IVAR JACOBSON INTERNATIONAL

DiVetro analyse • sourcing • management

THE SMARTER WAY

Use Cases – Agility included (UC2.0)

Module 0 - Introduction

Course Objectives

To understand:

- the principles behind just-enough and just-in-time
- the principles behind use-case modelling
- how to capture requirements in a use-case model
- how to build a product backlog using use cases
- how to slice use cases to create well-defined pieces of work for agile teams
- how to manage scope using use cases
- where actors and use cases come from
- how to write use-case narratives
- how to iteratively develop the use-case narratives
- how to manage detail
- when and where to use optional features such as include and extend

Course Outline

- Module 1 – Fundamentals of documentation
- Module 2 – Fundamentals of use case modelling
- Module 3 – Building a backlog with use cases
- Module 4 – Writing a use case narrative
- Module 5 – Adding details to narrative
- Module 6 – Structuring the narrative
- Module 7 – Use case relationships
- Module 8 – Wrap up

IVAR JACOBSON INTERNATIONAL

DiVetro analyse • sourcing • management

THE SMARTER WAY

Use Cases – Agility included (UC2.0)

Module 1 – Fundamentals of Documentation

Objectives

- Understand the agile mindset regarding documentation
- Understand the different types of documentation
- Understand potential stakeholders for documentation
- Understand indicators that influence the need for documentation

Manifesto for Agile Software Development

Individuals and interactions

over Process and tools

Working software

over comprehensive documentation

Customer collaboration

over Contract negotiation

Responding to change

over Following a plan

Manifesto for Agile Software Development

Individuals and interactions

over Process and tools

Working software

over comprehensive documentation

Customer collaboration

over Contract negotiation

Responding to change

over Following a plan

Documentation

1. Requirements – Statements that identify attributes, capabilities, characteristics, or qualities of a system. This is the foundation for what will be or has been implemented.
2. Architecture/Design – Overview of software. Includes relations to an environment and construction principles to be used in design of software components.
3. Technical – Documentation of code, algorithms, interfaces and APIs.
4. End user – Manuals for the end-user, system administrators and support staff.
5. Marketing – How to market the product and analysis of the market demand.

WIKIPEDIA
The Free Encyclopedia

Stakeholders for requirements

REQUIREMENTS

STAKEHOLDERS
- Business
- Security / Safety
- Maintenance
- Support
- …

TEAM
- Developers
- Testers
- …

TEAM(S)
- Feature teams
- In or external
- …

The need for documentation

- Complexity
- Number of teams
- Team size
- Lifecycle of solution
- Domain knowledge team

- Legislation / severity of errors
- Geographical locations
- Number of Stories
- Accessibility domain experts (PO)

Summary

- Documentation might (still) be necessary
- The team is not the only stakeholder for documentation

IVAR JACOBSON INTERNATIONAL

DiVetro analyse • sourcing • management

THE SMARTER WAY

Use Cases – Agility included (UC2.0)

Module 8 – Wrap up

Objectives

- Wrap up just-in-time concept
- Wrap up just-enough concept
- Understand the Use Case 2.0 key principles
- More information
- Final logistics

Just-in-time

- Just-in-time could be:
  - Sequential flow of Requirements, Development and Test
  - Parallel flow of Requirements, Development and Test
  - Any other combination
- Just-in-time could be influenced by:
  - Maturity of organization / team in agile processes
  - Organizational structure (hierarchical culture)
- Focus on what you “really” need at a particular moment

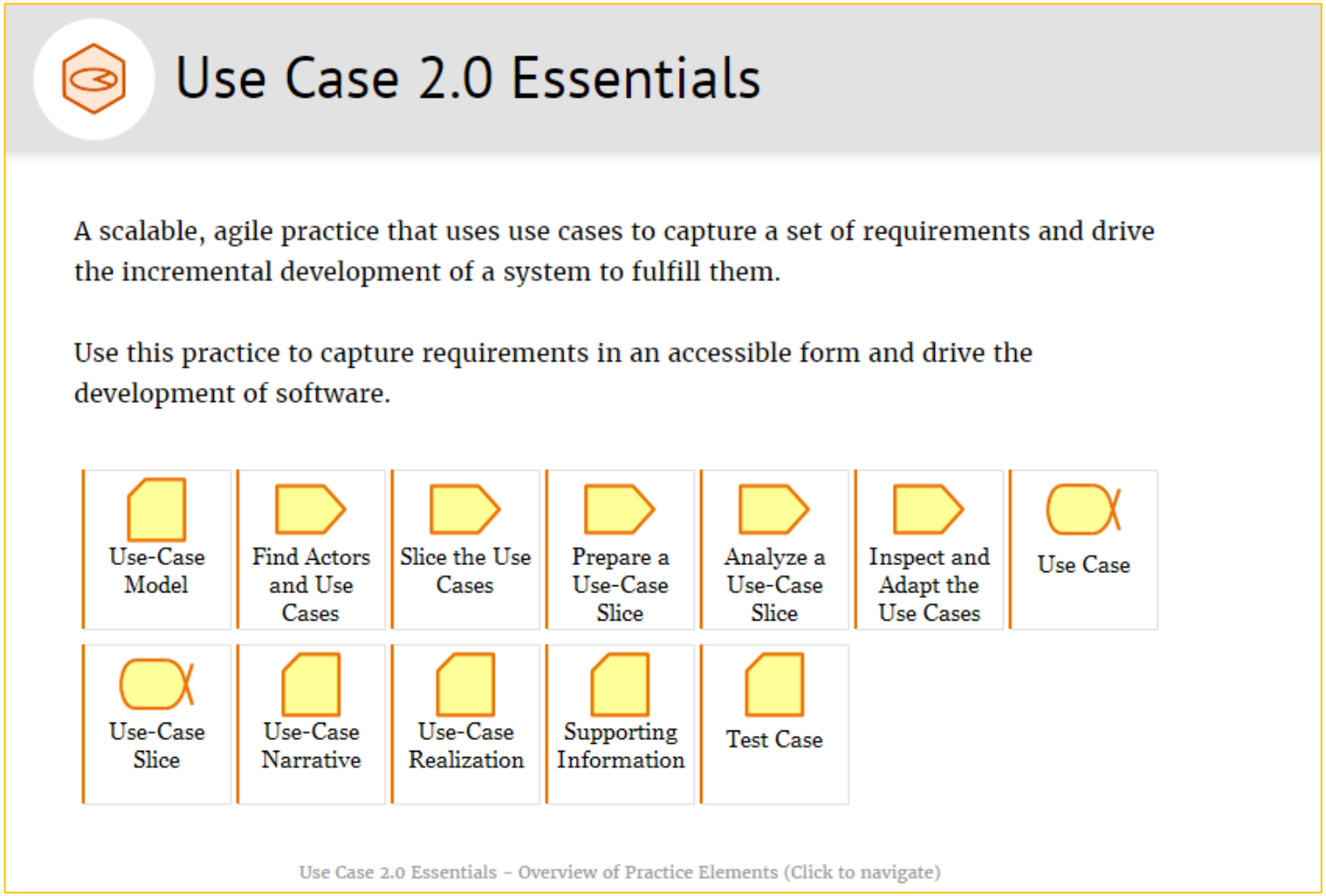
Just-in-time is a subjective concept

Just-in-time and Just-enough

- The level of detail is dependent on the moment that you need it
  - What's needed later, might blur your vision at this moment
  - What's needed now, could well be insufficient at a later moment
- Don't add details because you can

Just-in-time and Just-enough are related

Use Case 2.0



Key principles UC2.0

1. Keep it simple by telling stories
2. Understand the big picture
3. Focus on value
4. Build the system in user stories / slices
5. Deliver the system in increments
6. Adapt to meet the team's needs

Principle 1: Keep it simple by telling stories

- Make requirements understandable and testable by telling stories
- Use cases provide a way to identify and capture all the different but related stories in a simple but comprehensive way.

Basic Flow
- Insert card
- Validate card
- Select cash withdrawal
- Select amount
- Validate funds
- Return card
- Dispense cash

Alternative Flows
AF1 Validate card
AF2 Non standard amount
AF3 Receipt required
AF4 Insufficient funds ATM
AF5 Insufficient funds acc.
AF6 Overdraw
AF7 Card stuck
AF8 Card left behind
Etc…

Summary

- It’s possible to create proper documentation with limited time / effort
- UC2.0 provides the tools to get the right level of details (just-enough)
- Only provide the level of detail that is needed for a specific task (just-in-time)

More information

- divetro.nl/publicaties/analyse/
  - 3 overeenkomsten en 3 verschillen tussen use cases en user stories
  - Agile requirements bestaan niet
  - Requirements zijn geen eenpansmaaltijd
  - Efficiënte analyse: Just Enough in de praktijk
  - Agile en Requirements e-book
  - Etc…
- Use-Case 2.0 ebooks
- https://practicelibrary.ivarjacobson.com/

dennis.geluk@divetro.nl
richard.hilekes@divetro.nl

Meer informatie

- Bezoek onze website:
  - www.divetro.nl/trainingen
- Mail de DiVetro Academy:
  - academy@divetro.nl
- Neem contact op met Harald de Vries:
  - 06 38321489