INF5120
”Modellbasert Systemutvikling”
”Modelbased System development”

Lecture 7: 27.02.2017

Arne-Jørgen Berre

arneb@ifi.uio.no or Arne.J.Berre@sintef.no

Course parts (16 lectures) - 2017

- January (1-3) (Introduction to Modeling, Business Architecture and the Smart Building project):
- 1-16/1: Introduction to INF5120
- 2-23/1: Modeling structure and behaviour (UML and UML 2.0 and metamodeling) - (establish Oblig groups)
- 3-30/1: WebRatio for Web Apps/Portals and Mobile Apps – and Entity/Class modeling – (Getting started with WebRatio)

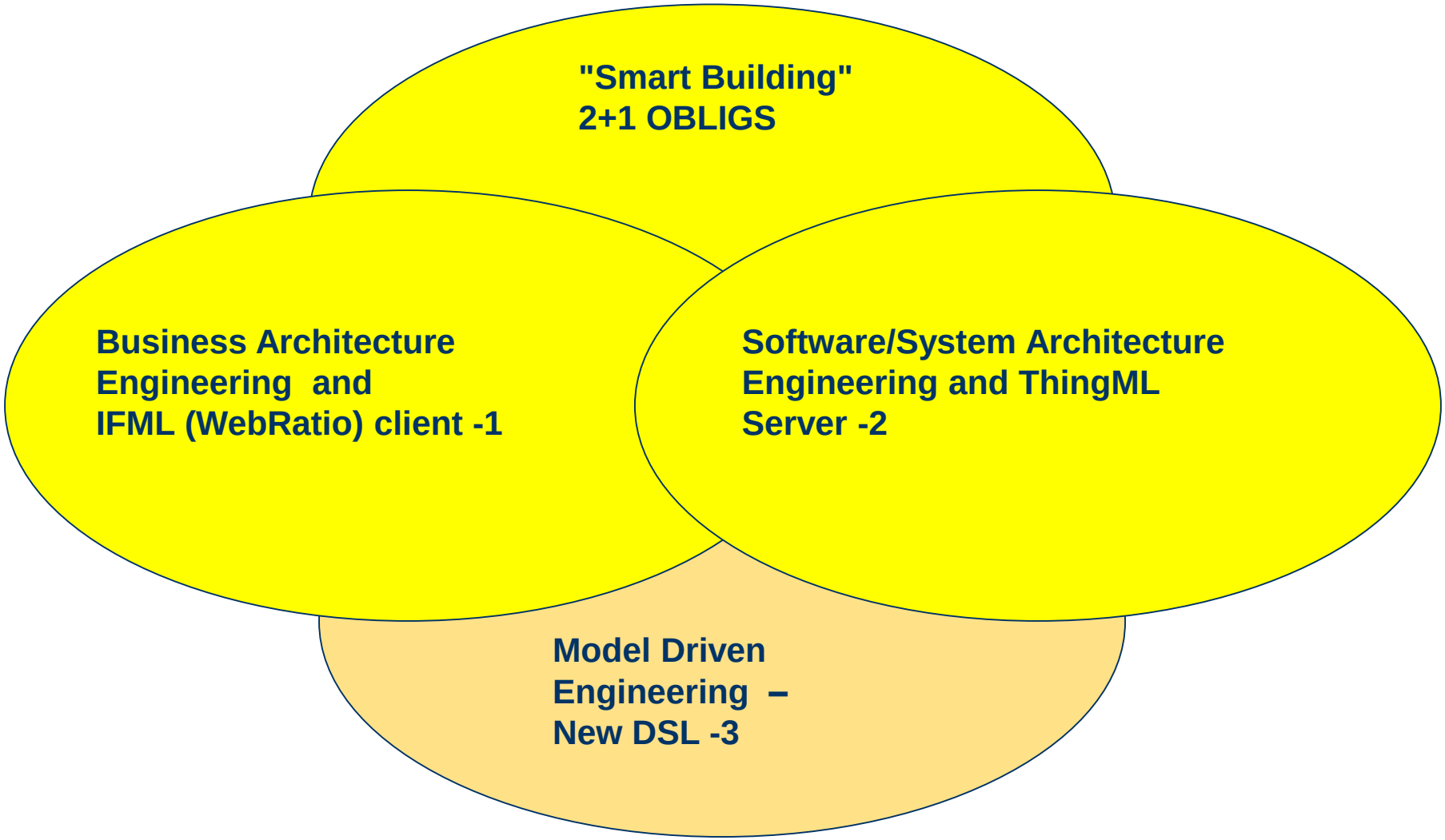
- February (4-7) (Modeling of User Interfaces, Flows and Data model diagrams, Apps/Web Portals - IFML/Client-Side):
- 4-6/2: Business Model Canvas, Value Proposition, Lean Canvas and Essence
- 5-13/2: IFML – Interaction Flow Modeling Language, WebRatio advanced – for Web and Apps
- 6-20/2: BPMN process, UML Activ.Diagrams, Workflow and Orchestration modelling value networks
- 7-27/2: Modeling principles – Quality in Models
- 27/2: Oblig 1: Smart Building – Business Architecture and App/Portal with IFML WebRatio UI for Smart Building

- March (8-11) (Modeling of IoT/CPS/Cloud, Services and Big Data – UML SM/SD/Collab, ThingML Server-Side):
- 8-6/3: DSL and ThingML, UML State Machines and Sequence Diagrams
- 9-13/3: UML Composite structures, State Machines and Sequence Diagrams II
- 10-20/3: Architectural models, Role modeling and UML Collaboration diagrams
- 11-27/3: UML Service Modeling, ServiceML,SoaML, REST, UML 2.0 Composition, MagicDraw
- 27/3: Oblig 2: Smart Building – Internet of Things control with ThingML – Raspberry Pi, Wireless sensors (temperature, humidity), actuators (power control)

- April/May (12-14) (MDE – Creating Your own Domain Specific Language):
- 12-3/4: Model driven engineering – Metamodels, DSL, UML Profiles, EMF, Sirius Editors
- EASTER – 10/4 og 17/4
- 13-24/4: MDE transformations, Non Functional requirements
- 1. Mai – Official holiday
- 14-8/5: Enterprise Architecture, TOGAF, UPDM, SysML – DSLs etc.
- 8/5: Oblig 3 - Your own Domain Specific Language

- May (15-17): (Bringing it together)
- 15-15/5: Summary of the course – Final demonstrations
- 16-22/5: Previous exams – group collaborations (No lecture)
- 17-29/5: Conclusions, Preparations for the Exam by old exams
- June (Exam)
- 13/6: Exam (4 hours), (June 13th, 0900)-1300

Course components



Manifesto for Agile Software Development

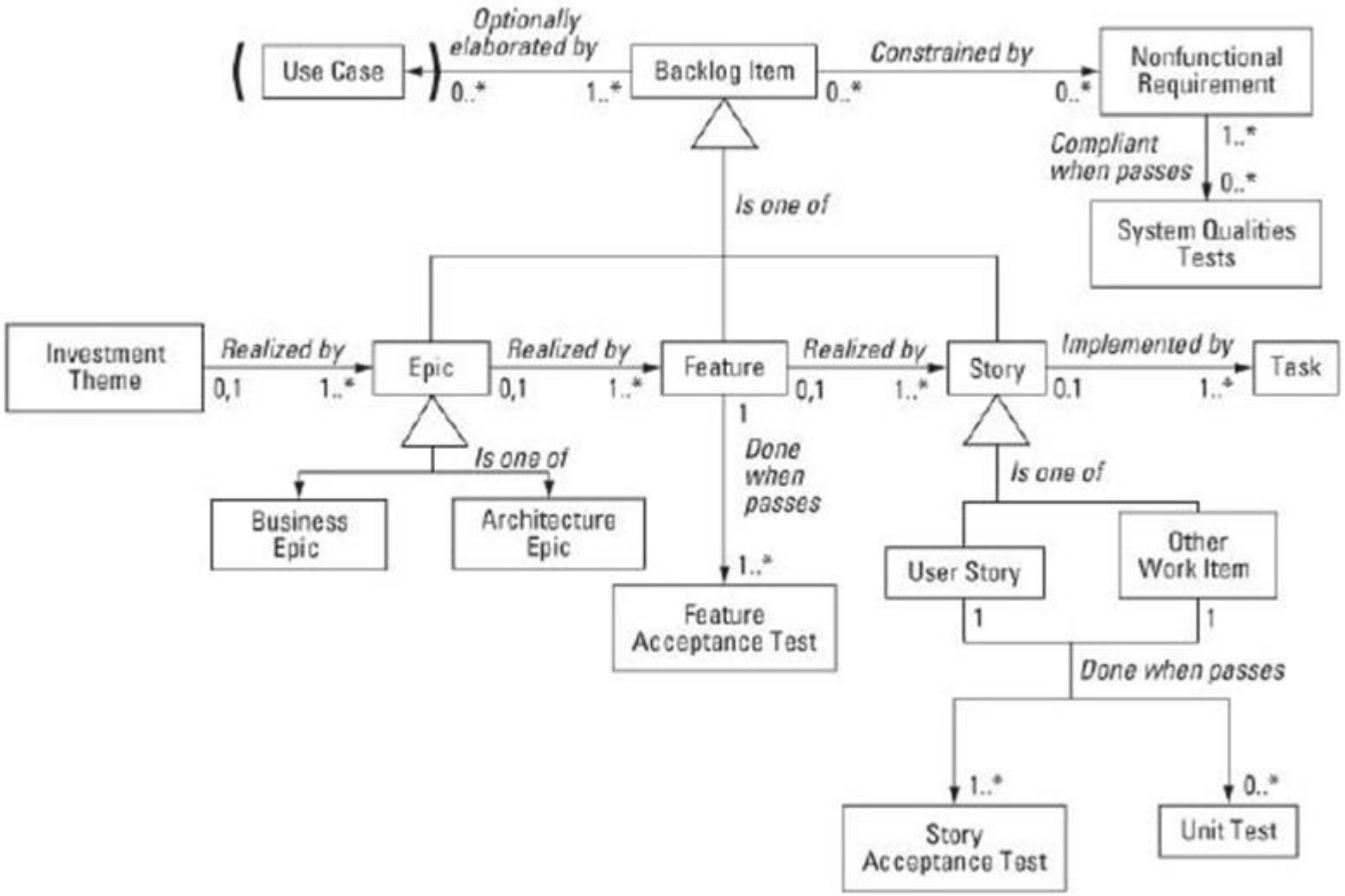
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

http://agilemanifesto.org/

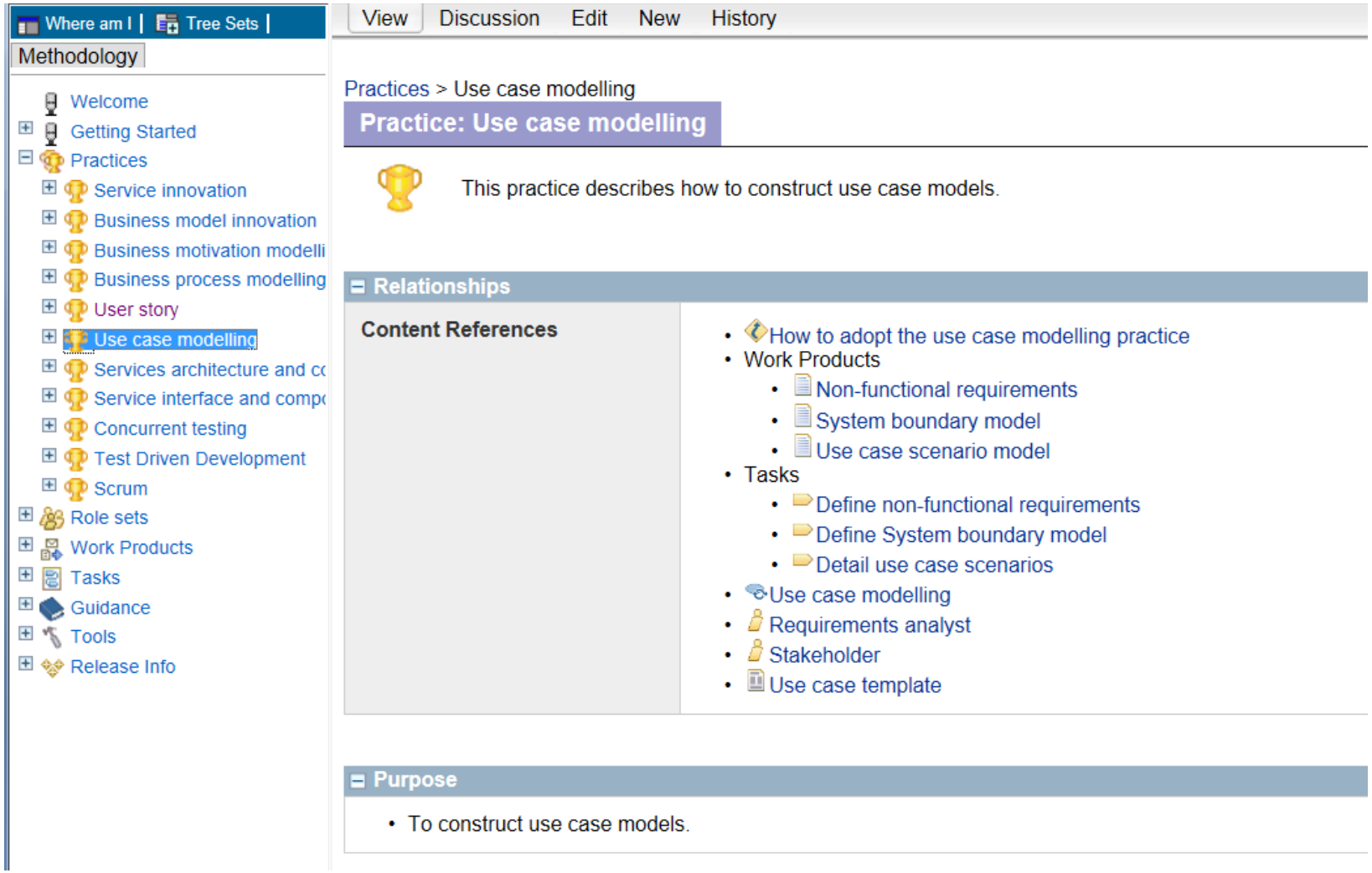
User Story template

- I <in the role of XX> needs functionality <zzz> to achieve the goal of <YYY>

Backlog metamodel



Inf5120.modelbased.net



Practices > User story

Practice: User story

This practice describes how to capture, define and manage user requirements written a

Relationships

Content References

- How to adopt the User story practice
- Work Products
  - Epics
  - Themes
  - User stories
- Tasks
  - Estimate user stories.
  - Prioritize user stories
  - Write user stories
- User story
- Stakeholder
- System developer

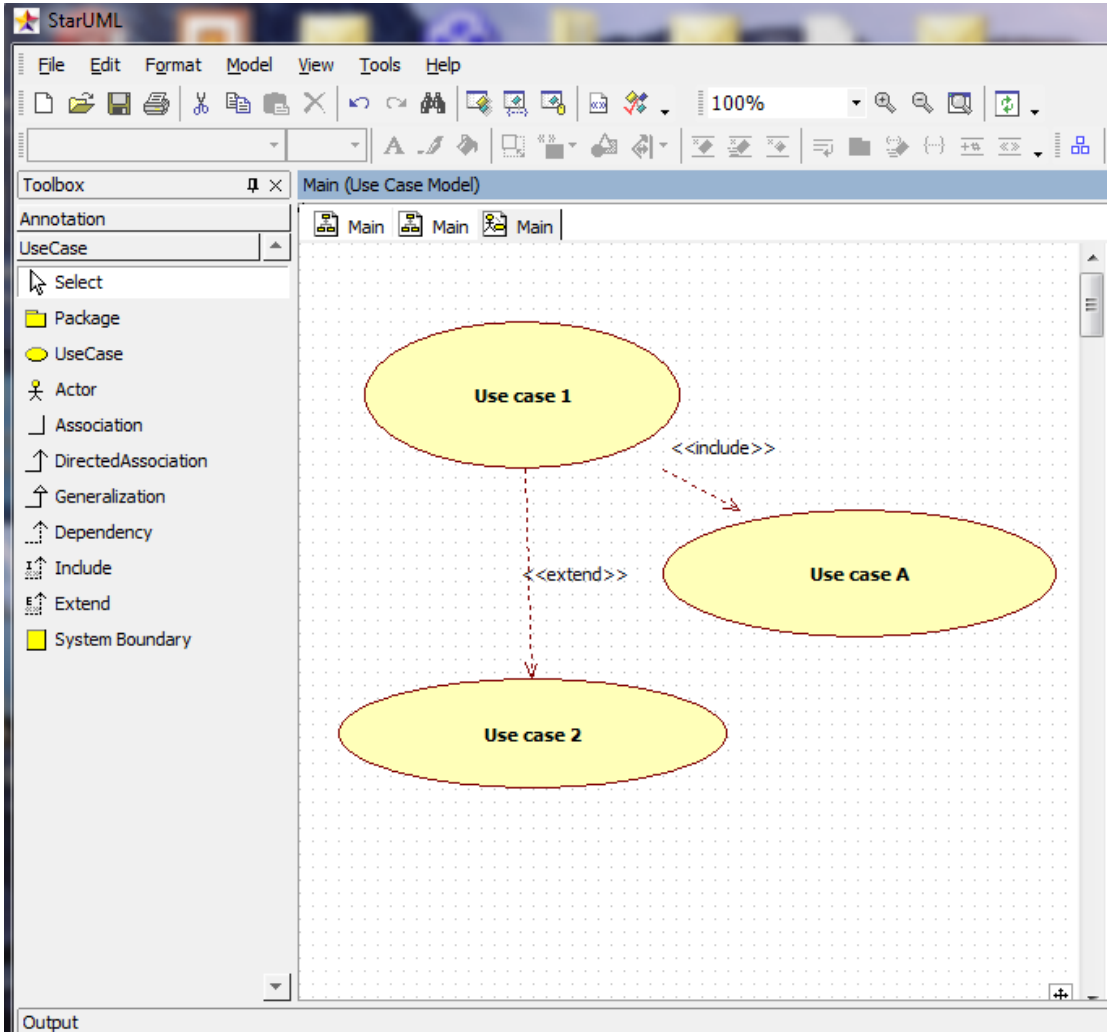
Purpose

- To capture, define and manage user requirements.

How to read this practice

- How to adopt the User story practice

Use case modeling



Create GUI Mockups

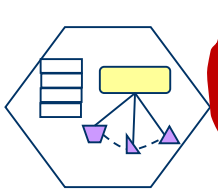
Balsamiq: http://www.balsamiq.com

Web Sites

iPhone Apps

Template of a Use Case Description

Use Case Template		Examples
Use Case Name		Visualise proposed water height after the tsunami event
Use Case ID		CS1-UC01
Revision		CS1-UC01-01
Status		Active
Goal		To get a map of the affected area with the proposed water height after the tsunami event
Summary		The user opens the browser which shows map-window with the water height after the tsunami event in the affected area
Category		primary
Actor		Employee in a local tsunami warning centre
Primary Actor		Employee in a local tsunami warning centre
Stakeholder		
Requested Information Resources	Data input	satellite scene with near infra-red and visible spectrum (e.g. Landsat); bounding box with spatial extent (e.g. WGS84); temporal extent (ttmmjjjj, hh:mm), calculated forecast of the water height
	Data access control	no special access control
	Data format	digital raster dataset image in the browser
Preconditions		The user has opened the portal successfully.



Latest Books By Dean Leffingwell

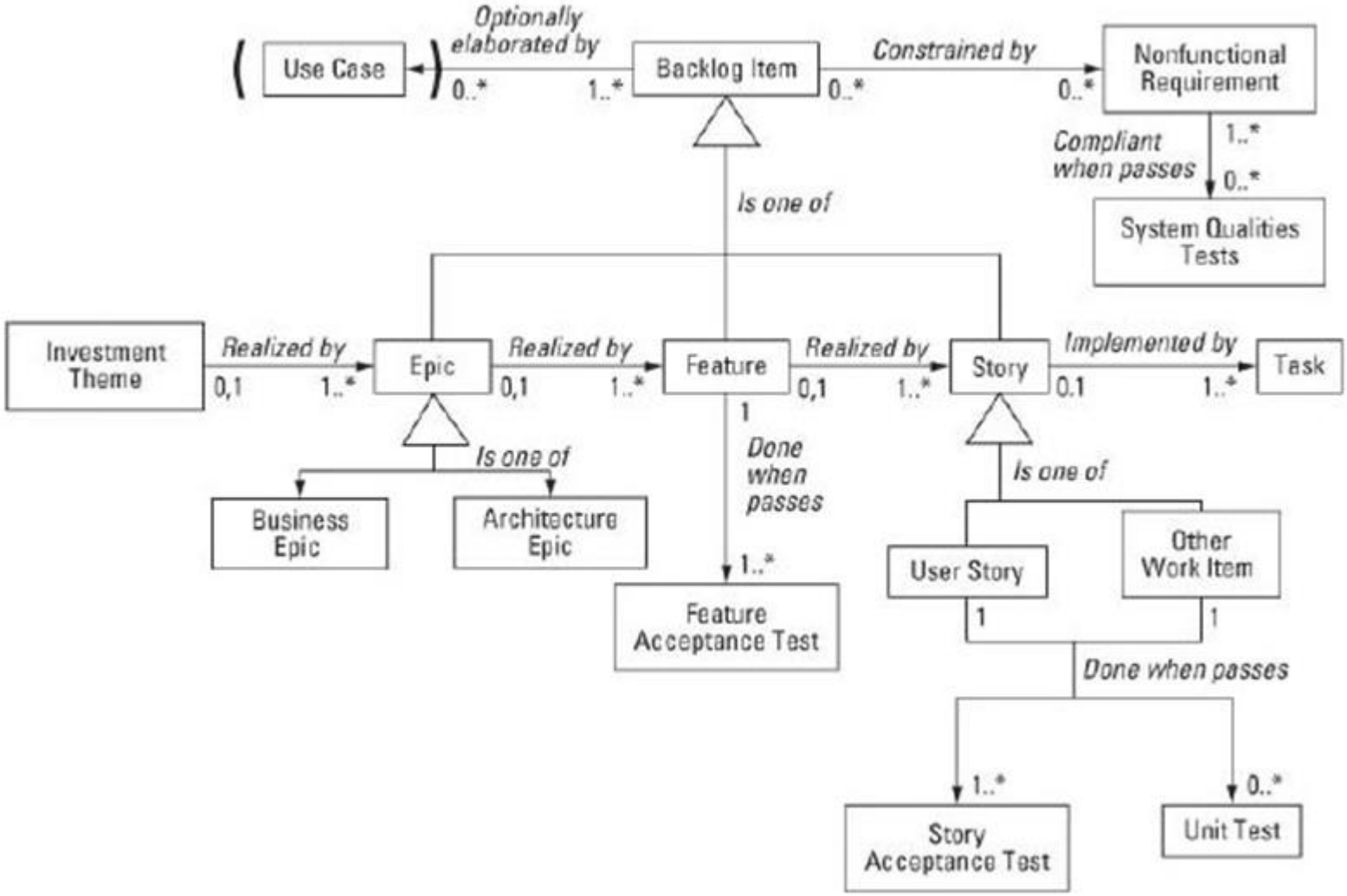
Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise

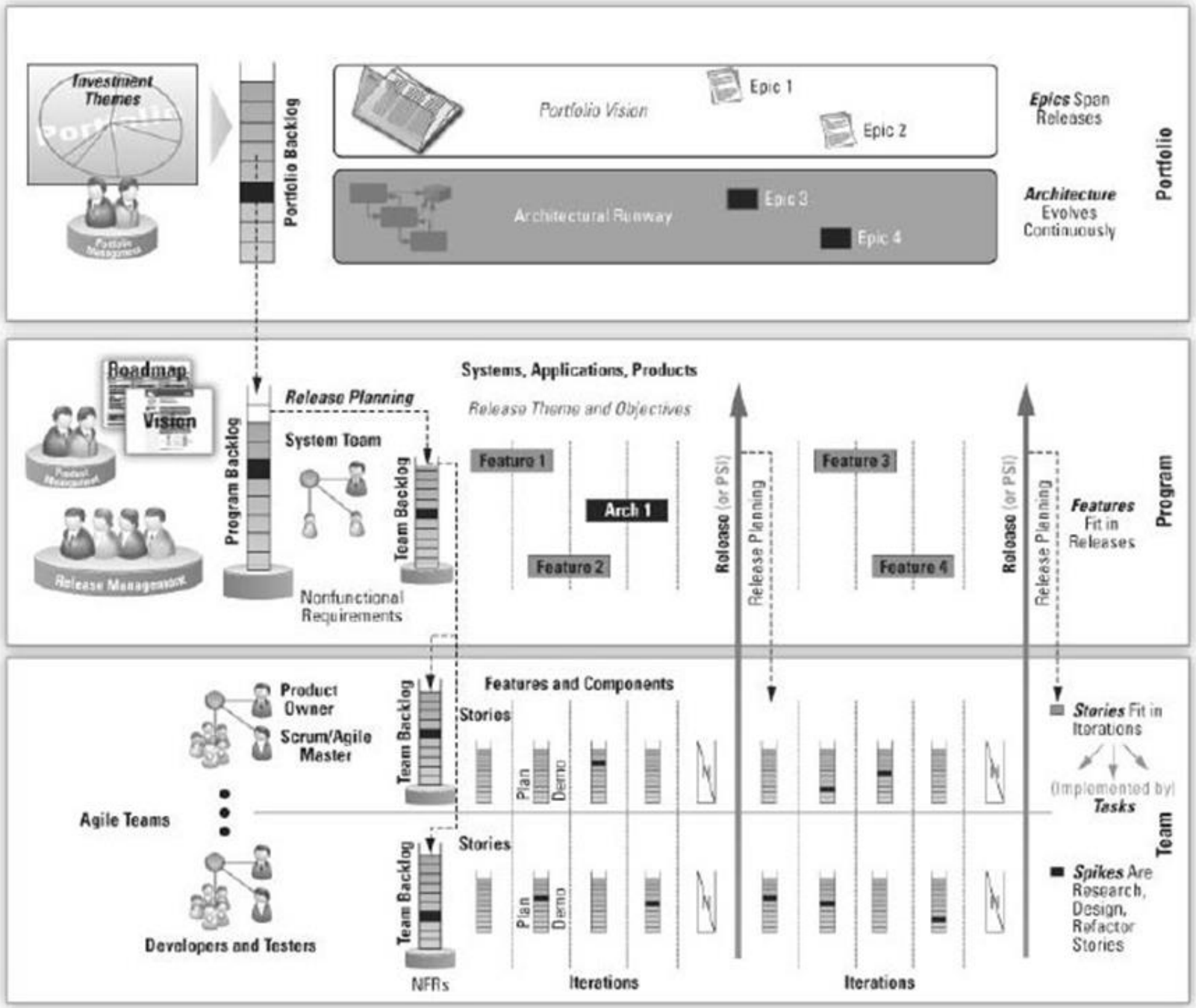
Scaling Software Agility: Best Practices for Large Enterprises

User Story template

- I <in the role of XX> needs functionality <zzz> to achieve the goal of <YYY>

Backlog metamodel




Investment Themes
Portfolio Backlog
Portfolio Vision
Epic 1
Epic 2
Epics Span Releases
Architectural Runway
Epic 3
Epic 4
Architecture Evolves Continuously
Portfolio
Roadmap
Vision
Release Management
Program Backlog
Release Planning
System Team
Team Backlog
Nonfunctional Requirements
Systems, Applications, Products
Release Theme and Objectives
Feature 1
Arch 1
Feature 2
Feature 3
Feature 4
Release (or PSI)
Release Planning
Features Fit in Releases
Program
Product Owner
Scrum/Agile Master
Agile Teams
Developers and Testers
Features and Components
Stories
Plan
Demo
NFRs
Iterations
Stories Fit in Iterations
Implemented by
Tasks
Spikes Are Research, Design, Refactor Stories
Team

IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Use-Case 2.0

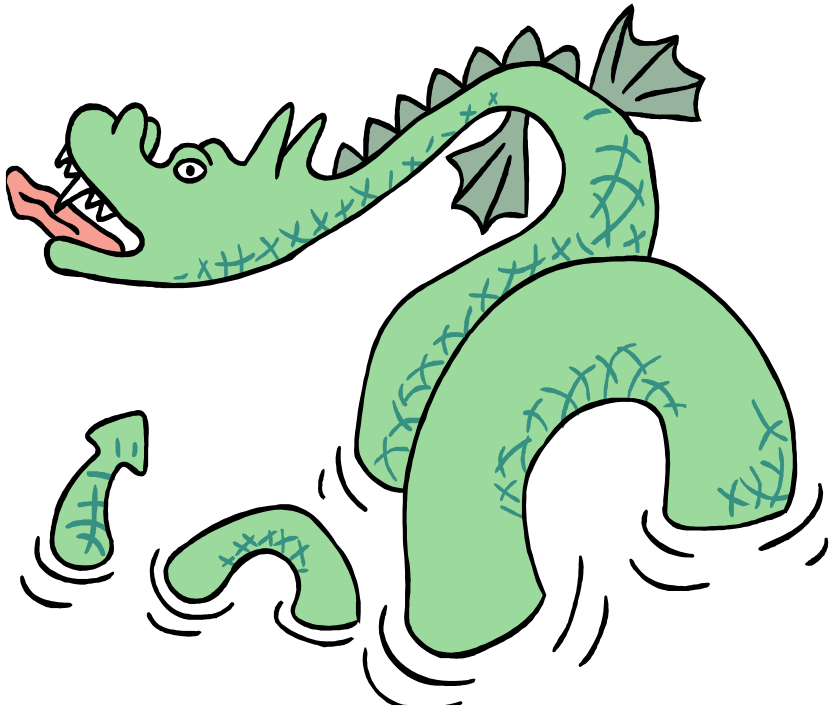
Module 7 - Adapting Your Use-Case Model - Using Include and Extend

www.ivarjacobson.com

Objectives

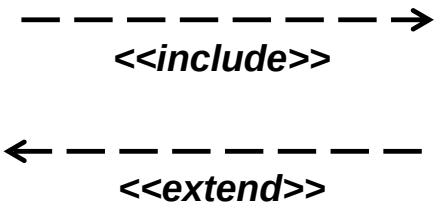
- Understand the use-case relationships, *include* and *extend*
- Understand the appropriate use of use-case relationships
- How to avoid common use-case relationship mistakes

An introduction to use-case relationships



"Here there be Dragons"

"If there is one thing that sets teams down the wrong path, it's the misuse of the use-case relationships: *include* and *extend*"

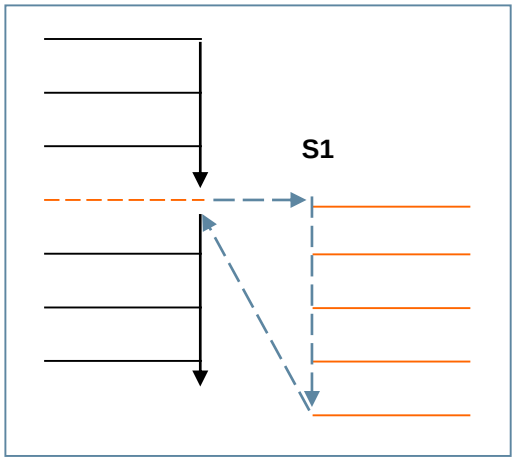


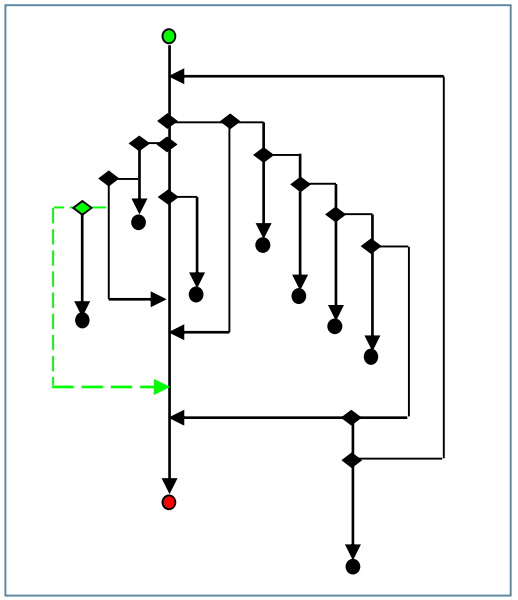
IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Using named subflows / alternative flows to structure text

- Named subflows and alternative flows provide powerful techniques for structuring a use case's flow of events *without* resorting to use case relationships:
- These techniques should be used to their fullest before additional relationships between use cases are introduced
- Most systems can be fully described without resorting to use-case relationships



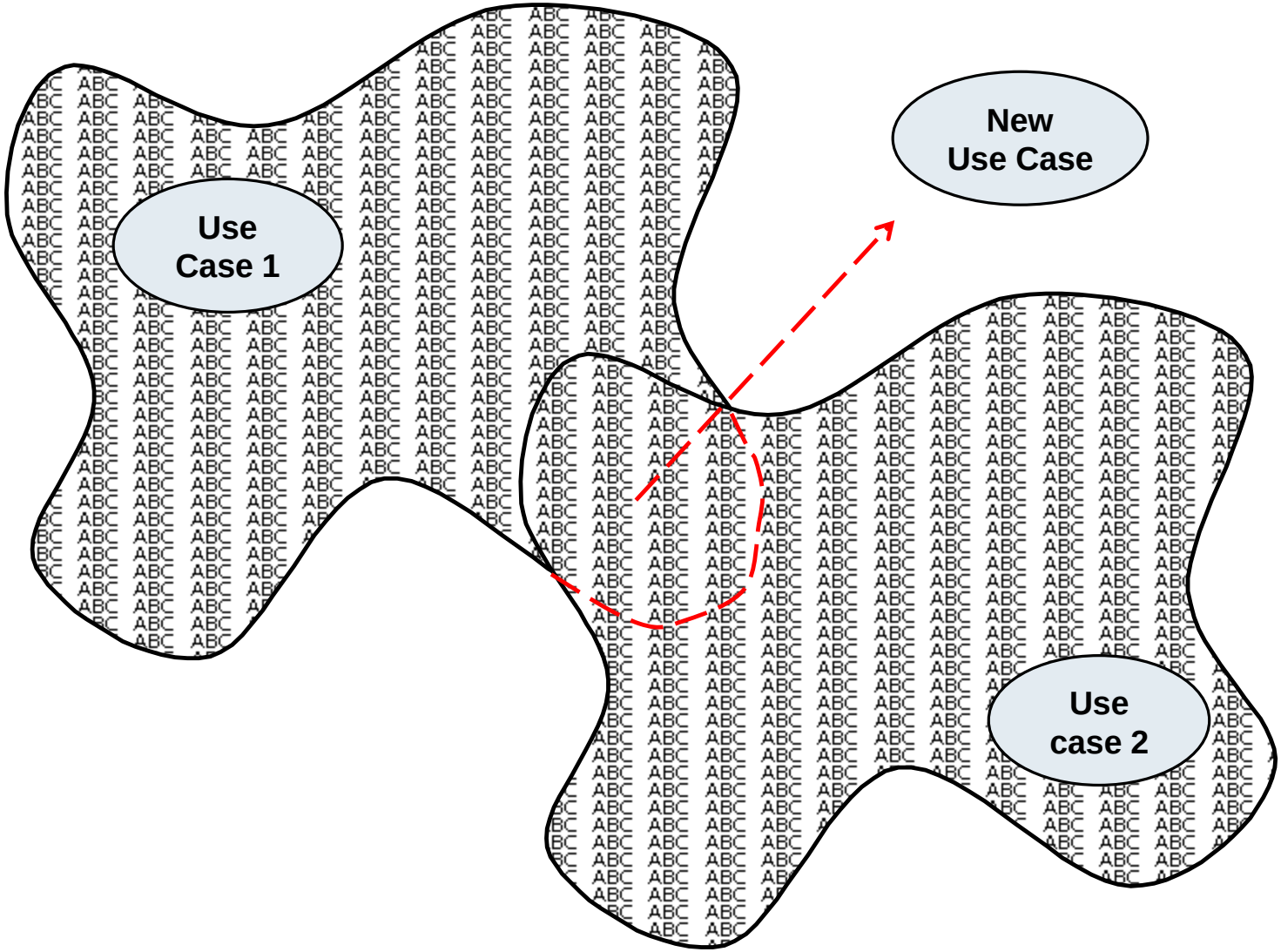


Defining Relationships Between Use Cases

- If most systems can be described without them, and most teams get into trouble when they try them, why persevere with use-case relationships?
  - Commonality of behavior between two or more use cases (*include*)
  - Adding additional behavior to an existing use case (*extend*)
  - Reducing complexity by isolating portions of use cases (*extend*)

"Never introduce relationships between use cases until you have at least a draft of the flow of events of the use cases"

Using the Include Relationship



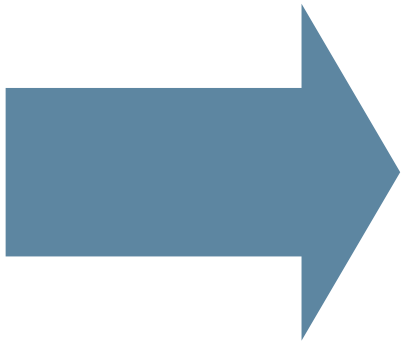
Using the Include Relationship

- The *include* relationship provides the ability to extract common sections from two or more use-case narratives
- In order to use the include relationship, you must have the same descriptive text in at least two different use-case narratives
  - Only introduce an *include* after a use-case narrative is written (beware of working only with diagrams)
  - Never introduce an include that is included by only one use case – totally pointless!

Using the Include Relationship

Base Use Case

I am in charge. Just as with a sub-flow, "I decide when and where I use him"



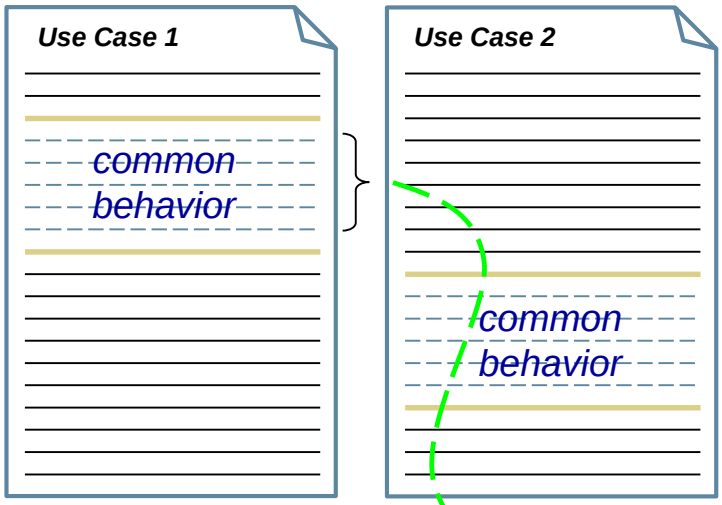
Included Use Case

I am Just like a sub-flow, "I have no knowledge of him"

The included use case has no knowledge of the base, including use case.

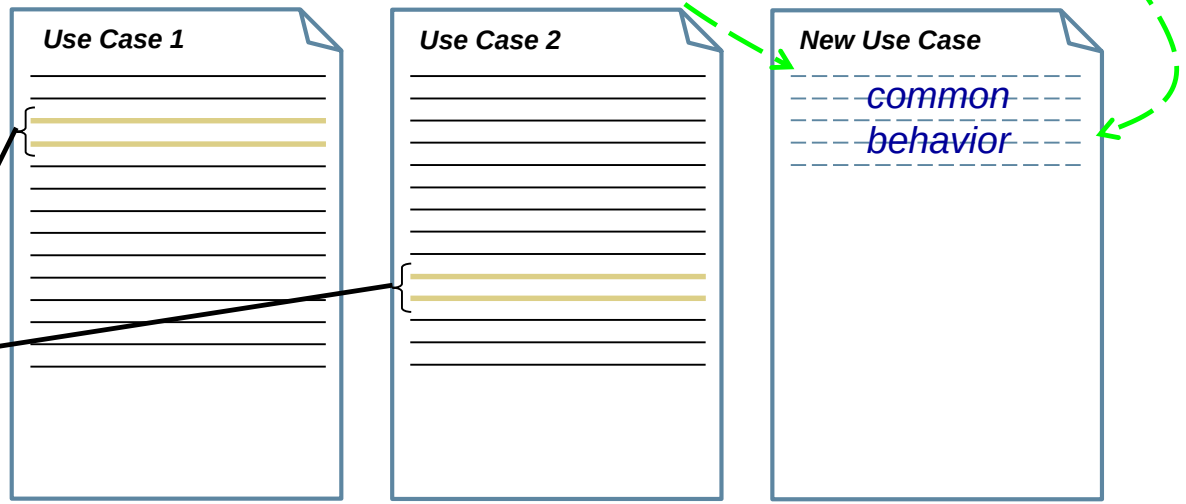
Using the Include Relationship

Before
<<*Include*>>



After
<<*Include*>>

…"include use case [New Use Case] [reason]"…

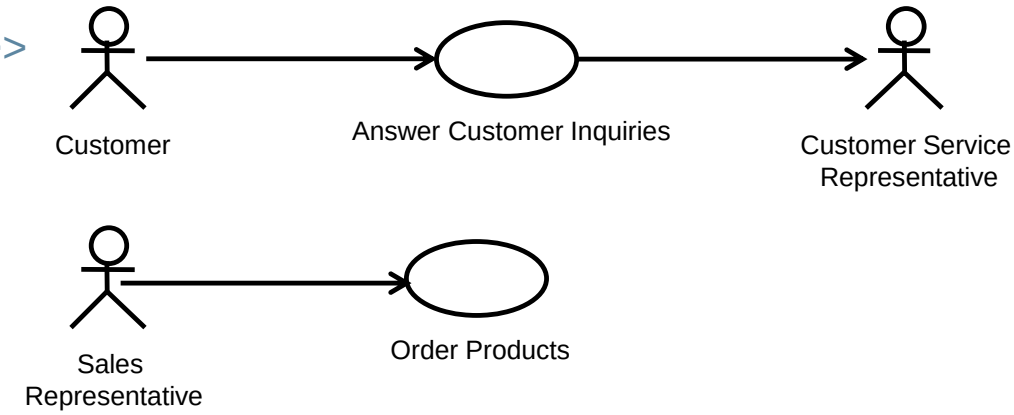


IVAR JACOBSON INTERNATIONAL

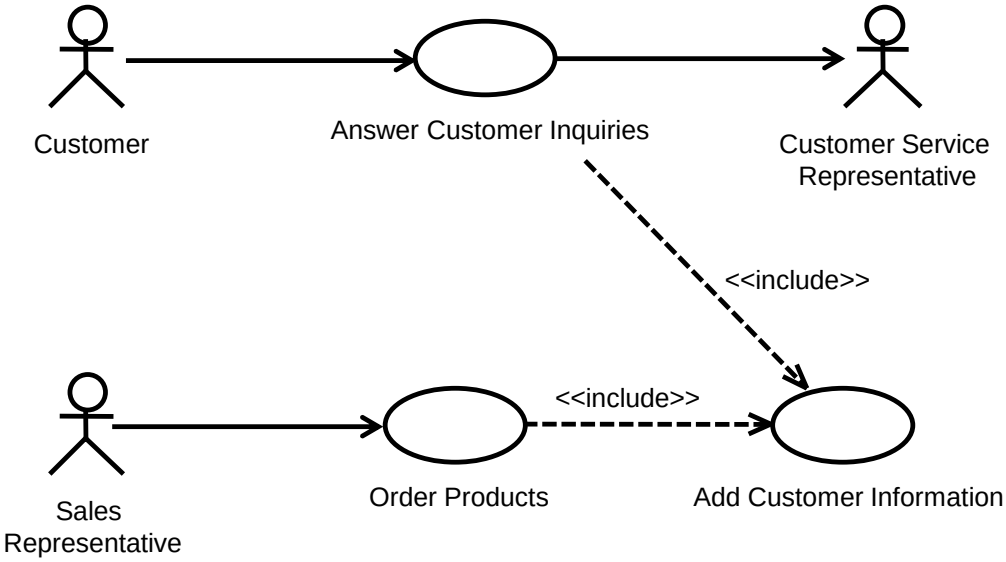
THE SMARTER WAY

Using the Include Relationship

Before <<*Include*>>



After <<*Include*>>



IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Discussion: Using the Include Relationship

- Find and read the Use-Case Narrative "*Include*" handout
- Observations:
  - By using an included use case, the two original use cases are simplified
  - The definition of *customer information* has been hived off to the glossary
  - In creating the new included use case, parts of both original use cases had to be re-written (not to mention the writing of the new use case)
- The included use case *never* has specific knowledge of the use case that included it
  - Hence included use cases are reusable

Handout:
Include Relationship

Common Errors Using the Include Relationship

- Used to perform functional decomposition:
  - Treating the included use case as some form of menu option
    - The including use case ends up as a shell and can no longer provide value on it's own
- The behavior in the included use case is expanded outside the context of the use cases that include it
  - Scope creep
  - When an include is used it means that the whole of the included use case is part of the including use case

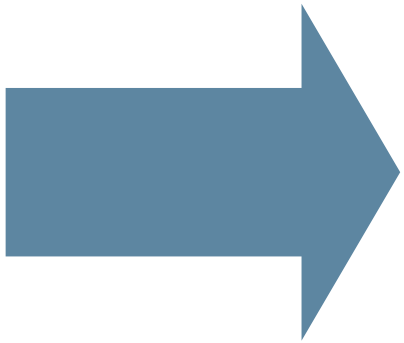
Using the Extends Relationship

- The extend relationship is used where optional or exceptional behavior is inserted into an existing use case
  - The original purpose was a mechanism for specifying options that could be added to an existing product
- The extending use case needs no change to the use case it extends, possible circumstances of use:
  - Descriptions of features that are optional to the basic behavior, “optionally purchased”
  - Complex exception-handling that would otherwise obscure the primary behavior, (alternative flows that are longer than the main flow)
  - Customization of the requirements model for specific customer needs
  - Scope and release management

Using the Extends Relationship

Extending Use Case

I know my place. Just like an alternative flow "I have detailed knowledge of the existence and the potential extension points of him"



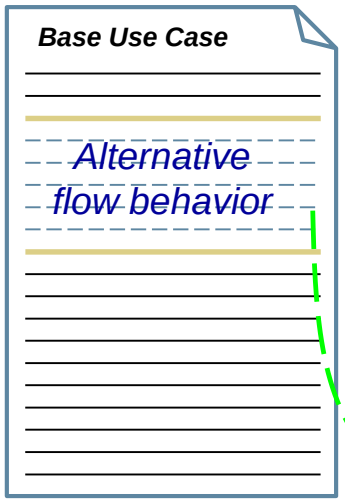
Base Use Case

… "But I have no knowledge of even the existence of him"

The extended, base use case has no knowledge of the extending use case.

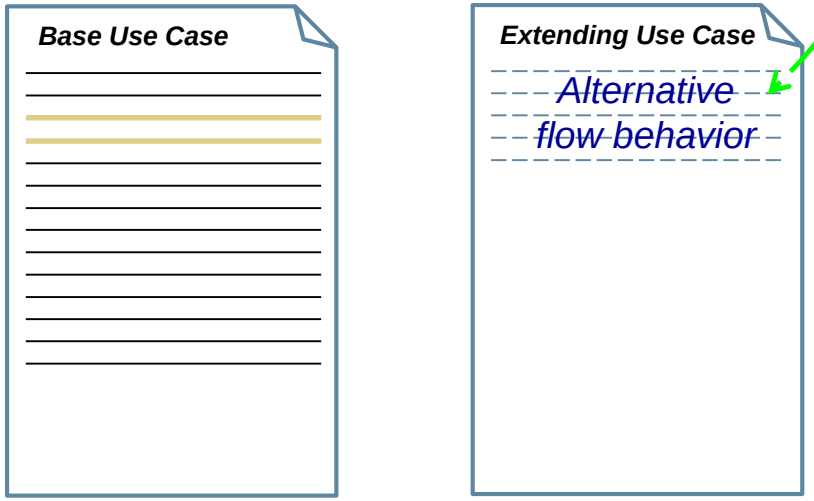
Using the Extends Relationship

Before <<*extend*>>



This illustration shows how an alternative flow can become an extending use case

After <<*extend*>>



…"***extends*** **use case [Base Use Case] at {extension point} where [condition]"…**

IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

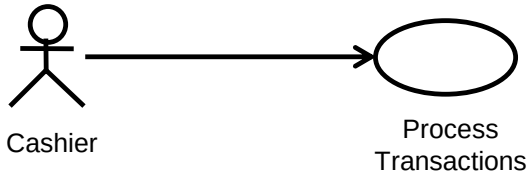
Discussion: Using the Extend Relationship

- Find and read the Use-Case Narrative “Extend” handout
- Notes:
  - The extension allows us to add-on features of the system in a simple way
  - The base use case must remain intact and valuable on it’s own
  - The extension cannot modify the base use case

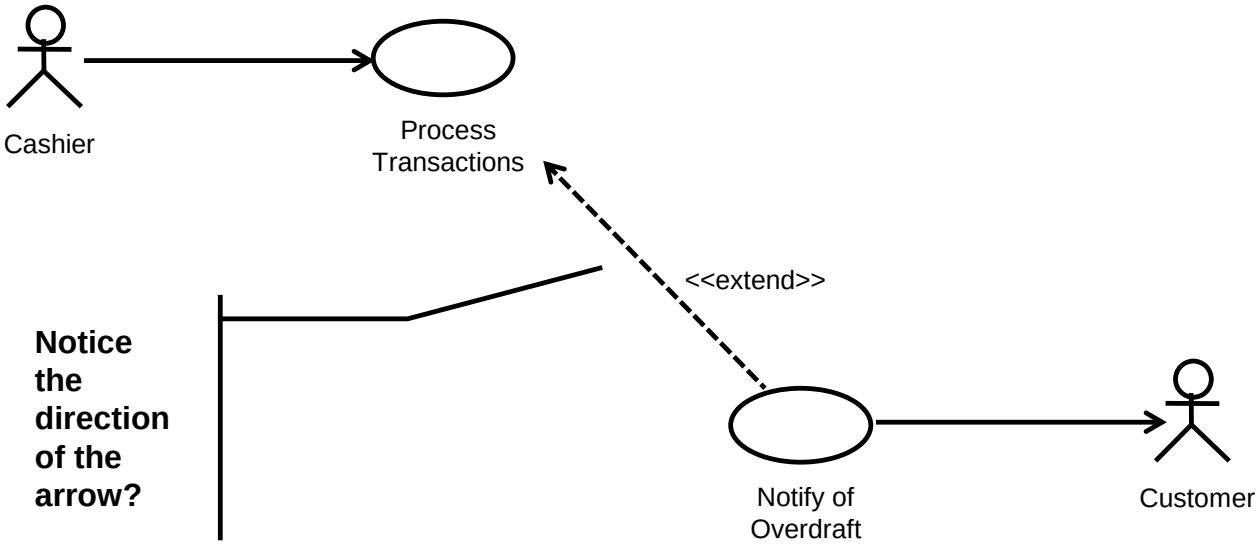
Handout:
Extend Relationship

Using the Extends Relationship

Before <<*Extend*>>



After <<*Extend*>>



Extension Points Revisited

- Extension Points can be public or private
  - Private – visible only within the use case in which they occur
  - Public – visible externally to extending use cases
- The extension point mechanism is used for both alternative flows and extending use cases – {Extension Point}
- There is a section in the use-case narrative to declare public extension points

Use Case – Browse Products and Place Orders

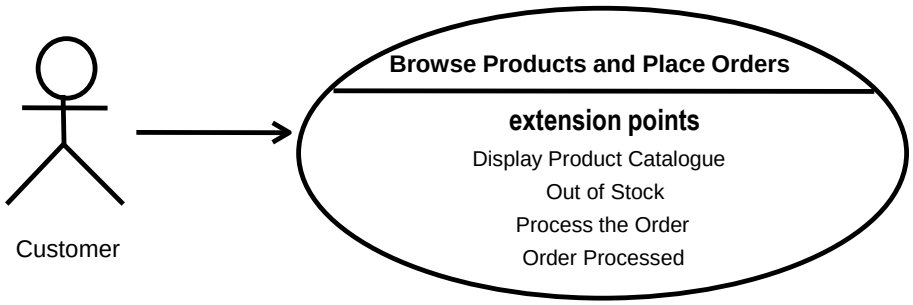
Public Extension Points

{Display Product Catalogue}
{Out of Stock}
{Process the Order}
{Order Processed}

Extension Points Revisited

- Only extension points that represent locations at which the use case can be extended should be made public
- New extension points can be declared in the Public Extension Points section of the use-case where the use-case narrative itself is under strict configuration control thus:

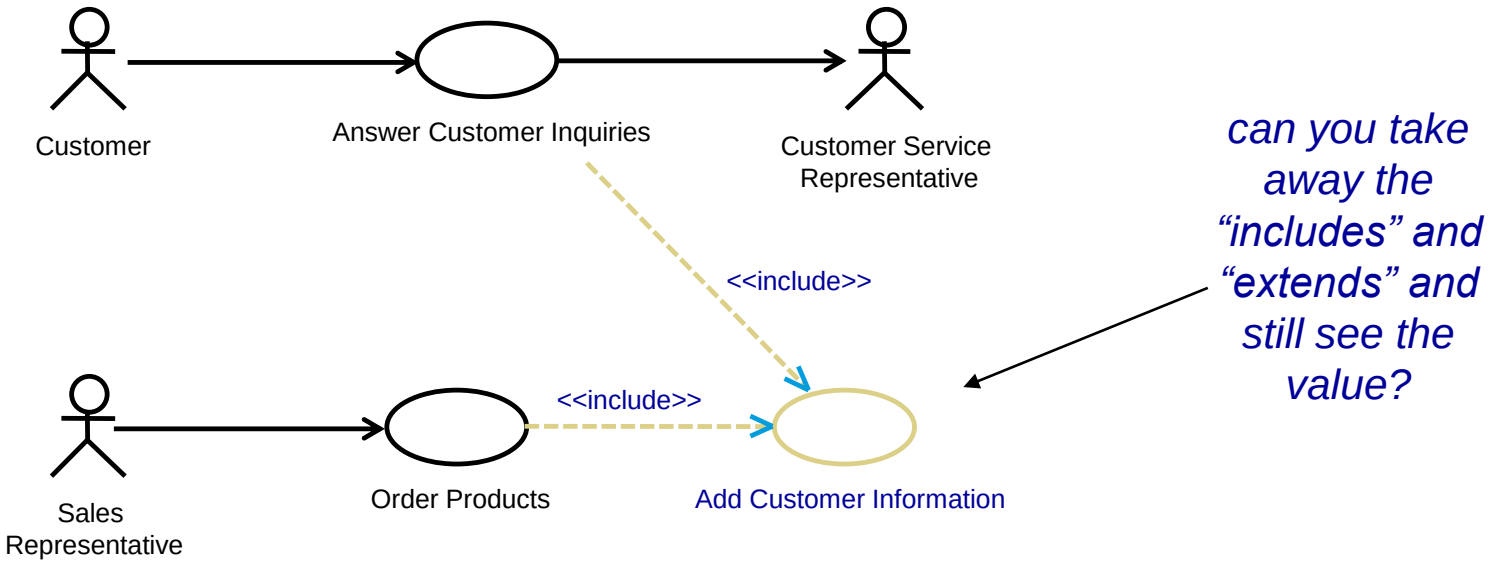
> **{extension point name}**
>
> At <some location in the use-case narrative>, or before <some location in the use-case narrative>, or after <some location in the use-case narrative>



Public extension points can be shown as part of the use case on use-case diagrams, in a compartment named extension points.

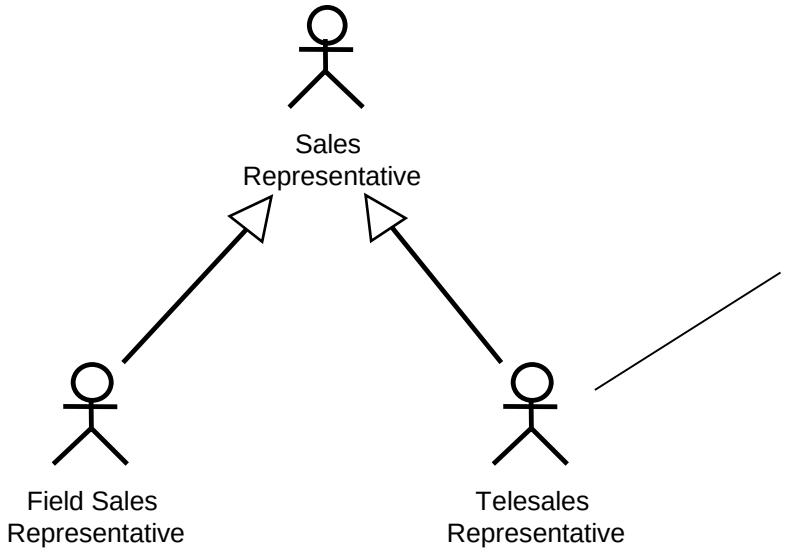
Evaluating the Resulting Use-Case Model

- The basic use cases of the system should reflect the value provided by the system



Defining Relationships Between Actors

- The only relationship *between* actors is *generalization*
  - It is used to show similarity between actors
  - The main value is to show that some group of actors share common responsibilities or common characteristics

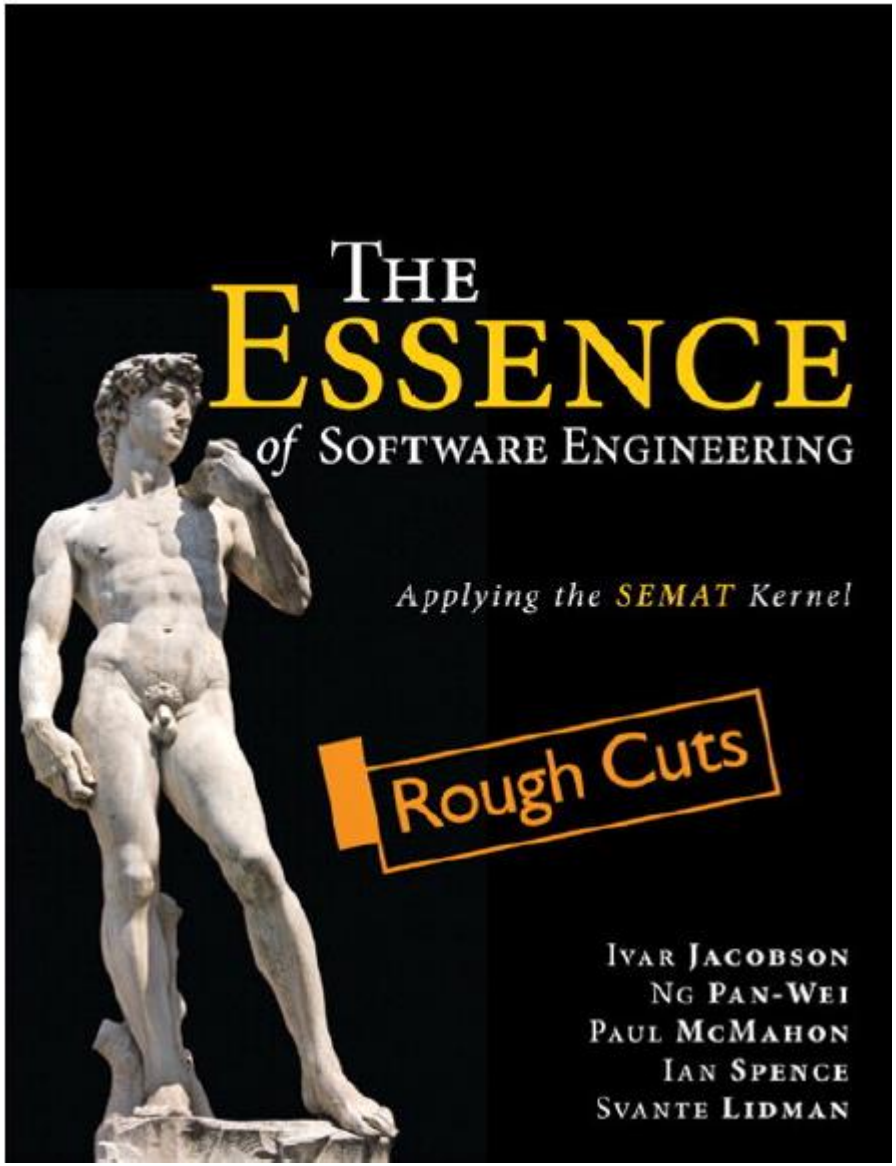


The Field and Telesales Representatives inherit characteristics from the Sales Representative, including the *communicates* relationships with other use cases

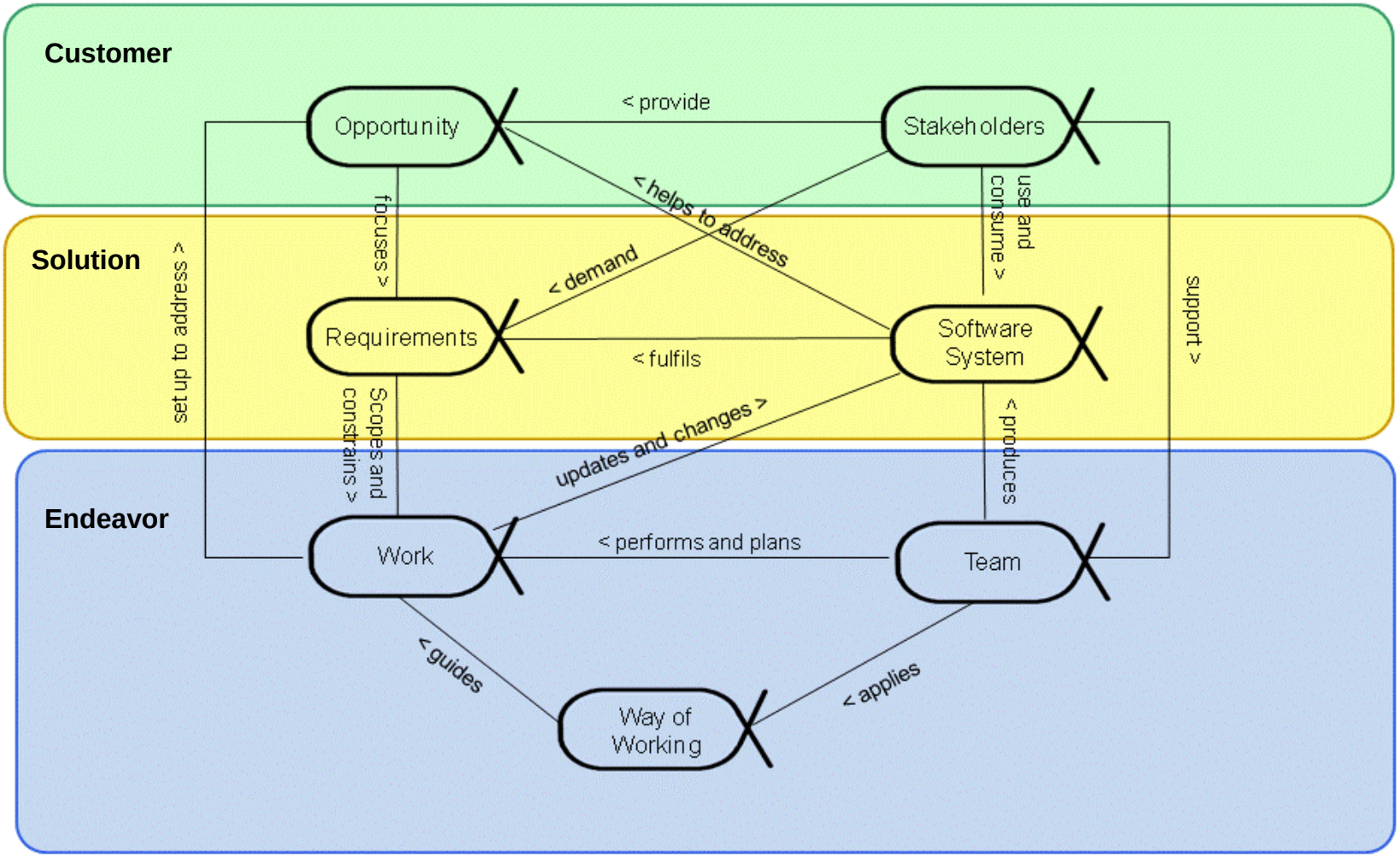
Summary

- Use-Case relationships are often misused and should be undertaken with care
- Never structure the use-case model before you have written any use-case narratives
- The *include* relationship is for situations where there is truly common behavior to more than one use case
- The *extend* relationship is primarily used to extend the behavior of an existing use case

Book is available now – Safaribooksonline/Addison Wesley



Alphas: The Essential Things to Work With



IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Scrum– with Essence

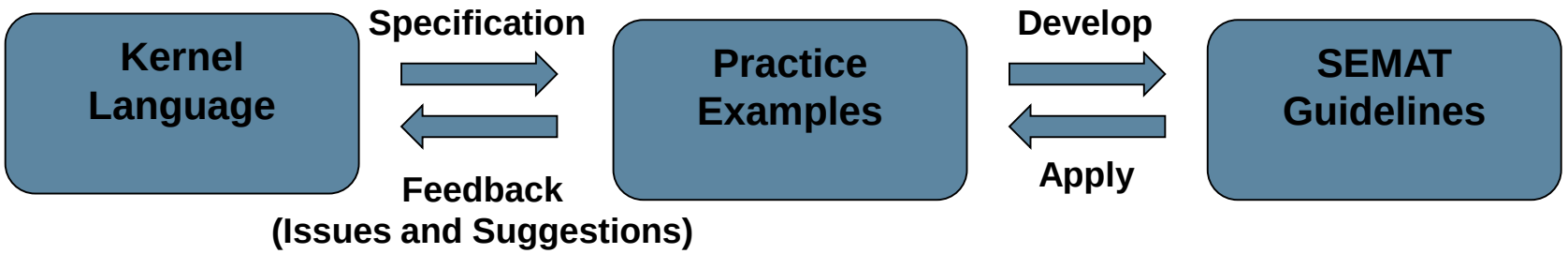
Brian Elvesæter

Arne J. Berre

www.ivarjacobson.com

Objective

- Exercise the SEMAT Kernel and Language



- Illustrate the SEMAT approach
  - One example of how the Scrum practice may be mapped to the SEMAT Kernel and Language

- Develop and apply methods for projects
  - Agile requirements with User Stories and/or Use Cases practices
  - Agile project management with Scrum or "Scrum-like" practices

About Scrum

- Scrum consists of Scrum Teams and their associated roles, events, artifacts, and rules.
- Scrum's roles, artifacts, events, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum.

- Source
  - K. Schwaber and J. Sutherland, "The Scrum Guide", Scrum.org, October 2011.
  - http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf

Scrum Concepts

- Scrum team (roles)
  - Product Owner
  - Development Team (of developers)
  - Scrum Master
- Scrum artifacts
  - Product Backlog
  - Sprint Backlog
  - Increment
- Scrum events
  - The Sprint
  - Sprint Planning Meeting
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective

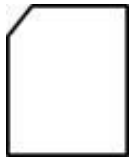
Step 0: SEMAT Kernel & Essence Language Concepts

- A standard **Kernel** provides a baseline starting point – a "map" of the software development endeavour.
- **Practices** add details and provide specific guidance on particular aspects of the software development
- Key language concepts: **Alpha**, **Activity Space**, **Work Product** and **Activity**

Kernel

Practice

Alpha

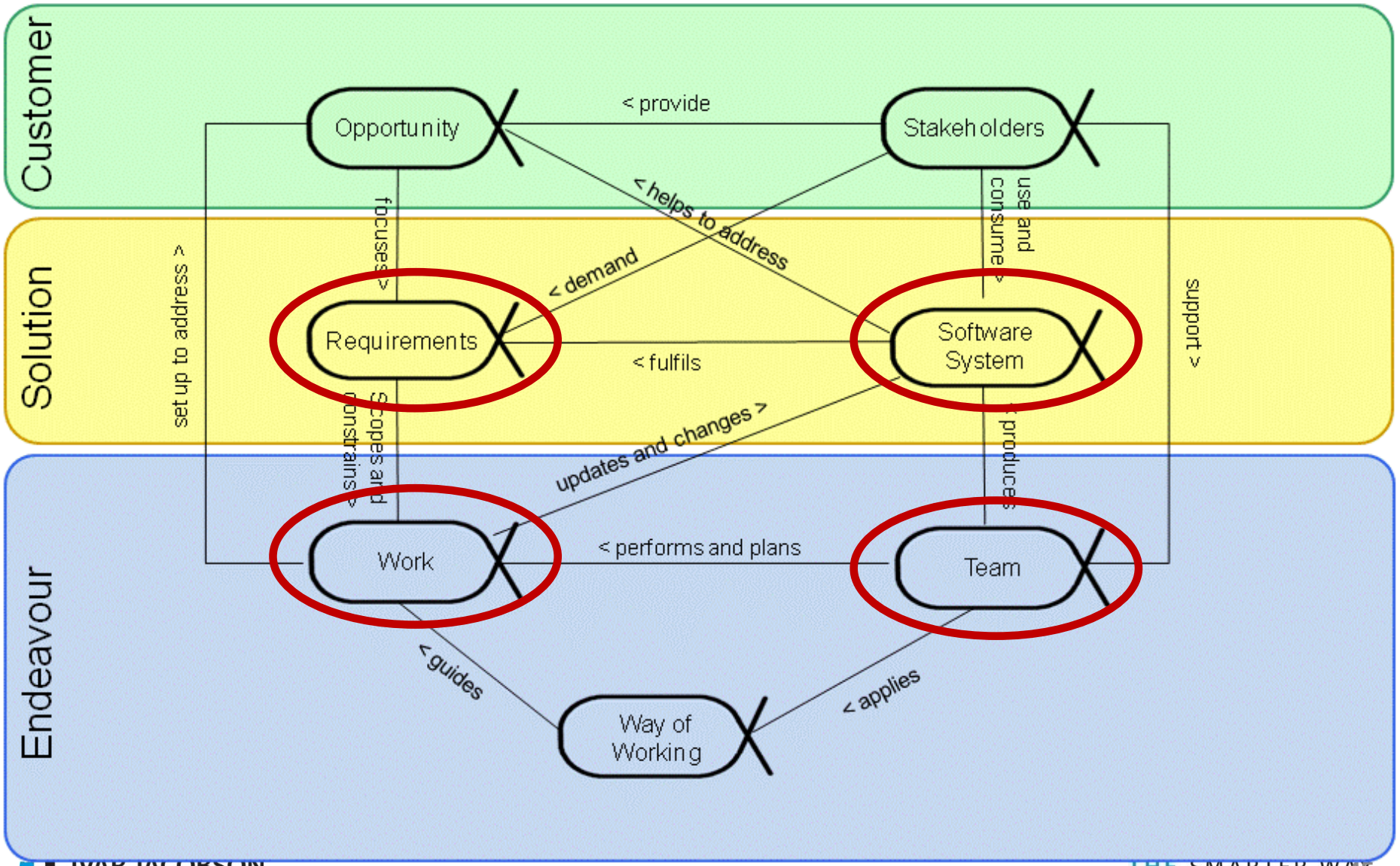


Work Product

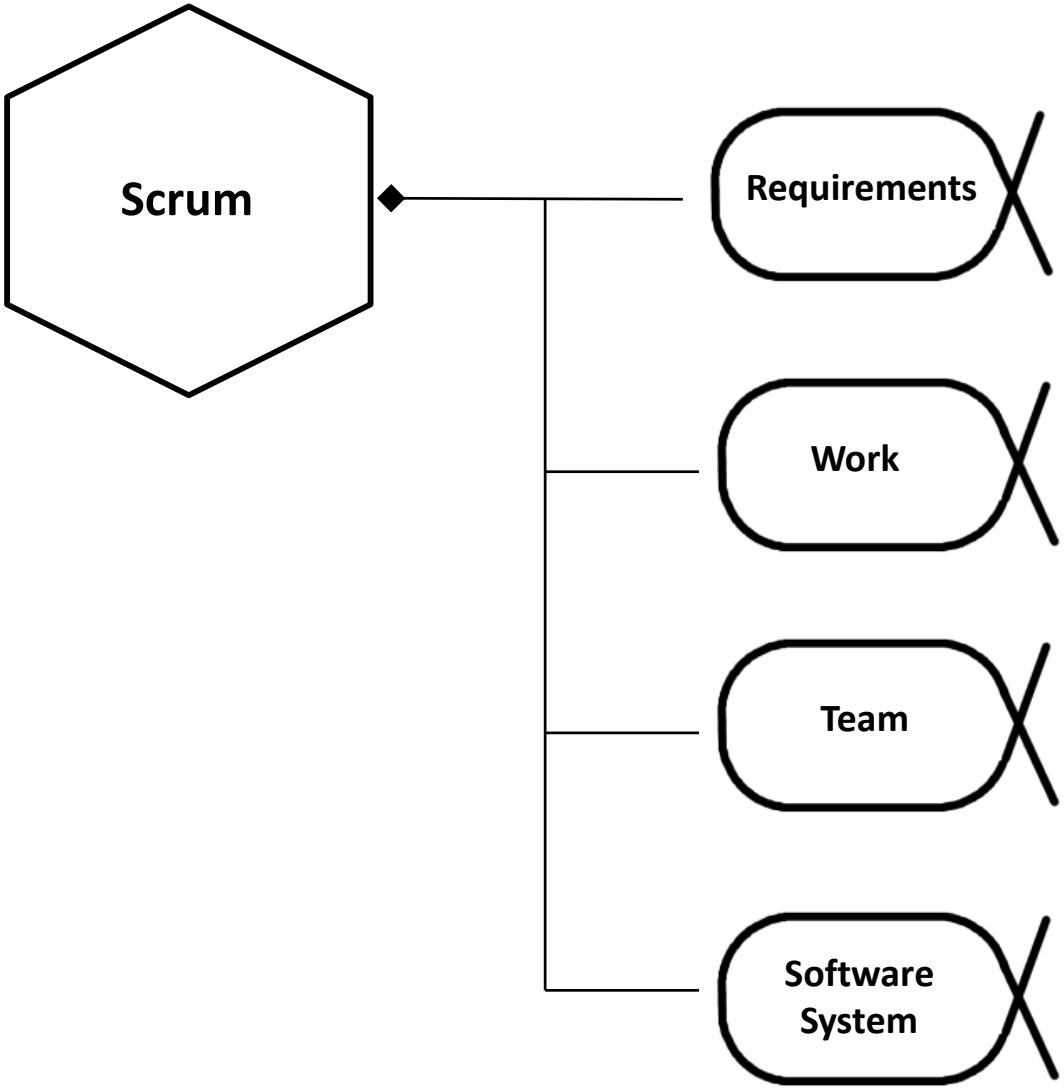
Activity Space

Activity

Step 1a: Identify relevant Kernel Alphas

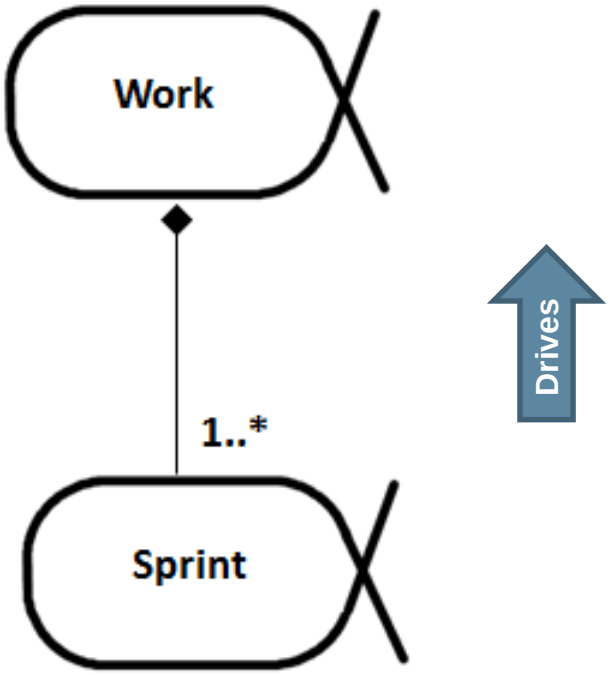


Step 1b: Outline the Scrum Practice



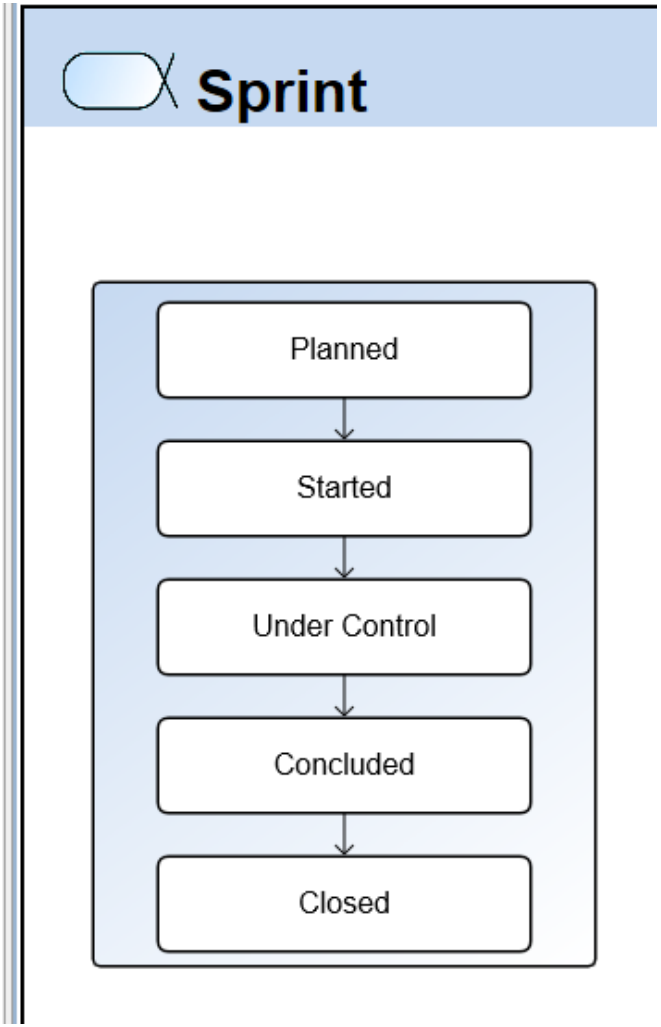
Step 2a: Add sub-alphas

- Extending the Work Alpha

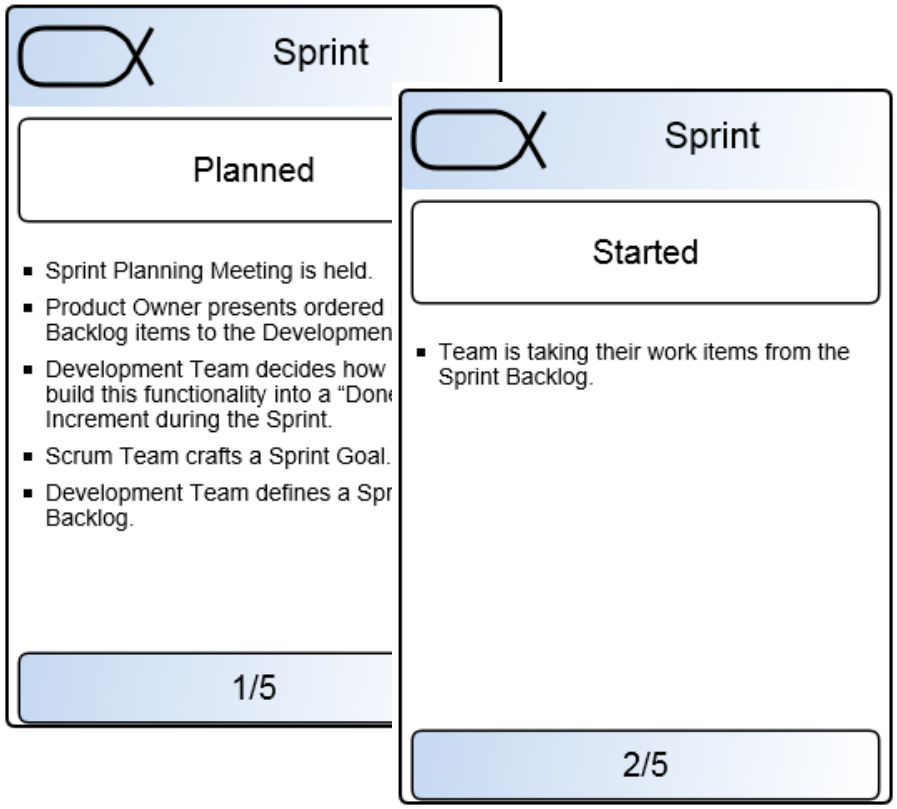


- The Work alpha is typically used for the duration of a development project that may cover a number of sprints.
- Thus we define a new sub-alpha called **Sprint**.
- Sub-alphas drive their parent alphas

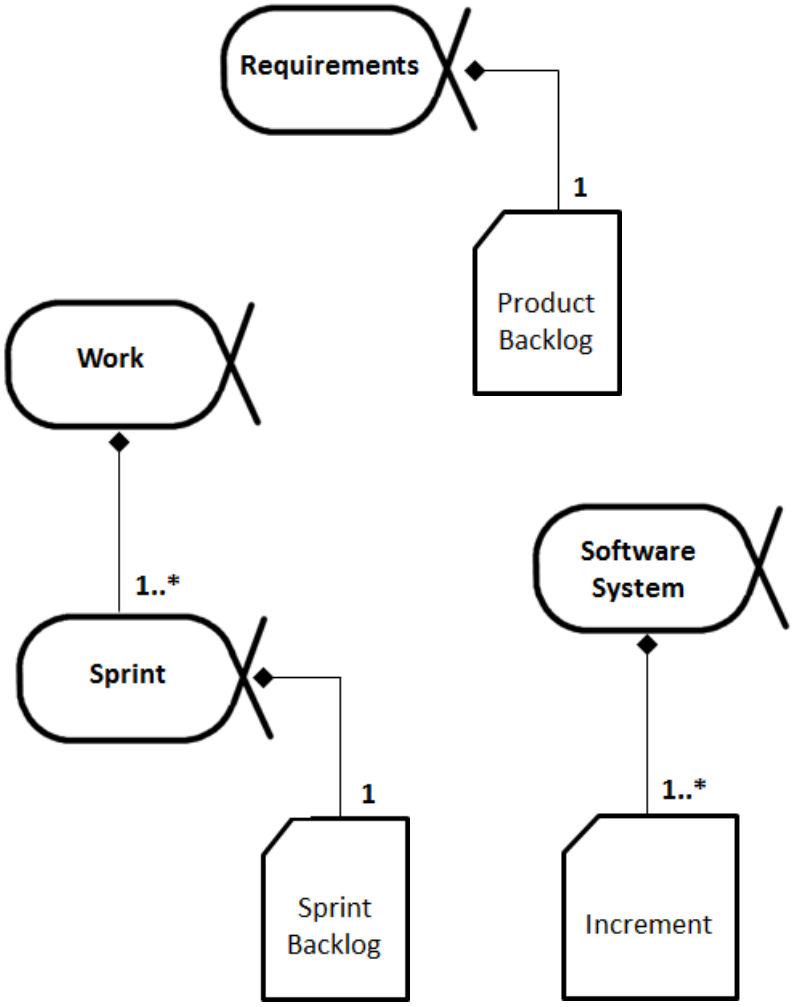
Step 2b: Define alpha states and checkpoints



- Specific Scrum rules are defined as part of the alpha state checkpoints.

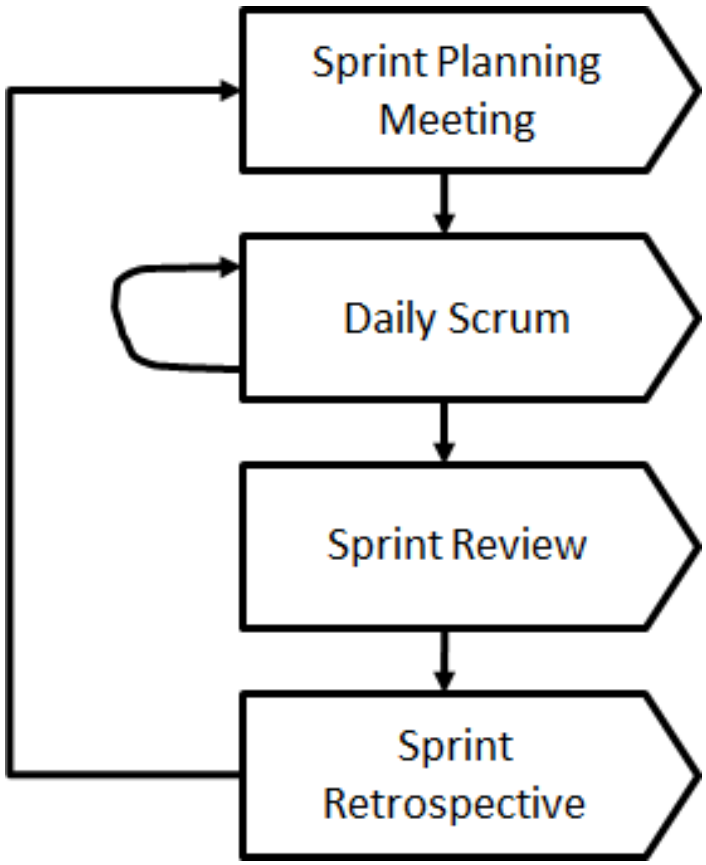


Step 3: Add Work Products



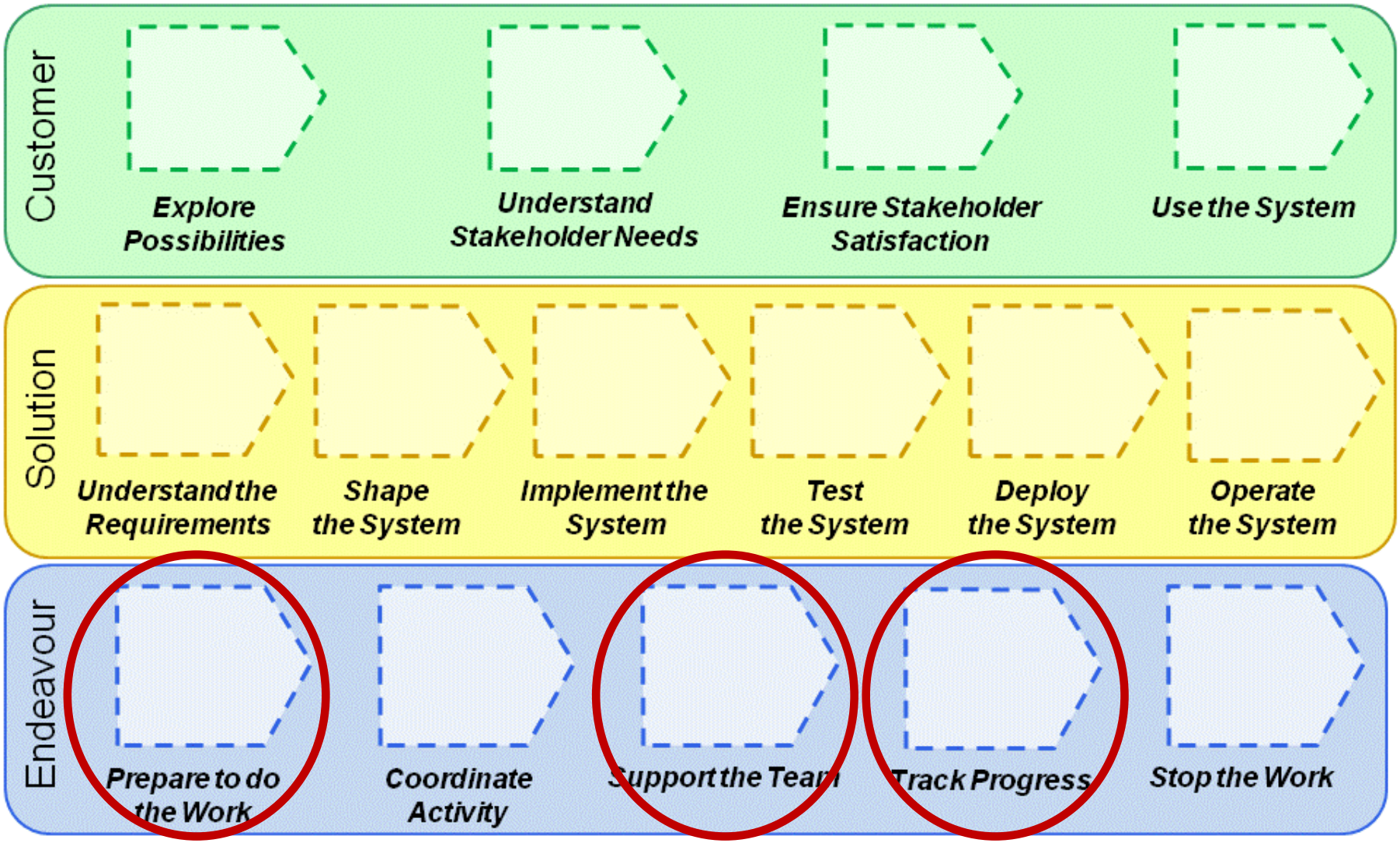
- "The **Product Backlog** is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product."
- "The **Sprint Backlog** is the set of Product Backlog items selected for the Sprint plus a plan for delivering the product Increment and realizing the Sprint Goal."
- "The **Increment** is the sum of all the Product Backlog items completed during a Sprint and all previous Sprints."

Step 4a: Define Activities



- "The work to be performed in the Sprint is planned at the Sprint Planning Meeting."
- "The Daily Scrum is a 15-minute time-boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours."
- "A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed."
- "The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning Meeting."

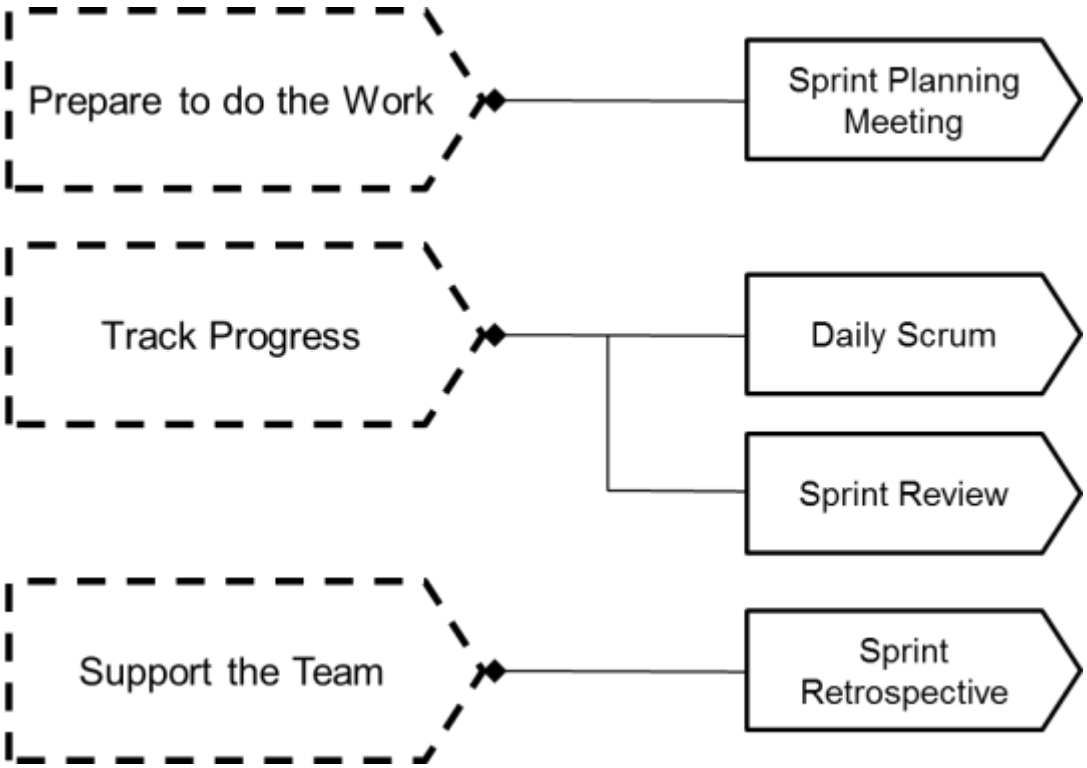
Step 4b: Identify relevant Kernel Activity Spaces



IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Step 4c: Relate activities to Kernel Activity Spaces



- NB! Just one possible suggestion. The organization depends amongst others on how one interpret and define the completion criteria of the Activities.

Outline

- Scrum Practice – Reference Example

- Defining the Scrum Practice

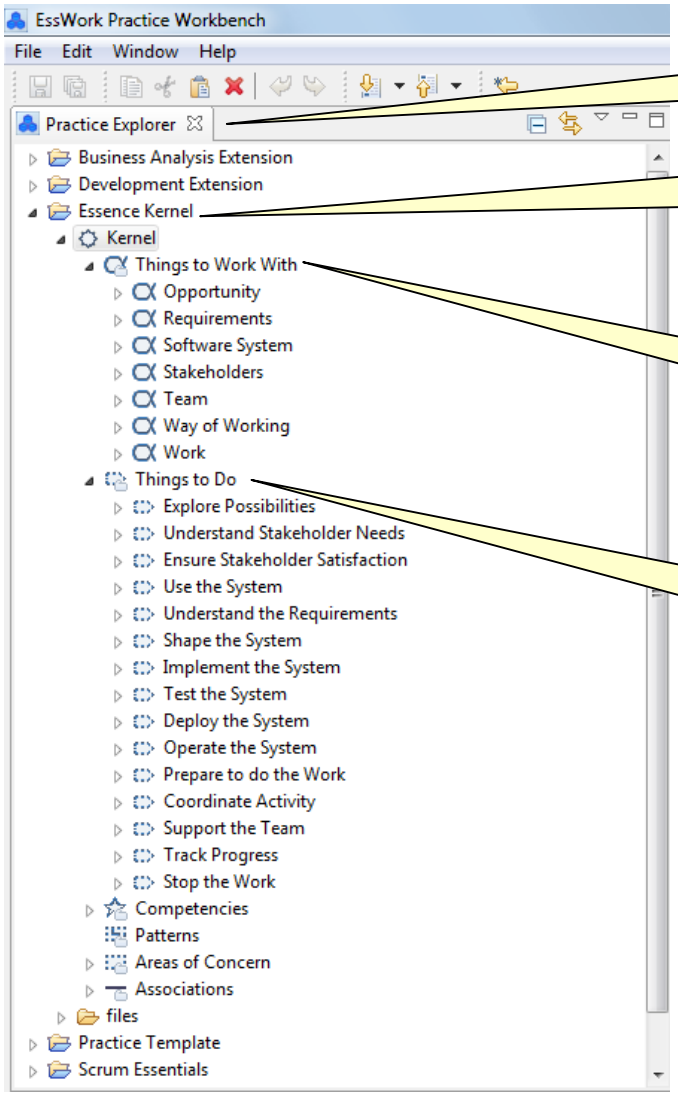
- Authoring the Practice in the EssWork Practice Workbench

- Questions

Sources

- OMG Essence specification
  - OMG, "Essence – Kernel and Language for Software Engineering Methods", OMG Document ad/2013-02-01, 18 February 2013.
  - http://semat.org/wp-content/uploads/2013/02/Essence_final_submission_18Feb13.pdf
- Scrum Guide
  - Ken Schwaber and Jeff Sutherland, "Scrum Guide", October 2011.
  - http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf
- Practice authoring tool
  - EssWork Practice Workbench
  - http://www.ivarjacobson.com/EssWork_Practice_Workbench/

Practice Explorer



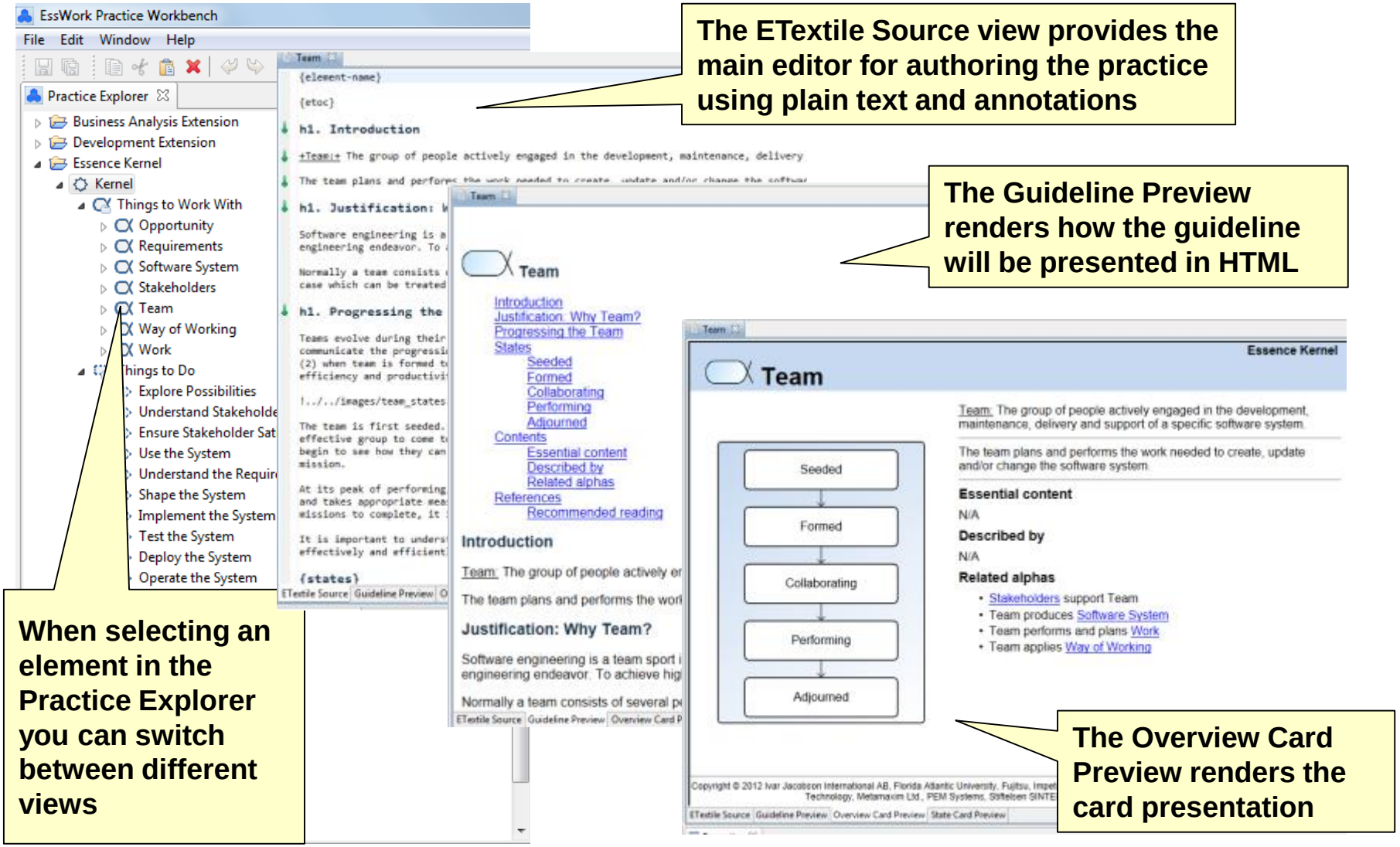
The Practice Explorer shows Practice Workbench projects

The Essence Kernel project contains the elements defined in the OMG Essence specification

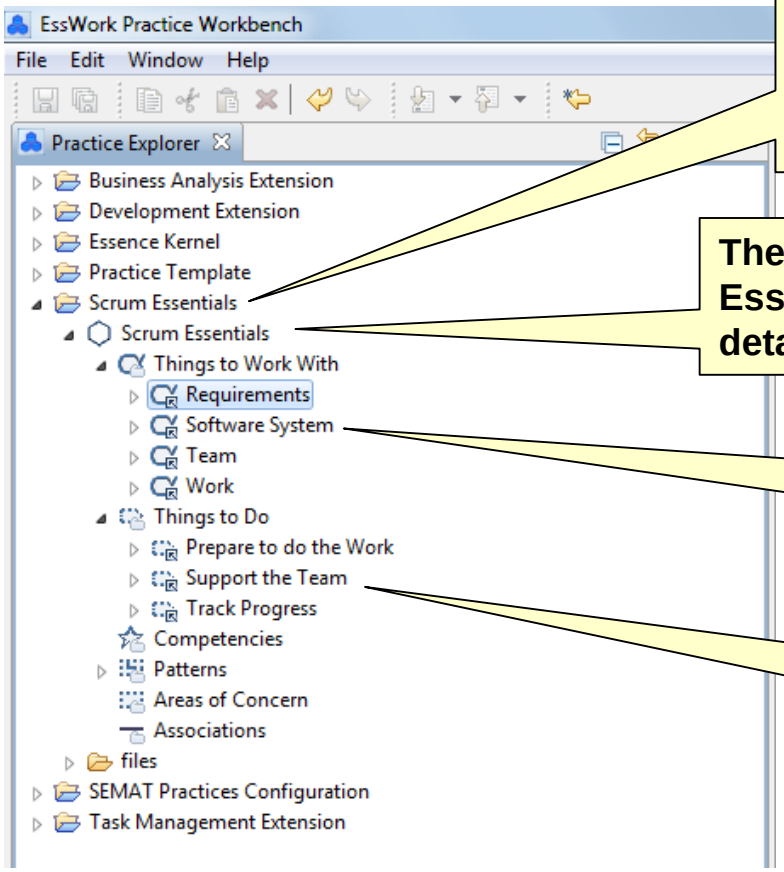
Alphas that represent the essential things to work with

Activity Spaces that represent the essential things to do

ETextile, Guideline and Card views



Scrum Essentials



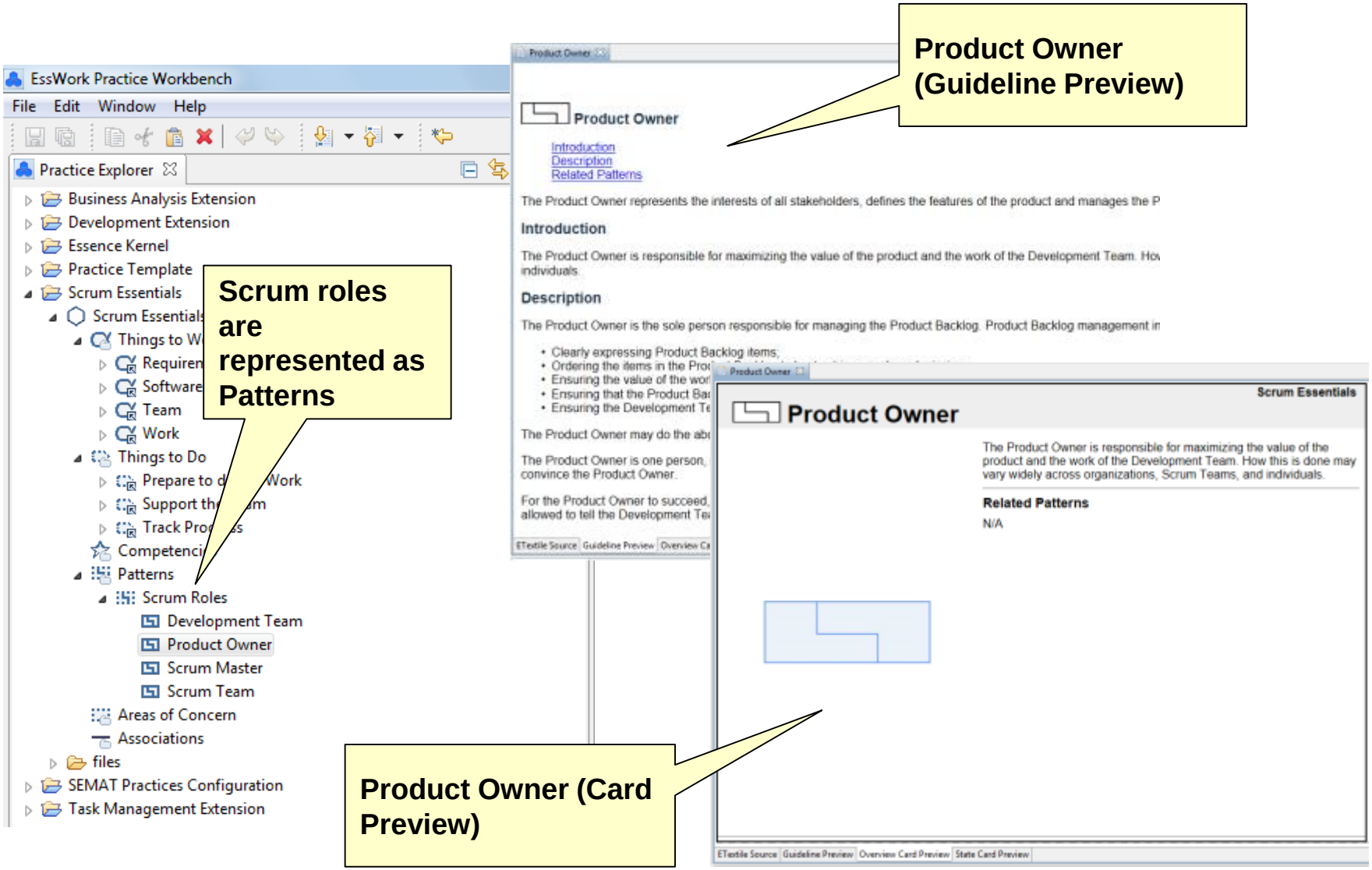
The Scrum practice is created as a separate practice project in the Practice Workbench

The Scrum practice extends the Essence Kernel by providing more detailed guidance.

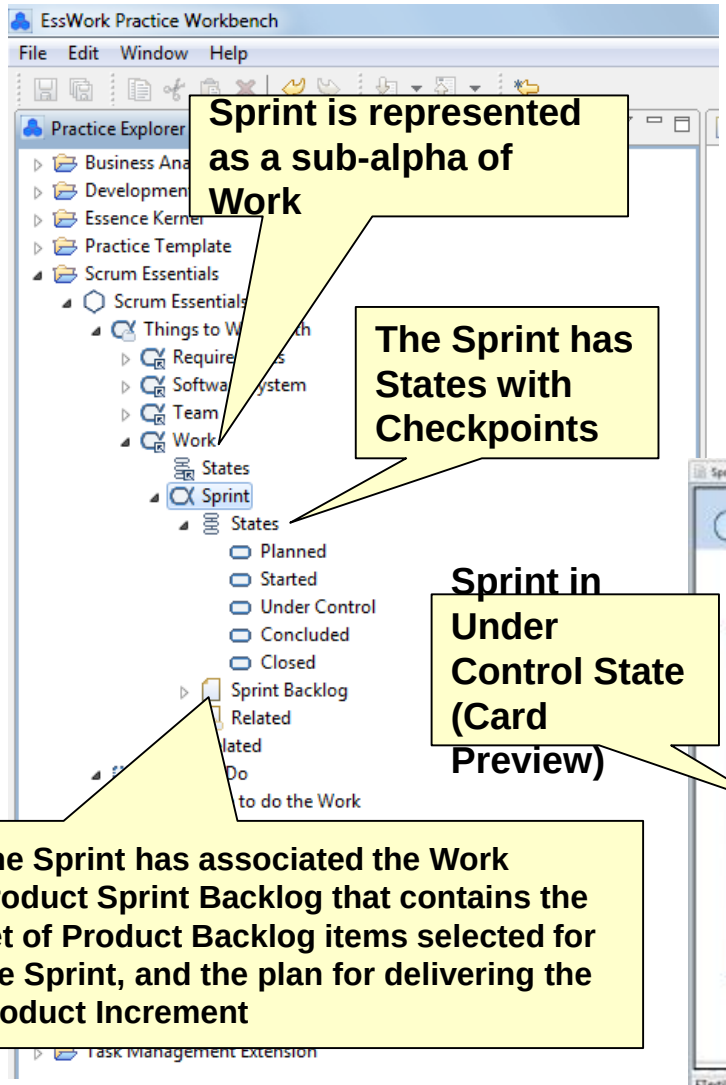
Drag and drop the relevant Alphas to extend from the Essence Kernel into the Scrum practice project

Drag and drop the relevant Activity Spaces to extend from the Essence Kernel into the Scrum practice project

Scrum Roles



Scrum Sprint



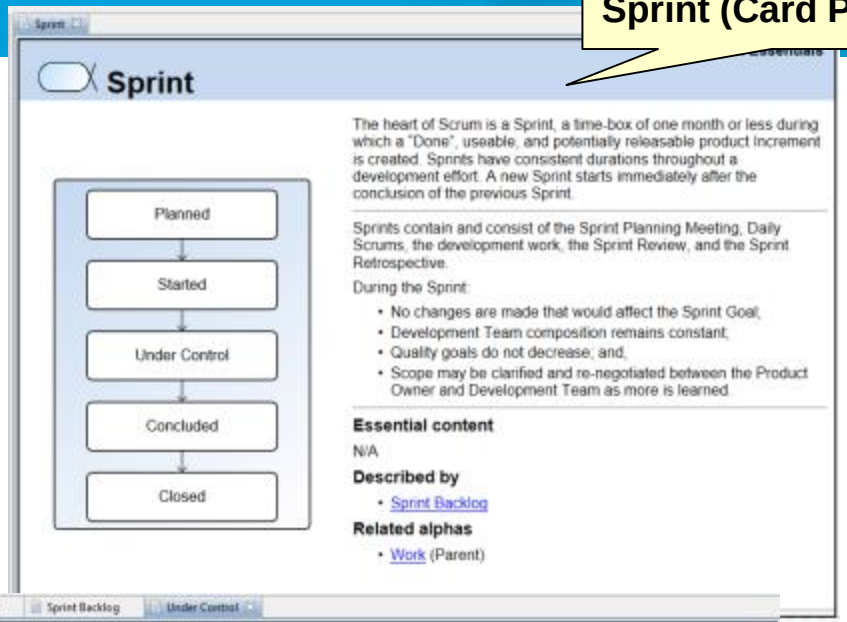
Sprint is represented as a sub-alpha of Work

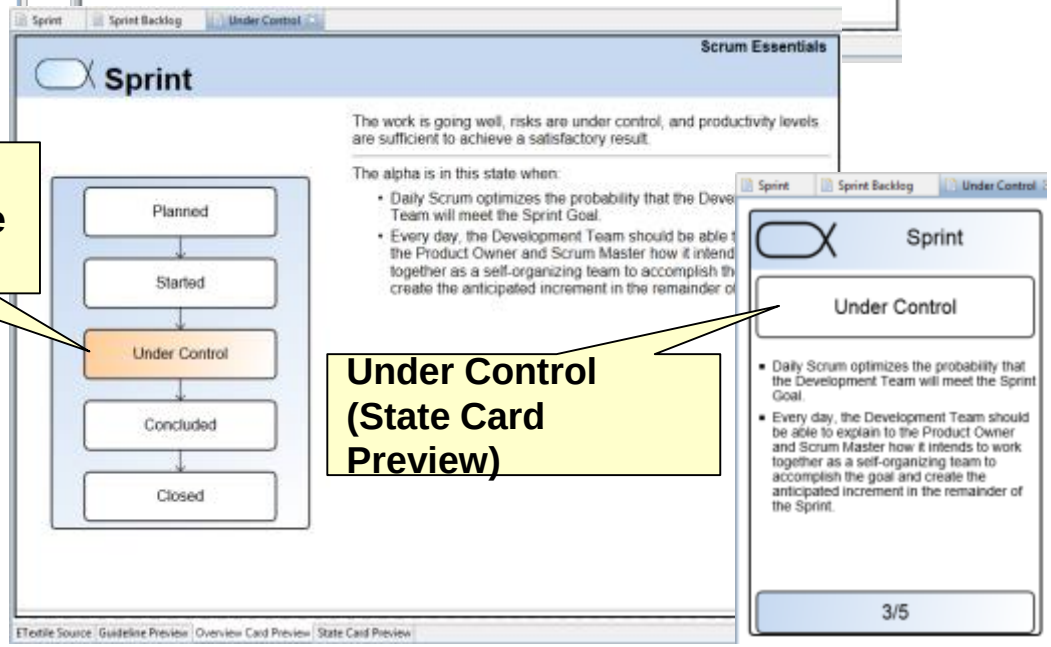
The Sprint has States with Checkpoints

Sprint in Under Control State (Card Preview)

The Sprint has associated the Work Product Sprint Backlog that contains the set of Product Backlog items selected for the Sprint, and the plan for delivering the product Increment

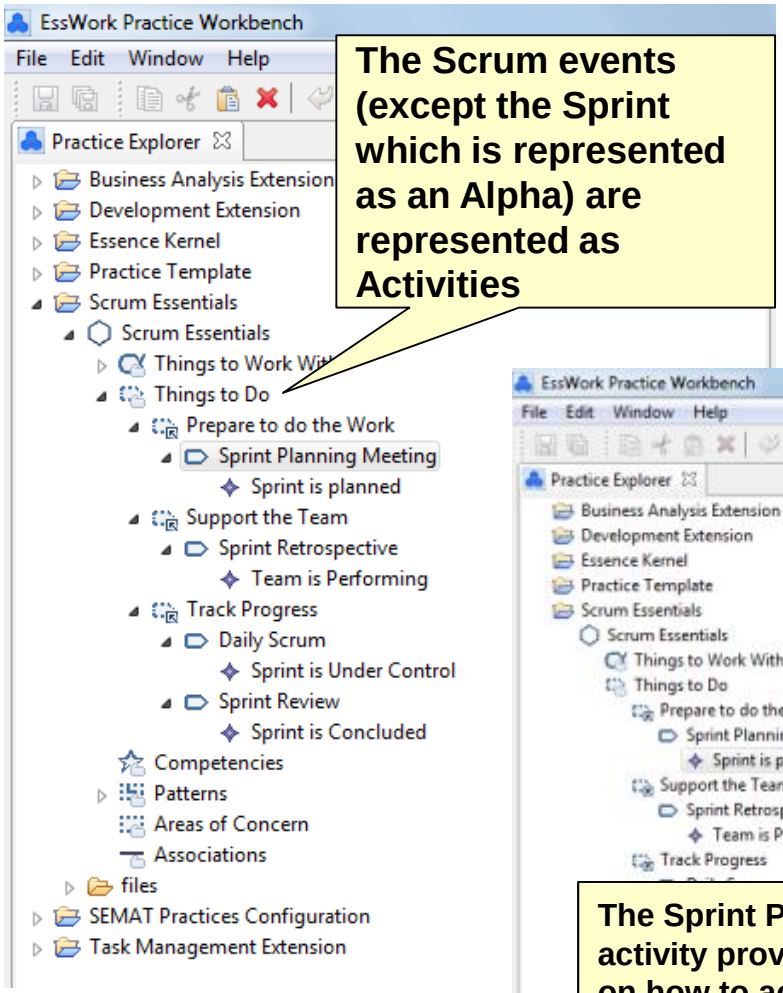
Sprint (Card Preview)



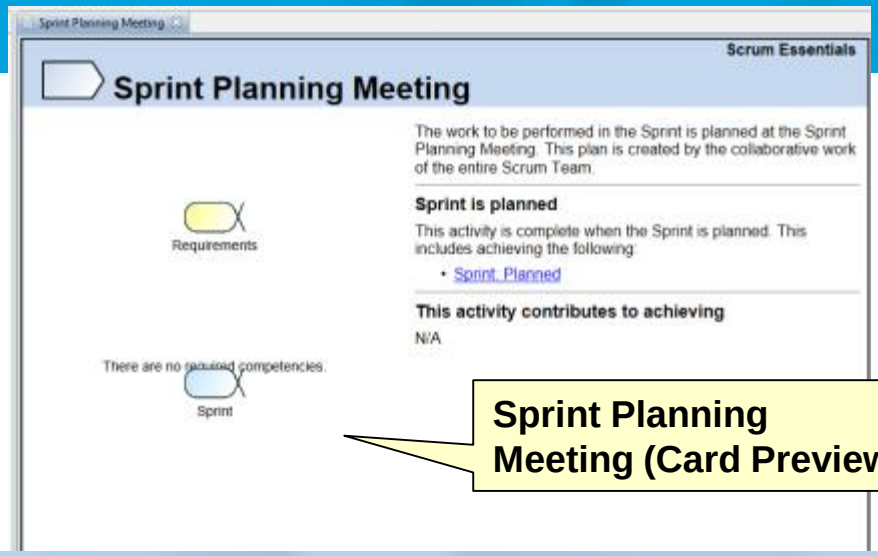


Under Control (State Card Preview)

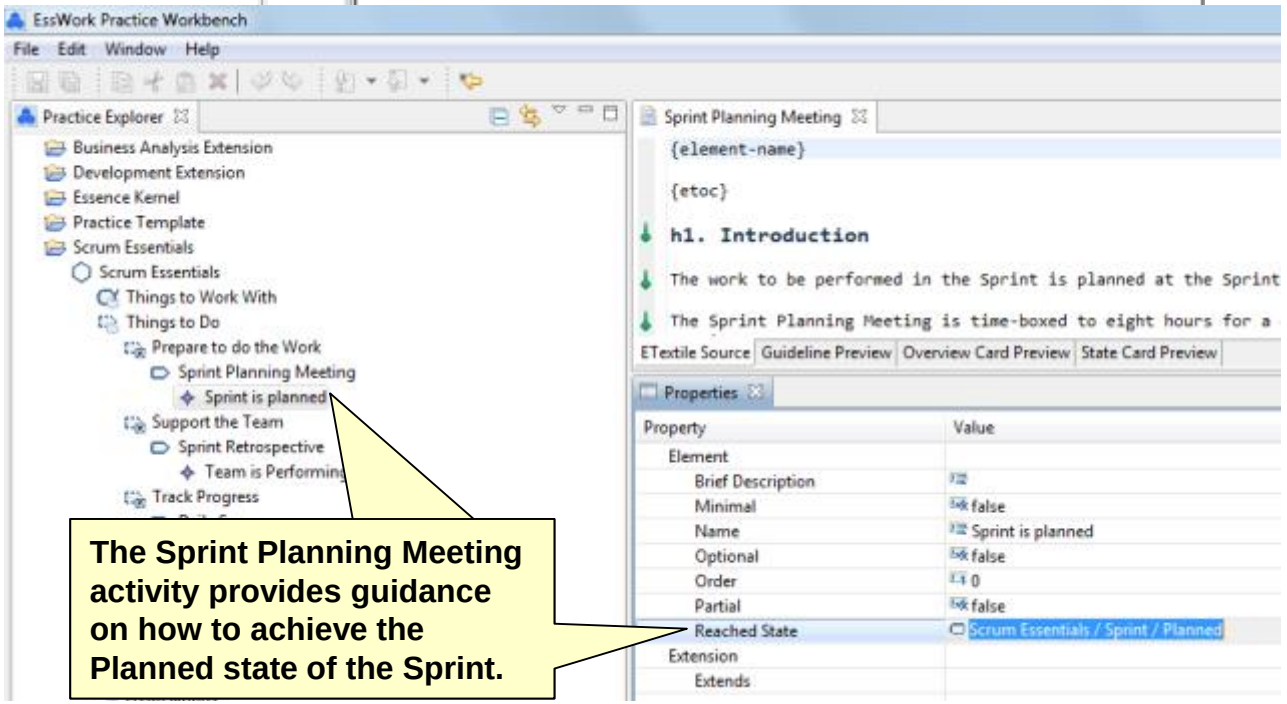
Scrum Activities



The Scrum events (except the Sprint which is represented as an Alpha) are represented as Activities



Sprint Planning Meeting (Card Preview)



The Sprint Planning Meeting activity provides guidance on how to achieve the Planned state of the Sprint.

Questions

- Email:
  - brian.elvesater@sintef.no
- OMG website:
  - http://www.omg.org
- SEMAT website:
  - http://www.semat.org