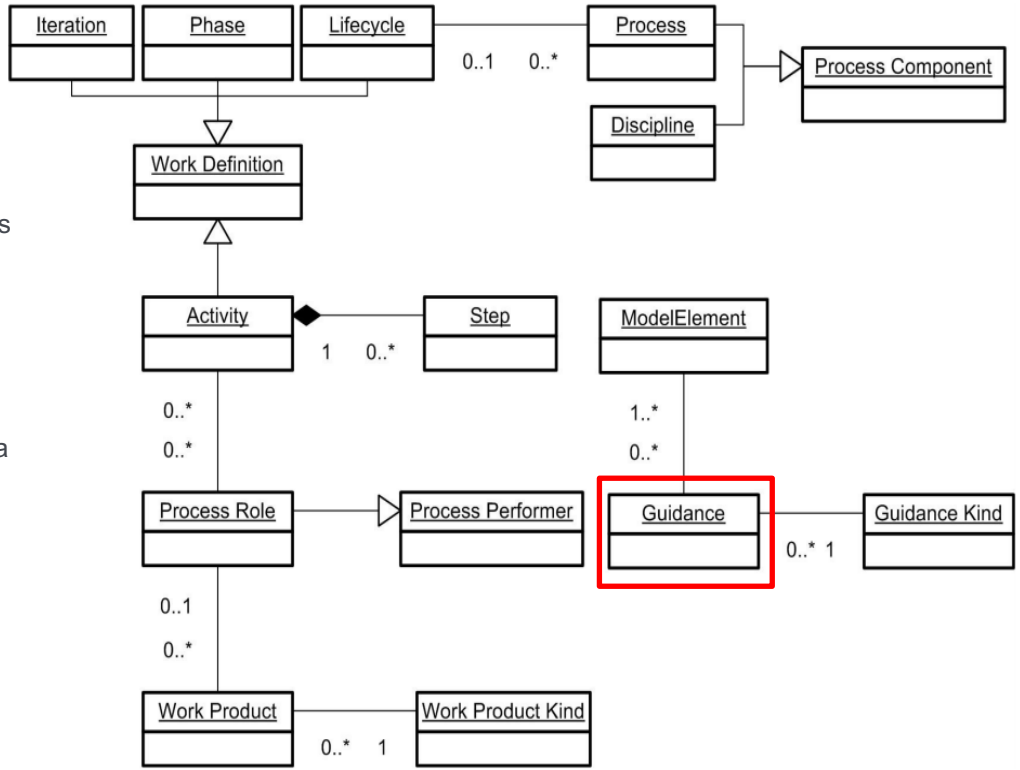
- Concurrent Testing
- Continuous Integration
- Evolutionary Architecture
- Evolutionary Design
- Shared Vision
- Test Driven Development
- Use Case Driven Development

Practices for Scaling Agility

SPEM Meta Model - Simplified

Activities define the process, the sequences of work. Activities reference Tasks as ‘descriptors’.

Tasks are isolated re-usable units of work. They achieve goals, via ‘steps’ that may be optional or sequential. Tasks can be linked to ‘guidance’ elements.



Practise – a type of Guidance

Describes a proven way of doing something or common approaches and strategies that represent best practices. Practices are intended as "chunks" of process for adoption, enablement, and configuration. Also used to represent standards and policies related to methods.

Practices enable a new approach to building methods - practice composition

Benefits:

- Focused on business results
- Reusability, adaptability and scalability
- Incremental adoption
- Easy to configure and use
- Community development

Practise Composition

Focused on Business Results (provided capabilities and resulting work products). They translate ideas into action and deliver high value. Practices are refined based on experiences and lessons learned ("practice makes perfect!"). Practices focus on what matters (e.g., what practices should we use to meet our business objectives vs what process is "best").

- A practice has a ***positive impact on one or several business objectives*** (e.g., Time-to-Market, Improve Quality, Increase Innovation, etc.).
- The adoption of a practice, and it's impact on business objectives, can be effectively ***measured***.

Reusability, Adaptability and Scalability A practice is a ***reusable*** and ***scalable*** process package that may be general, domain-specific, technique-specific, organization-specific, etc. Practices can be extended/enhanced by other practices and/or techniques. Practices can be **adapted** to support a range of solutions. In particular, practices can be adapted to suit your organization and supplemented by your own practices.

Incremental Adoption

- Can be ***adopted independently and incrementally*** by an organization to build an organizational capability. Practices support easier adoption of lighter processes. Organizations only use what they really need. They can adopt one or a few practice at a time and/or adopt a practice at higher levels over time (evolutionary and incremental adoption).
- Incremental adoption is supported by the fact that each practice is described as a standalone capability and provides one-stop-shopping for all collateral related to the practice, e.g. courses, tool features, services, articles, process content, enactment, etc.

Practise Composition

Easy to Configure and Use

- Practices are designed to be ***interchangeable***, they may be mixed and matched or swapped out for alternative practices. Practice-based techniques recognize that "one-size fits all" is too limiting for processes. Practices allow alternatives. Creating a method is as simple as selecting the practices that you wish to adopt, and then publishing the results. Each practice adds itself into the framework so that content can be viewed by practice, or across practices by work product, role, task and so on.

Community Development

- Since a practice can be easily authored on its own, practices are ideal for community development. Practices enable a richer eco-system as it is easier to develop an individual practice than to author an entire method.

Practise Structure

Includes:

- Key Concepts Guidance
- Work Products
- Tasks
- Other Guidance

Work Product Types

- **Artefacts** – provides a description and definition for tangible, non-trivial work products that are consumed, produced, or modified by tasks.
  - May be composed of other work products, e.g. model composed of model elements.
  - Candidates for re-use harvesting.
- **Deliverables** – used to define the primary outputs that represent value, material or otherwise, to the client, customer or other stakeholders.
  - Typically the result of packing other work products for delivery to internal or external parties.
- **Outcomes** – intangible work products that are a result or state, e.g. installed server.
  - Not candidates for re-use.

SPEM Objects – Method Content

Category	Categorize content based on the user's criteria.
Guidance	Identify reference items such as Guidelines, Templates, Checklists, Tool Mentors, Estimates, Supporting Materials, Reports and Concepts.
Metric	Define a standard measurement for instances of Method Content elements.
Role Definition	Define a set of related skills, competencies, and responsibilities.
Step	Represent parts or subunits of a Task Definition.
Task Definition	Describe an assignable unit of work. Every Task Definition is assigned to specific Role Definitions. A Task is associated with input and output Work Products.
Tool Definition	Describe the tools that are recommended or necessary for completing a specific Task.
Work Product Definition	Define any forms of document, report or outcome that are consumed, produced or modified by Tasks.

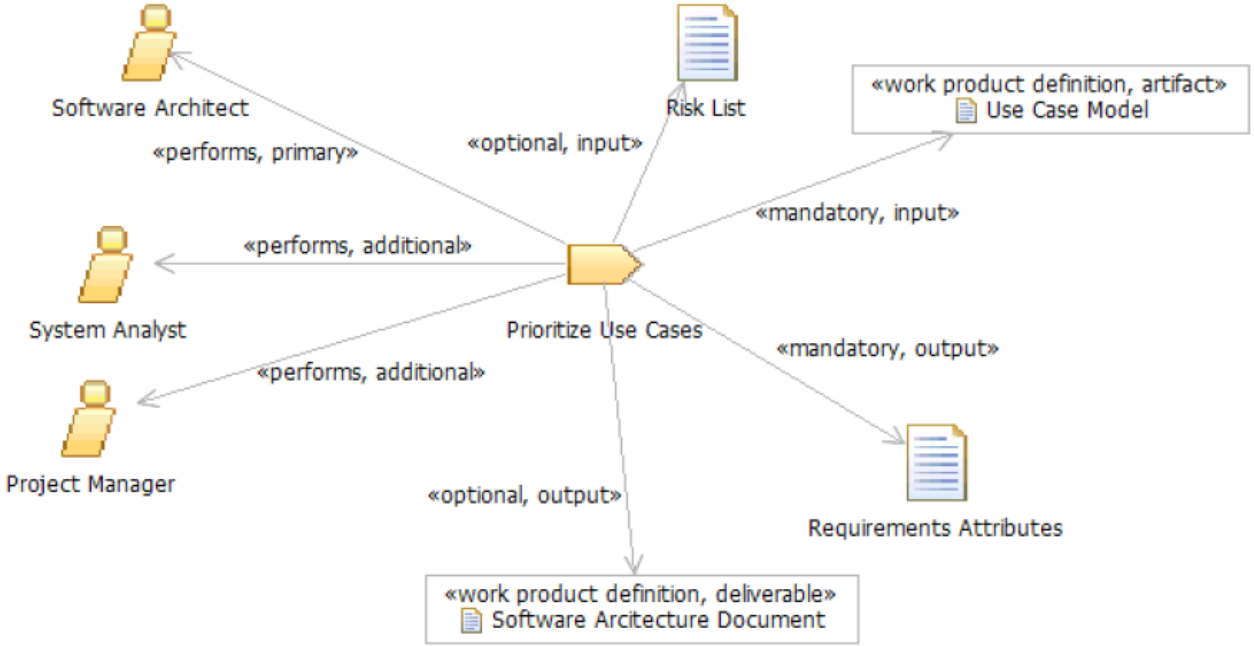
SPEM Objects - Process

Activity	Define basic units of work within a Process as well as the Process itself.
Composite Role	Represent an aggregation of Role Definition references for an Activity.
Milestone	Represent any significant events in a development project.
Process	Create a special Activity that describes a structure for particular types of development project.
Role Use	Represent a Role Definition in the context of one specific Activity.
Task Use	Represent a Task Definition in the context of one specific Activity.
Team Profile	Define a nested hierarchy of teams and team members.
Work Product Use	Represent a Work Product Definition in the context of one specific Activity.

SPEM Objects

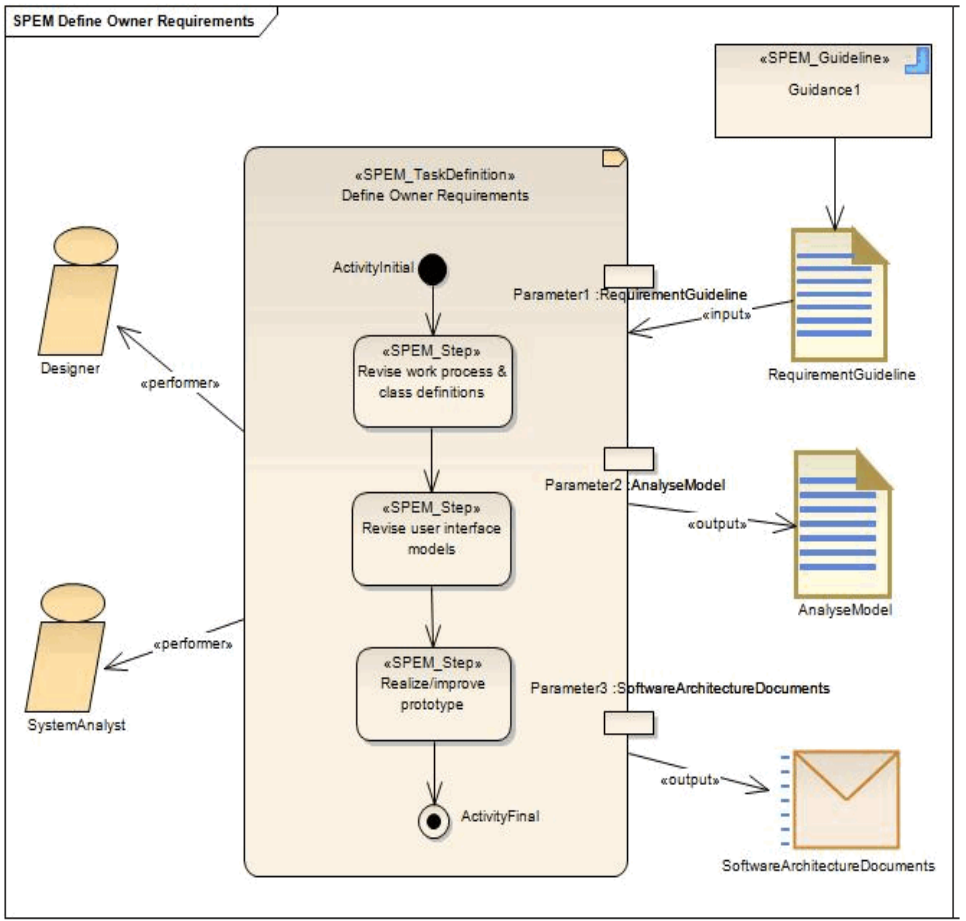
Phase	Create a predefined special Activity representing a significant period in a project.
Iteration	Group a set of nested Activities that are repeated more than once. Typically, Iteration is an Activity for which the default value of the *isRepeatable* attribute is **True**.
Process	Represent a special Activity that describes a structure for particular types of development projects, or parts of them.
Delivery Process	Represent a special Process describing a complete and integrated approach for implementing a specific project type.
Process Pattern	Represent a special Process to describe a reusable cluster of Activities in a general process area that provides a consistent development approach to common problems.
Process Planning Template	Represent a special Process that is prepared for instantiation by a project planning tool.
Artifact	Represent a Work Product Definition that provides a description and definition for tangible work product types
Deliverable	Represent a Work Product Definition that provides a description and definition for packaging other Work Products, and that can be delivered to an internal or external party.
Outcome	Represent a Work Product Definition that provides a description and definition for non-tangible work products.

UML SPEM 2.0 Profile – Example Task

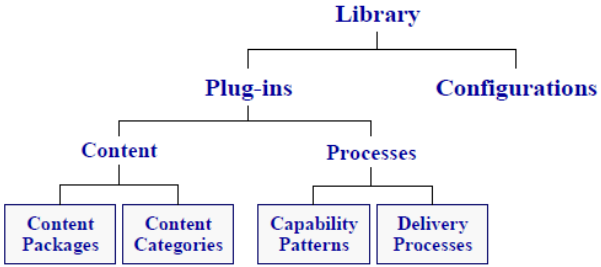


SPEM Diagram

Example From System Architect



Method Content



- The **Method Library** contains Method Plug-ins and Method Configurations.
- All **method content** is organised in Method Plug-ins.
- **Processes** organise method content elements into semi-ordered sequences customised for specific types of projects.
- **A method configuration** is the manifest of method plug-ins used to generate a specific instance of process guidance.

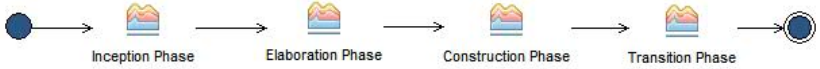
1. Method Plug-ins
   - Method Content
     - Content Packages
       - Roles
       - Tasks
       - Work Products
       - Guidance
     - Standard Categories
       - Disciplines
       - Domains
       - Work Product Kinds
       - Role Sets
       - Tools
     - Custom Categories
   - Processes
     - Capability Patterns
     - Delivery Processes
2. Method Configurations

3 types of Process Diagrams

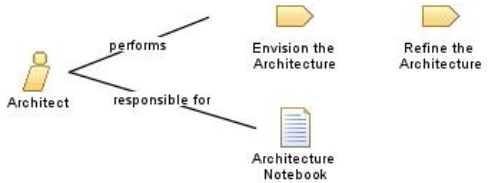
- **Activity diagrams:** These diagrams show the subordinate activities in a higher level activity. They also show the sequence relationships between those activities.
- **Activity detail diagrams:** These diagrams show tasks in an activity with their performing roles along with input and output work products. Activity detail diagrams are similar to workflow detail diagrams.
- **Work product dependency diagrams:** These diagrams illustrate work product dependencies on other work products.

Diagrams

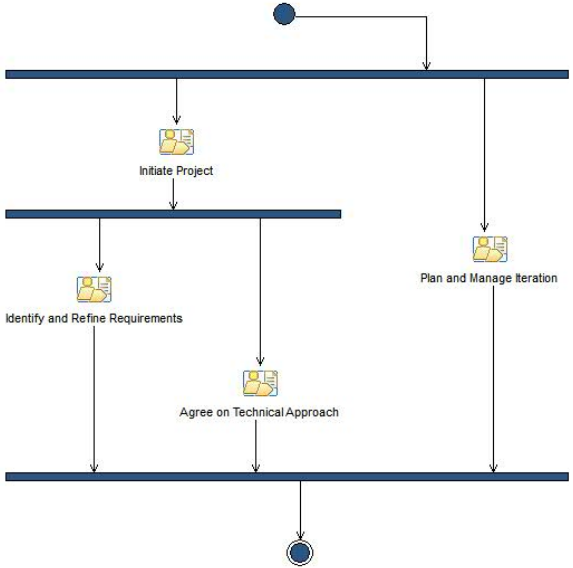
WBS: Workflow – Includes Phases



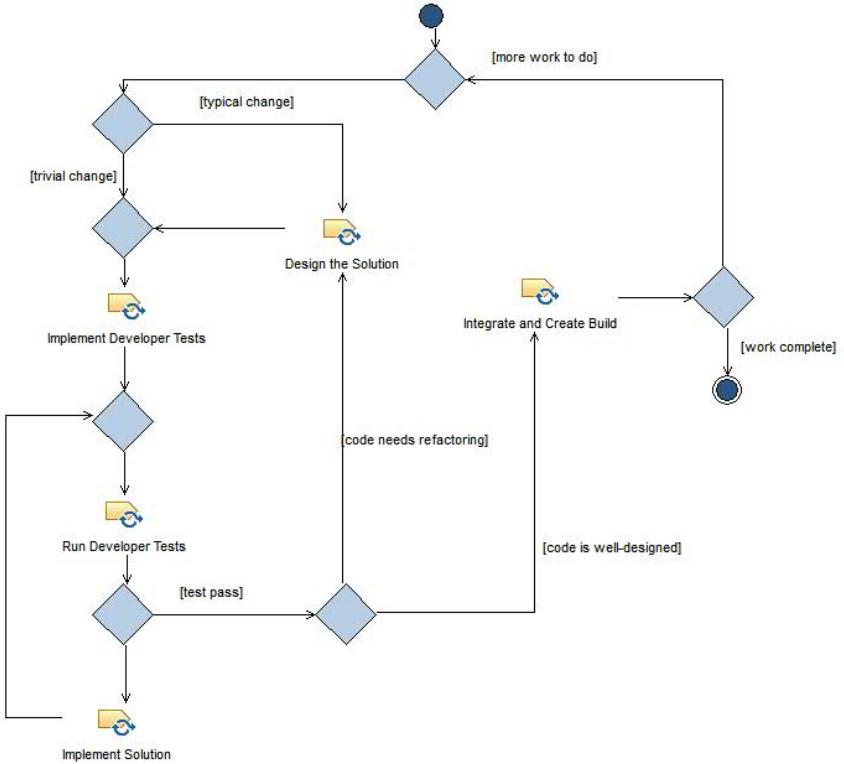
Relationships: Architect



WBS: Workflow – Includes Activities

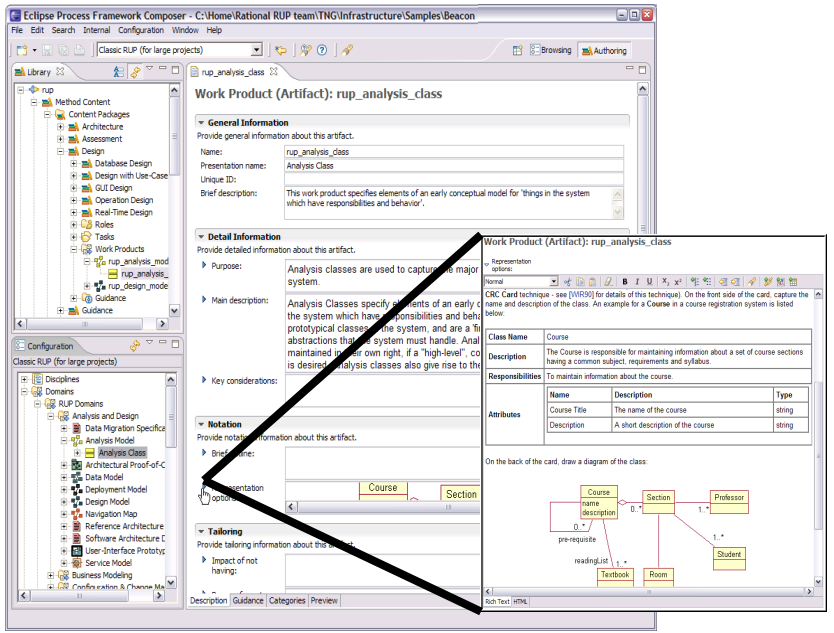


The Activity : Develop Solution Increment

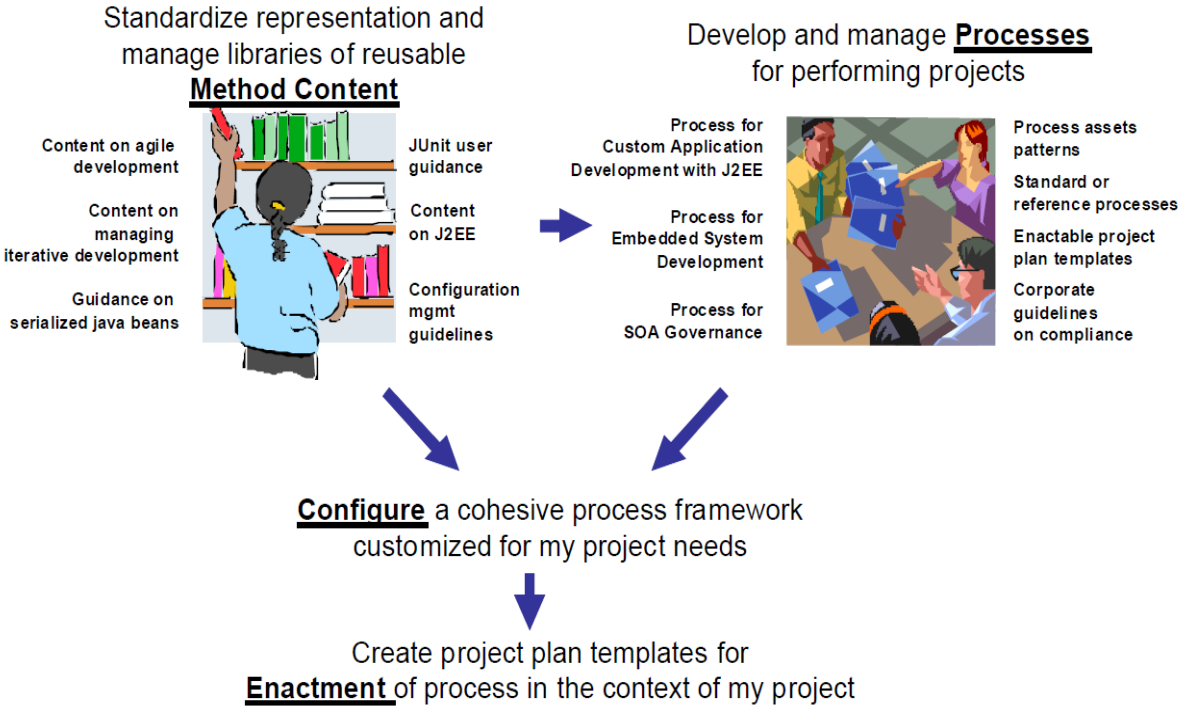


Authoring Tool for Software Processes

- Allows you to author a process then publish a tailored configuration.
- Provides re-usable Library capability.
- Available Libraries include:
  - XP
  - Agile Kernel - Practices
  - SCRUM
  - Business Rules
  - Open UP
  - TOGAF 9, 9.1 – APG Library
  - APTL = Solution Arch and Governance from APG



Conceptual Usage Framework

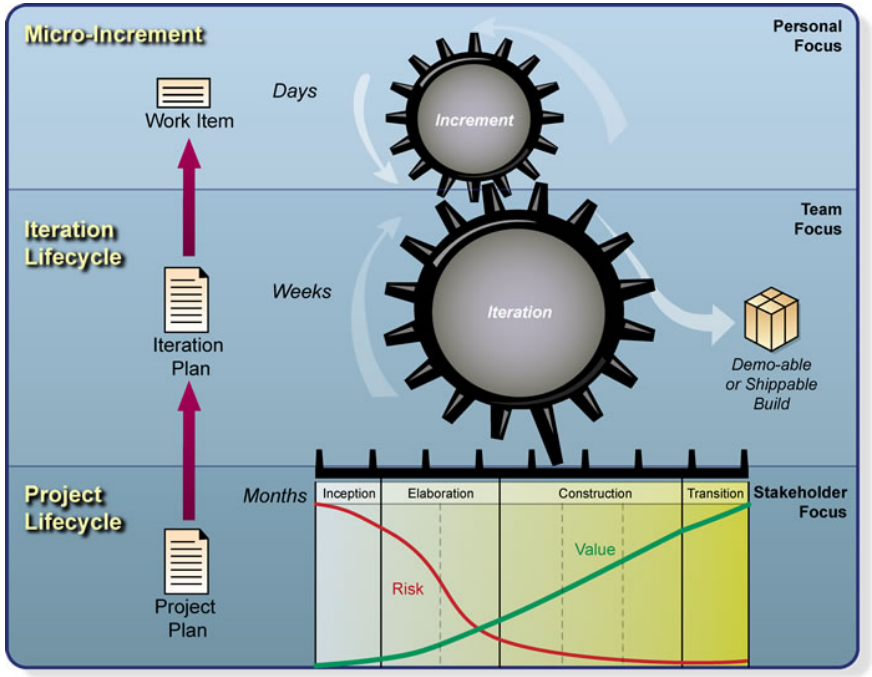


What is Open UP ?

"An Agile Inspired process with its roots in the UP"

An iterative software development process that is minimal, complete, and extensible

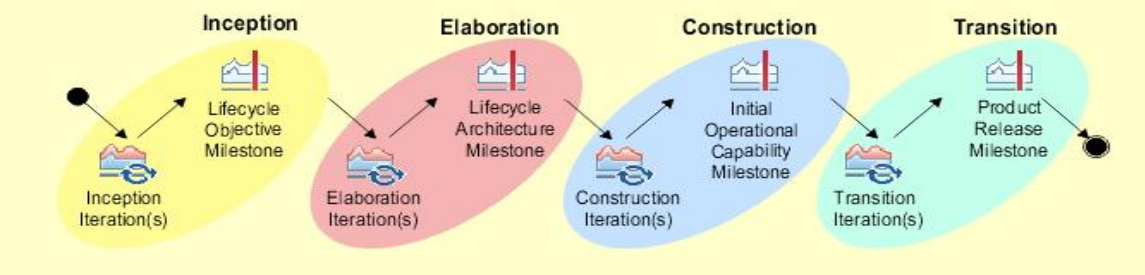
- **Minimal** - Contains only vital roles, tasks and guidance
- **Complete** - Complete for small co-located teams
- **Extensible** - Serves as a foundation that can be extended and tailored



OpenUP Project Lifecycle

- OpenUP uses an iterative, incremental lifecycle.
- The lifecycle is divided into 4 phases, each with a particular purpose and milestone criteria to exit the phase:
  - **Inception**: To understand the problem.
  - **Elaboration**: To validate the solution architecture.
  - **Construction**: To build and verify the solution in increments.
  - **Transition**: To transition the solution to the operational environment and validate the solution.

Add Here



Add Here

Typical Extensions or Tailoring of Phases includes:

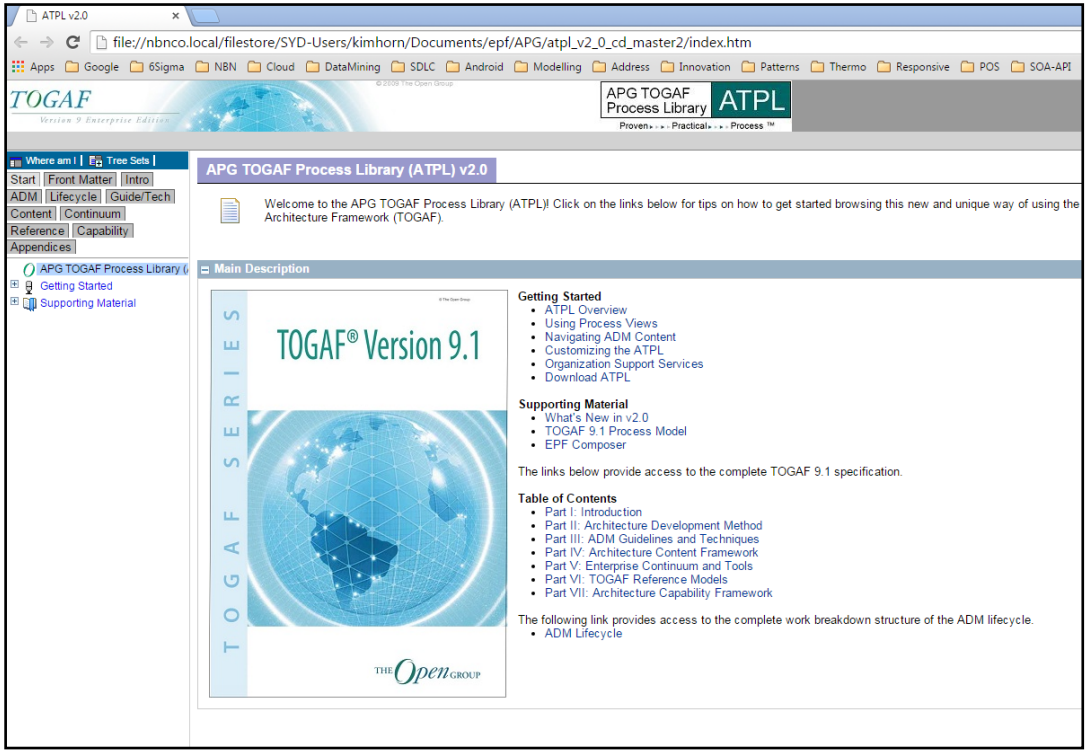
- **Adding phases at Start:** e.g. Enterprise, Sales, Engagement, or Concept
- **Adding phases at End:** e.g. Operation, Production, Support or Retirement

91

APG & GSA

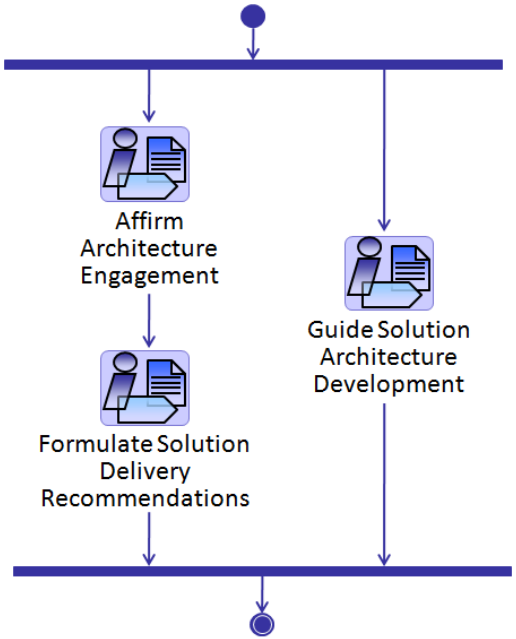
Armstrong Process Group TOGAF 9.1

Provide an EPF - TOGAF Process Library (ATPL)



Armstrong Process Group

- **Provides a Framework to link EA and SDLCs**
- *ATPL+ Govern Solution Architectures (ATPL+GSA)* is a method plugin in the APG TOGAF Process Library (ATPL) that refines the descriptive guidance found in Phase G: Implementation Governance of TOGAF.
- Implementation governance is where the enterprise architecture (EA) lifecycle touches the solution delivery lifecycle as projects are launched to implement the target architecture.
- ATPL+GSA specifies an overall workflow for implementation governance that includes three activities:
  - Affirm Architecture Engagement
  - Formulate Solution Delivery Recommendations
  - Guide Solution Architecture Development
- ATPL+GSA identifies three major new deliverables (and over a dozen additional artifacts) including:
  - Solution Architecture Contract
  - Solution Delivery Recommendations
  - Architecture Review Report



ATPL Solution and Governance Material

Solution architecture context

- Describes EA context for project – the "blueprint"
- Potentially one for each EA domain

Architecture assets

- Identifies existing or new EA assets project will use, create, or update

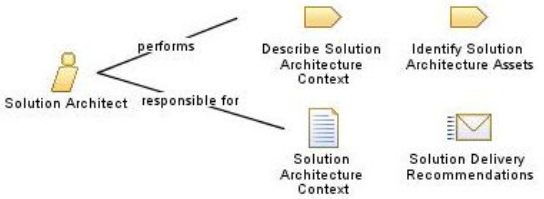
Architecture requirements

- Describes which EA requirements project is responsible for
- Business and technical
- Also includes architecture constraints

Conformance requirements

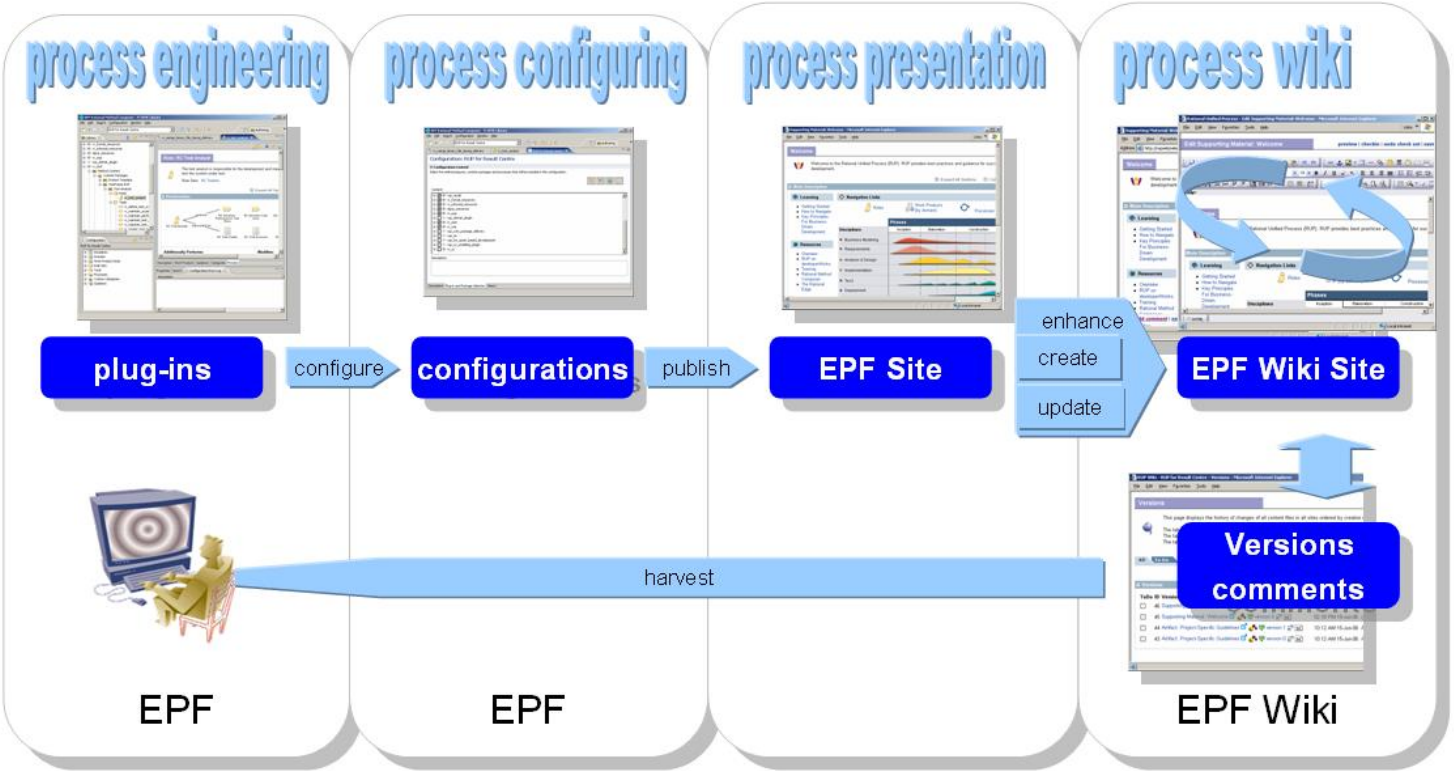
- Describes governance and compliance requirements project is subject to

Relationships



Additionally Performs	• Approve Architecture Engagement • Conduct Architecture Review • Describe Architecture Roles and Responsibilities • Describe Conformance Requirements • Identify Migration Plan Milestones • Identify Solution Architecture Requirements • Prepare for Architecture Review • Present Architecture Review Report
Modifies	• Solution Architecture Context • Architecture Asset

EPF Composer and EPF Wiki

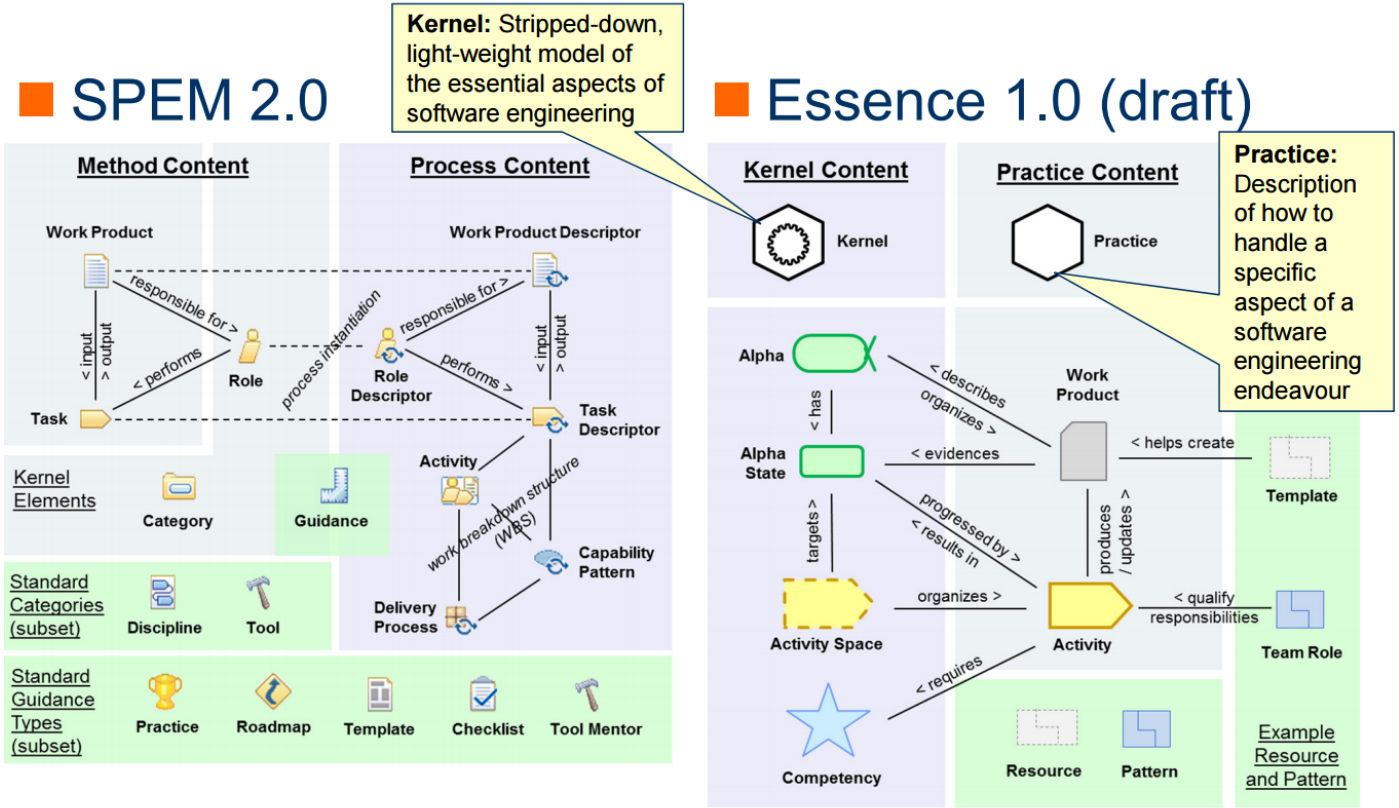


The End

SPEM and ESSENCE

SPEM	Essence	Comment
Task	Activity	Similar in concepts, but an *Activity* in Essence additionally defines *completion criteria*.
Work Product	Work Product	Similar in concepts, but a *Work Product* in Essence additionally defines *levels of details* (not discussed in the illustrative example) to specify different ranges of details required.
Role	n/a	*Role* allows to author responsibilities of work products and performers of tasks in SPEM. Essence does not specify the concept of a role and advices to use the *Pattern* concept to define roles. The rationale is that definitions of roles may differ from method to method.
Category	Tag	*Categories* (both default and custom) are used to structure the elements of SPEM content, e.g. to support navigability in the published guidance. Essence provides the concept of a *Tag* (not discussed in the illustrative example) that allows adding user defined or tool specific information.
Guidance	all elements	The Essence language does not differentiate between a specific guidance concept and other concepts such as *Activity* (which basically contains guidance for performing the work).
Practice	Practice	SPEM defines *Practice* as a specific type of *Guidance* that can reference any element defined in SPEM. Essence on the other hand introduces Practice as a specific concept in the language which allows more operational semantics to support composition etc. to be formalized and better supported by tooling.
…	Resource or Pattern	If additional guidance types are needed the advice is to use *Patterns* or *Resources*.
Activity Capability Pattern Delivery Process	Activity Space and Activity	The *Activity*, *Capability Pattern* and Delivery Process concepts in SPEM allows to define work breakdown structures and reusable process content. The Essence language allows *activity associations* between abstract activities (which are either Activity Spaces or Activities) to represent a relationship or dependency between activities.
n/a	Alpha	The *Alpha* concept does not have an equivalent in the SPEM 2.0 specification, and is essential to the underlying method architecture of the Essence language to monitor, track and drive method enactment based on alpha states and completion criteria.

SPEM and Essence



Conclusion

- SPEM and ESSENSE provide the preferred language and taxonomy for our process work.
- EPF and the ESSENSE tools provide a way to document and implement our process.
- TOGAF can be used at the EA level and the ATPL+GSA libraries extend this down to the Solution Architecture level with full Governance of the implementations.

Issues with Processes in IT

The Problem of Denied Commonality

There are many IT SDLCs promoted in the software industry. Each promotes specific advantages over others. However, there is often more in common than not, why is this not apparent:

- Focus is on what's new and novel.
- The stuff in common gets new names and jargon, re-branded. Eventually a new language appears masking the commonality, making the new process appear completely novel.
- Debate centres on the subtle differences. *E.G. User Stories Vs Use Cases. Focusing on the differences ignores that both are User focused.*

There is more similarity than differences

The Problem of Creating a Complete Process

Each process wants to describe the complete picture; solve the whole SDLC problem.

- Its hard not to resist doing it all. A new issue results in adding to the process, rather than restricting the applicability of the process. *E.G. User stories cover Functional User Requirements but can also be used to cover Non-Functional, constraints and all other forms of requirements.*

- Completeness results in heaviness; as more and more stuff is added, with smaller and smaller benefit, taking longer and longer. The problem is nothing is taken out so it gets bigger and more complex, more tightly coupled.

- Desire to keep the process a market leader and address new markets. E.g. a Waterfall process is changed to also cover Agile.

- In practise people then need to cut out what's not required to make it work, lighter. However, due to the tight coupling this is very difficult.

Its better to build up from a small set, than try to lighten a large process.

The Problem of Adopting a Complete / One Size Fits All Process

Adopting a Complete Process:

- Every team has their own way of working, and adopting a new complete process means changing everything when only some practises may need improving.

Rather than starting from scratch, what's needed is a way to improve parts at a time, such as a single practise one at a time.

On size fits all:

- Teams work in different ways.
- Projects are different sizes.
- Different levels of formality (Agile to Heavy).
- Different Governance requirements, Defence, legal, HIPPA.
- Delivery is different; CI flow to Single Delivery.

Start from a small core and tailor to fit the problem at hand.

The Problem of Out-Of-Sync Processes

What the team does never matches the process. Everyone complains your not following the process. However, the point of software development is to develop good software, not to slavishly follow a predefined process.

Project- Process Gap. Process descriptions as are never kept up to date and people apply (practise) their own mix of expertise. Issues are:

- Successes are hard to replicate as not due to described process.
- Planned and monitored process does not match reality. Objectively understanding the process and thus improve its design, re-design and execution becomes difficult.
- Process assessments create the illusion of process conformance.
- Tools don't work, as they are for the process description not the applied process (practise). They don't get used or conversely become more important than the task.

Process should describe what is actually done, not what people think the team ought to do.

The Problem of Acquiring Process Knowledge

How do you communicate to Stakeholders the ‘to be used’ process?
Law of Nature, no one reads the manual.

- Process described as static books and web sites just doesn’t do it.
- Consequence is that it doesn’t make sense writing long and lengthy process descriptions.
- In reality people gain knowledge from in situ experience, and using a process in anger on real projects.

Example: Agile methods have been designed to rely on tacit knowledge instead of explicit knowledge; However the number of agile books is massive and agile experts are still writing more and more of them.

- Mentoring on real projects is often better than courses and manuals.
- Research suggests the value of networks to facilitate knowledge sharing, for example, create a community of practice.
- Light weight and simple material is needed, such as check lists and simple guidance cards, such as adopted by Essence.

The Problem of Acquiring Process Knowledge

There will be a gap between the process designers and performers.

Much of the process requires 'knowledge' to achieve a common understanding of the intent behind the codified process.

So along with the process creation is a knowledge management exercise.

The other issue is the difficulty of creating process and acquiring the 'art' or tacit knowledge that people practice. There may be large amounts of uncodified "know-how" that is not simple enough to be codified in process.

Decisions can be routine, informed or judgemental. Trying to codify Judgements in process will results in process that fail, as considerable experience and expertise is required. Informed decisions may be complex and be expensive to codify.

The Problem of Stupid Processes

- The Process you have to follow doesn't help you do your job; everyone knows it, no one follows it, and everyone just works around it.
- The practise knows the process is broken.
- Process material is very detailed, step by step instructions, and is not practical to follow in the heat of the moment

The Problem of Organisation processes that do not work together.

- Ubiquitous Language: Different areas of the business have their own knowledge and terms.
- Processes developed in each lines of business have different methods, standards and language to other areas.
- For example in IT, the Enterprise Architecture process, such as TOGAF, is not aligned with the project SDLC, such as Agile.
- The different processes in the business do not align or work together.
- Also the issue of what process is actually driving; is there one process or many independent processes for each business area.
- Across the organisation many different approaches may be used to build a process; that have different flavour. When they are all assembled they all look different.

The Problem of No process for building process.

A common and well define approach is required to build process. Yet many organisations stumble at the simplest part;

- How do we build a process; what are the steps and guidelines ?
- What is the agreed single definition of ‘Process’ ?
- Is there are clear view on Process Vs Practise ?
- In the same way processes are hard to describe so are the process used to build processes.
- There are many methodologies and approaches for building process. Which do we use ?

The Problem of Scalability and Agility

Processes work on a small scale but not when scaled up.

The coaches and mentors are overrun with trivial work, getting people up to speed, and no time to cover real problems.

When a process is put in place and the business changes, people still follow the old process.

- Processes in the heads of a few cannot scale.
- Processes that are not described cannot scale.
- Catch 22 - describing processes does not scale, and the description is always out of date.

The process needs to play an actively role, finding thinks, checking and doing trivial tasks.