

Use-Case 2.0

The Lightness of User Stories with the Power of Modeling

Ivar Jacobson



Scaling up, scaling out, zooming in – what is that?



Use cases scale in several dimensions:

- ***Scaling up:***
 - Though Use-Case 2.0 is designed for small teams and small projects, it scales without changing the fundamentals to large organizations and large projects.
- ***Scaling out:***
 - Though Use-Case 2.0 starts with requirements, it scales to many other lifecycle activities such as analysis, design, code, test, user experience, business design, etc.
- ***Zooming in:***
 - Use-Case 2.0 allows you to be as light as you want, focusing on the essentials only, or to zoom in with more and more detail for systems such as telecom or defense systems or more regulated systems such as life-critical systems.

A brief history of use cases



A practice that has learnt to adapt and evolve

A practice that has stood the test of time

Past

'87 – '96

Use-Case Driven Development Paper, OOPSLA, 87

The Objectory Process and Object-Oriented Software Engineering,

Addison Wesley, 1997

UML, OOPSLA, 1995

The Rational Objectory Process,

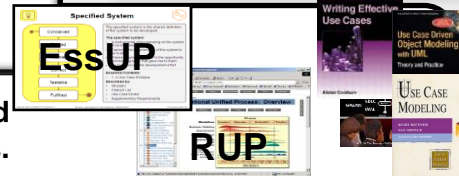
1997

The Unified Software Development Process, Addison Wesley, 1999

'96 - '00

Company X, Y & Z Methods

More and more books and methods using use cases.



Present

And now use-case 2.0...

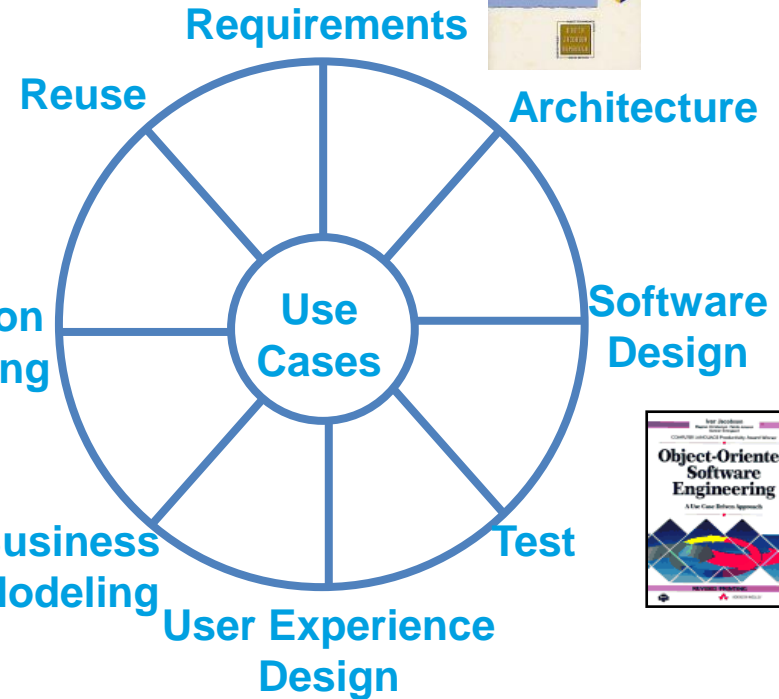


Use-Case Essentials

What made use cases so popular?



- They effectively communicate what a system is supposed to do
- They place the requirements into the context of a specific user's goals
- They are the test cases
- They are the starting point for the design of effective user experiences
- They 'drive' the development through design and code



Use-Case Modeling – A very simple idea.

To get to the heart of what a system must do, you should focus on who, (or what) will use it, and then look at what the system must do for them to help them achieve something useful.

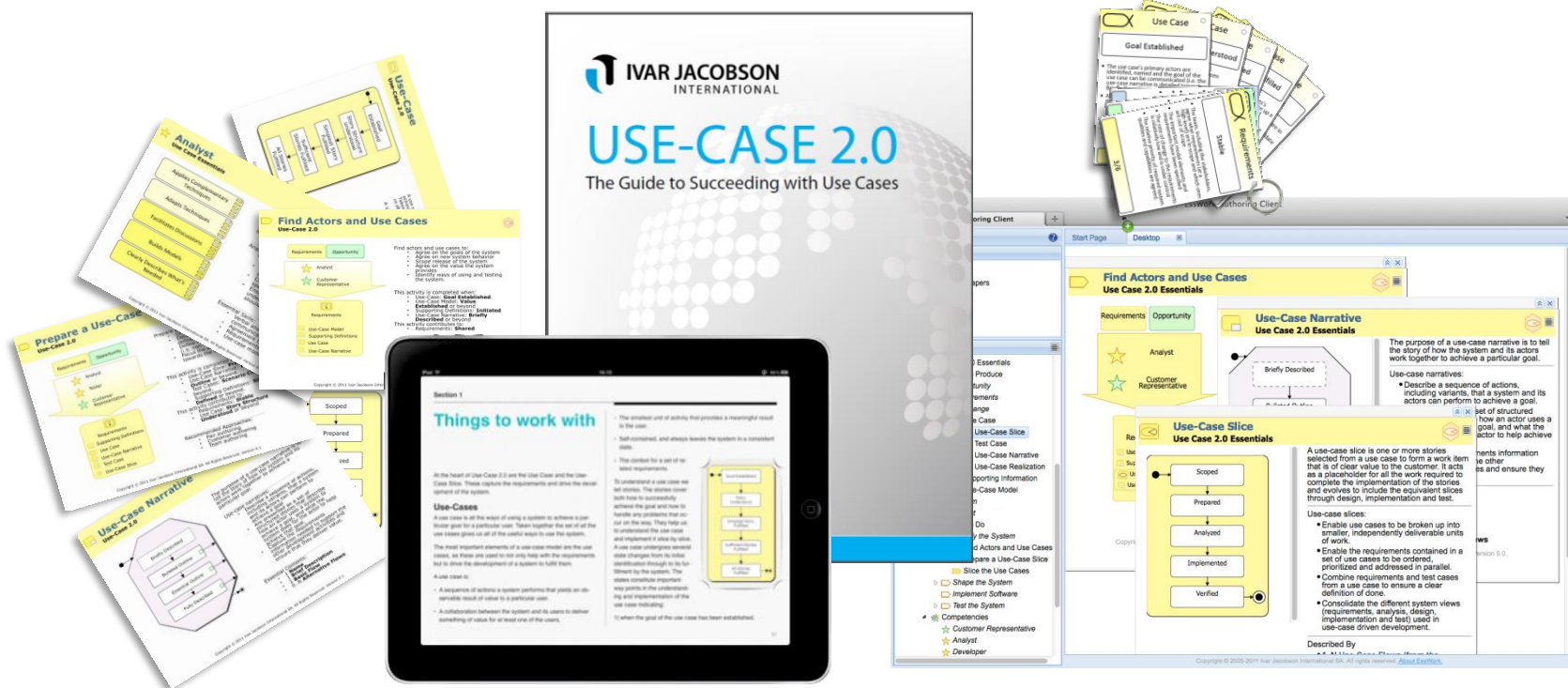


- Focus on Value
- Place the system and the requirements in context
- Package the requirements up into meaningful sub-sets
- Tell the story of what the system will do
- Bring the Business and IT closer together
- Facilitate both release planning and iteration planning
- Are easy for everybody to understand
- Can be identified and outlined very quickly





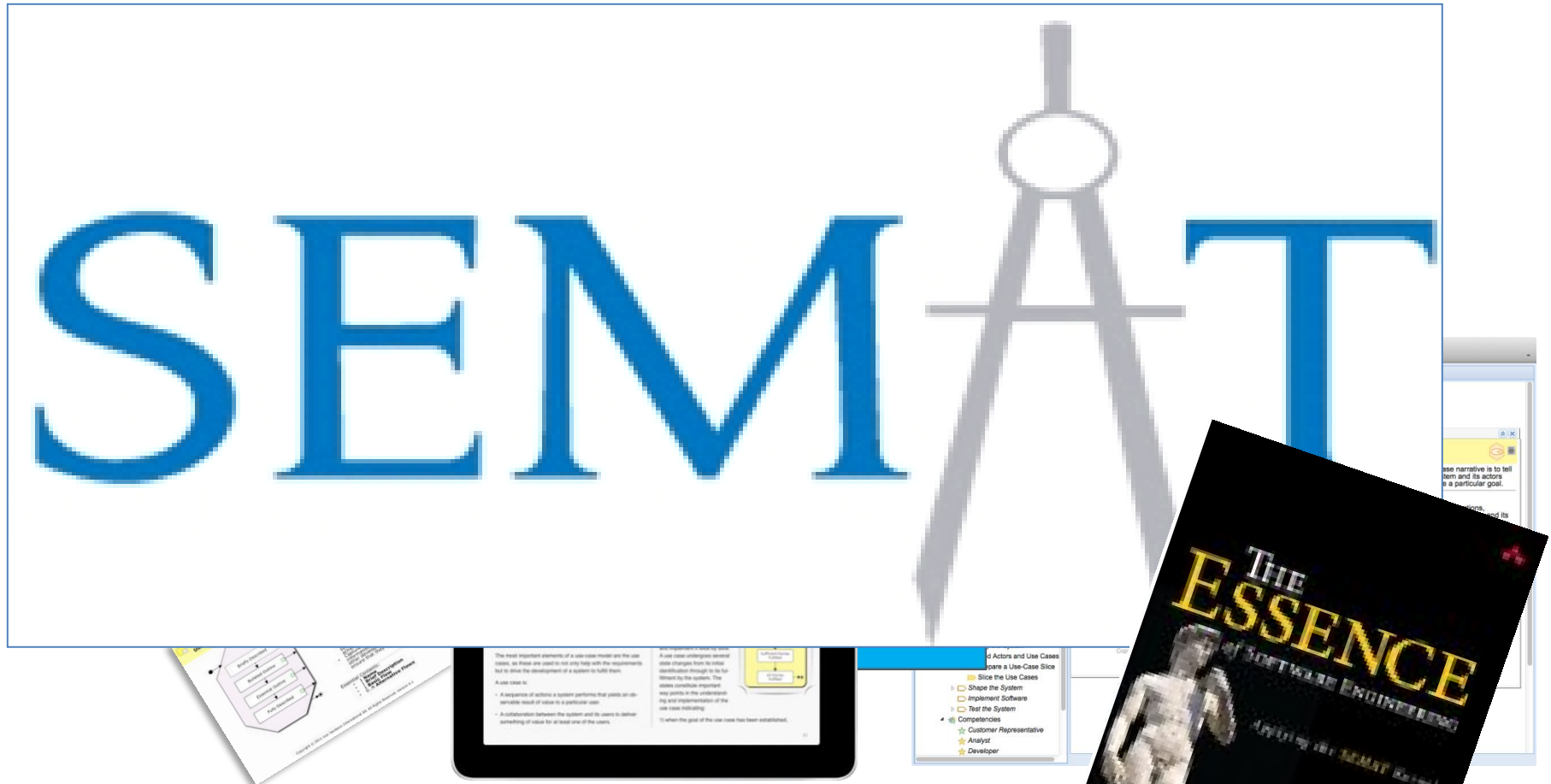
Use-Case 2.0: A scalable, agile practice that uses use cases to capture a set of requirements and drive the incremental development of a system to fulfil them.



Lightweight • Scalable • Versatile • Easy to use



Use-Case 2.0: A scalable, agile practice that uses use cases



Lightweight • Scalable • Versatile • Easy



- To correct some of the common misunderstandings:
 - Use-cases are lightweight **not** heavy-weight
 - Use-cases are stories **not** functions
 - Use-cases are simple **not** complicated
 - Use-cases are for all types of development **not** just green field application development
- To re-focus on the essentials
- To better support innovations and improvements such as test-driven development, Kanban, and Scrum

Use-Case 2.0

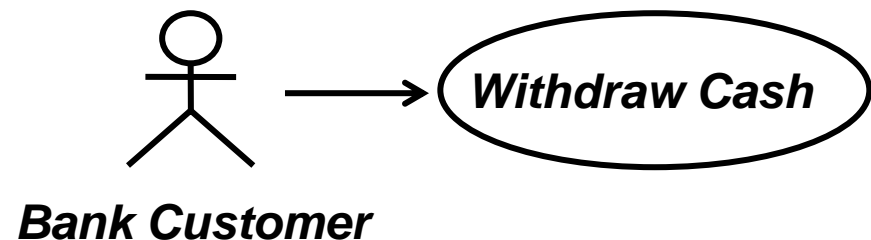
Scaling up, scaling out, scaling in.

The lightness of user stories with the power of modelling.



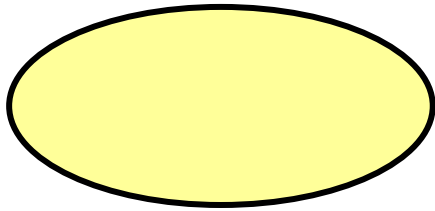
A use case is all the ways of using a system to achieve a particular goal for a particular user.

- Use cases can be shown in UML diagrams
- Use cases are described as narratives
 - Which tell the story of how the system and its users work together to achieve a particular goal

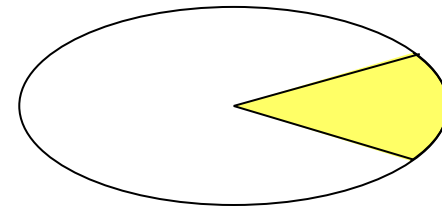


Use-Case Modeling – A very simple idea.

To get to the heart of what a system must do, you should focus on who, (or what) will use it, and then look at what the system must do for them to help them achieve something useful.



A Use Case

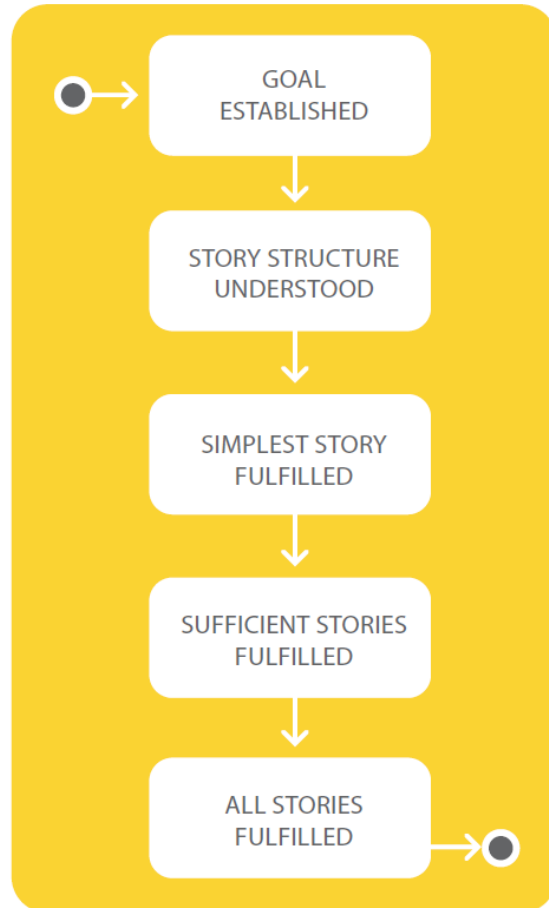


A Use-Case Slice

- Is described by a set of structured stories in the form of:
 - A use-case narrative containing flows and special requirements
 - And a set of matching Test Cases
- Is created by selecting one or more stories for implementation
- **..., acts as a placeholder for all the work required to complete the implementation of the stories**
- ..., and evolves to include the equivalent slices through design, implementation and test.



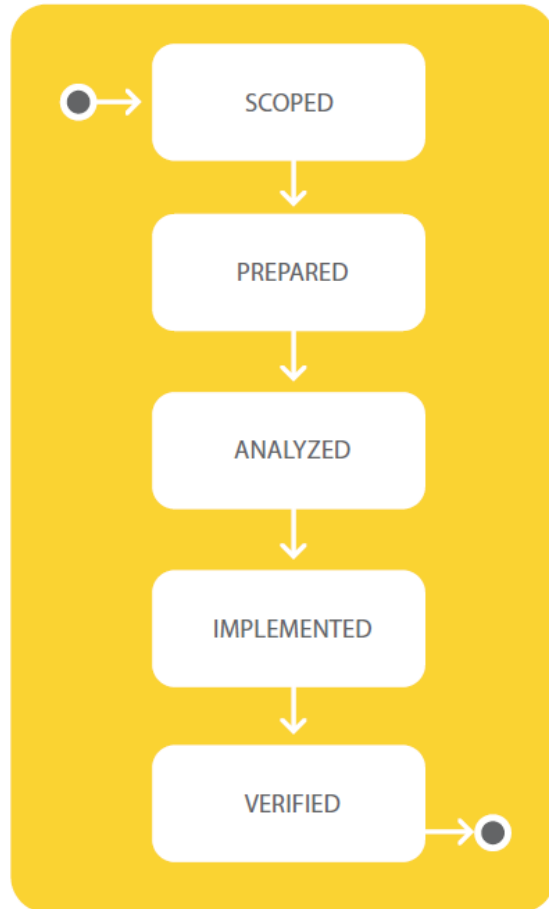
- A use case is:
 - A sequence of actions a system performs that yields an observable result of value to a particular user.
 - That specific behaviour of a system, which participates in a collaboration with a user to deliver something of value for that user.
 - The smallest unit of activity that provides a meaningful result to the user.
 - The context for a set of related requirements.
- To understand a use case we tell stories.
 - The stories cover both how to successfully achieve the goal and how to handle any problems that occur on the way.



- **Goal Established:** when the goal of the use case has been established.
- **Story Structure Understood:** when the structure of the use-case narrative has been understood enough for the team to start work identifying and implementing the first use-case slices.
- **Simplest Story Fulfilled:** when the system fulfils the simplest story that allows a user to achieve the goal.
- **Sufficient Stories Fulfilled:** when the system fulfils enough of the stories to provide a usable solution.
- **All Stories Fulfilled:** when the system fulfils all the stories told by the use case.

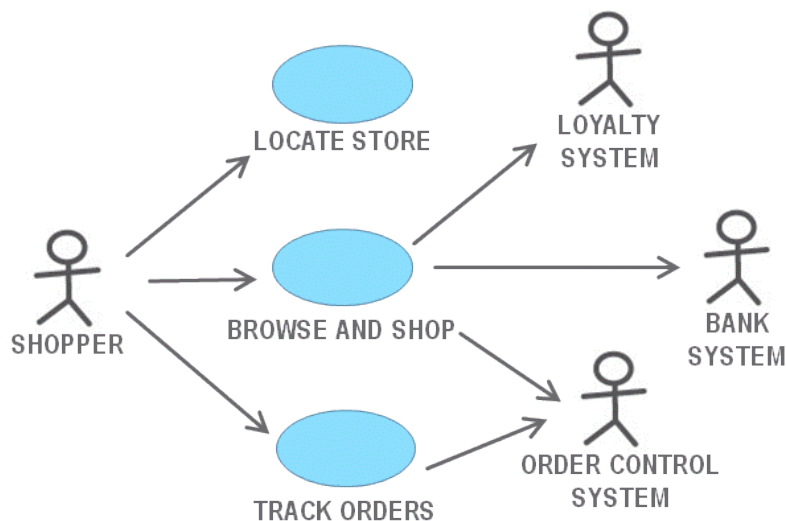


- Use-case slices:
 - Enable use cases to be broken up into smaller, independently deliverable units of work.
 - Enable the requirements contained in a set of use cases to be ordered, prioritized and addressed in parallel.
 - Link the different system models (requirements, analysis, design, implementation and test) used in use-case driven development.



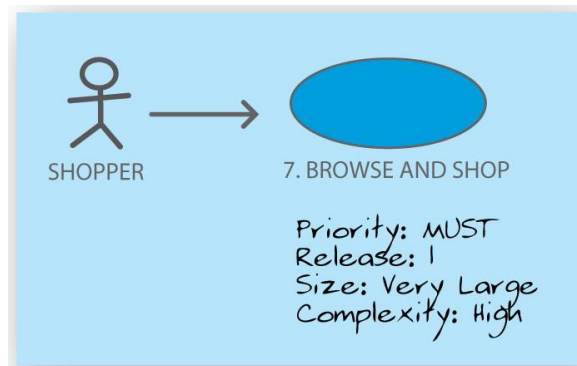
- **Scoped:** when it has been scoped and the extent of the stories covered has been clarified.
- **Prepared:** when the slice has been prepared by enhancing the narrative and test cases to clearly define what it means to successfully implement the slice.
- **Analyzed:** when the slice has been analyzed so its impact on the components of the system is understood and the pieces affected are ready for coding and developer testing.
- **Implemented:** when the software system has been enhanced to implement the slice and the slice is ready for testing.
- **Verified:** and finally when the slice has been verified as done and is ready for inclusion in a release.

Use Cases and Use-Case Slices



1. Create a use-case model to understand the big picture.

2. Select the use-case you want to work with and slice to drive your iterations.



A USE CASE AND ITS PROPERTIES
CAPTURED ON A POST-IT NOTE

7.1 Select and Buy
1 Product

Flows: BF
Test: 1 Product,
default payment,
valid details

5

7.3 Support
Systems Unavailable

Flows: BF, Ag, A10,
All, A12
Test: Select Product,
Provide Information,
Disconnect each system
in between

13

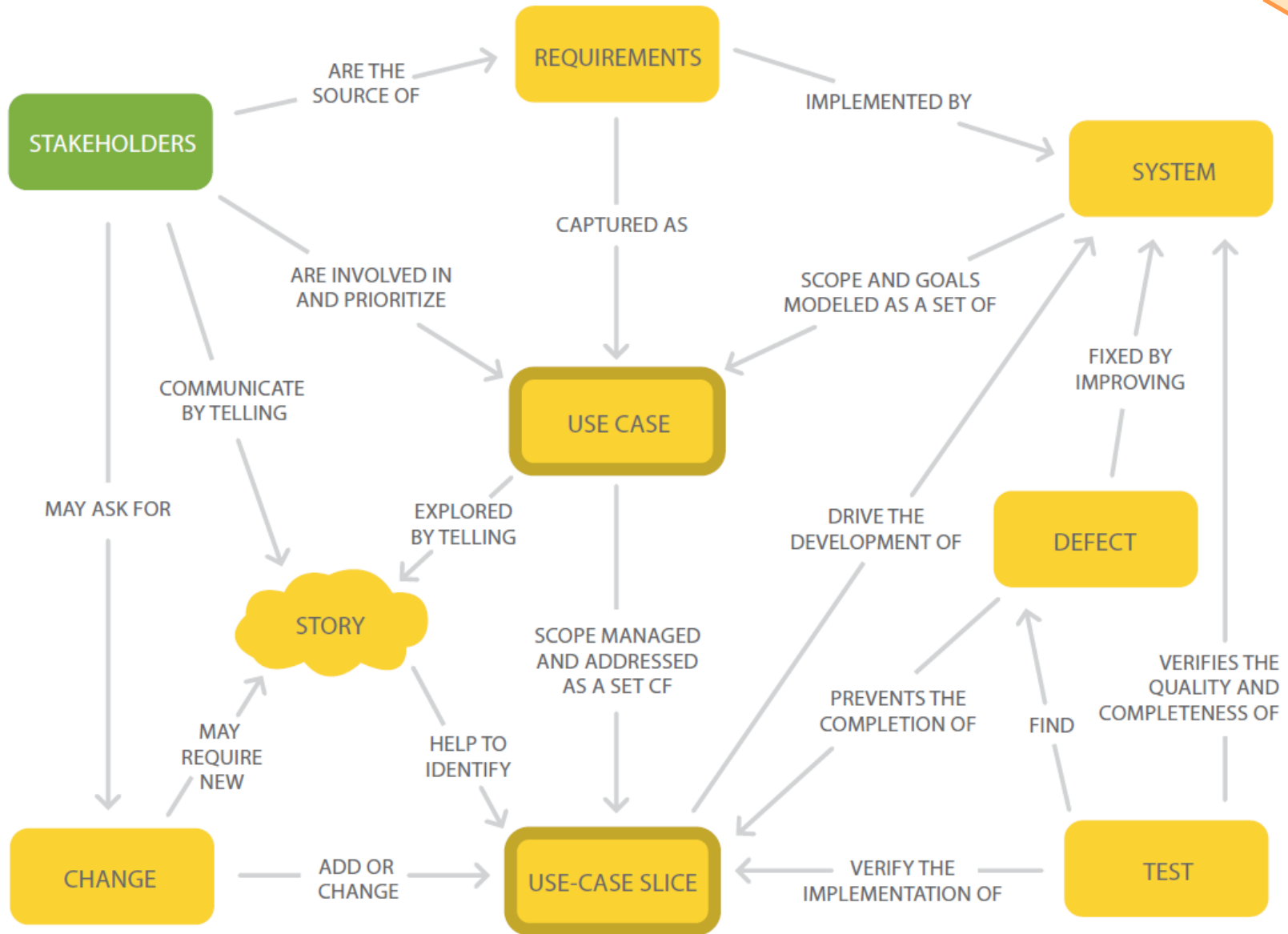
7.2 Select and Buy
100 Products

Flows: BF
Test: 100 Products,
default payment,
valid details

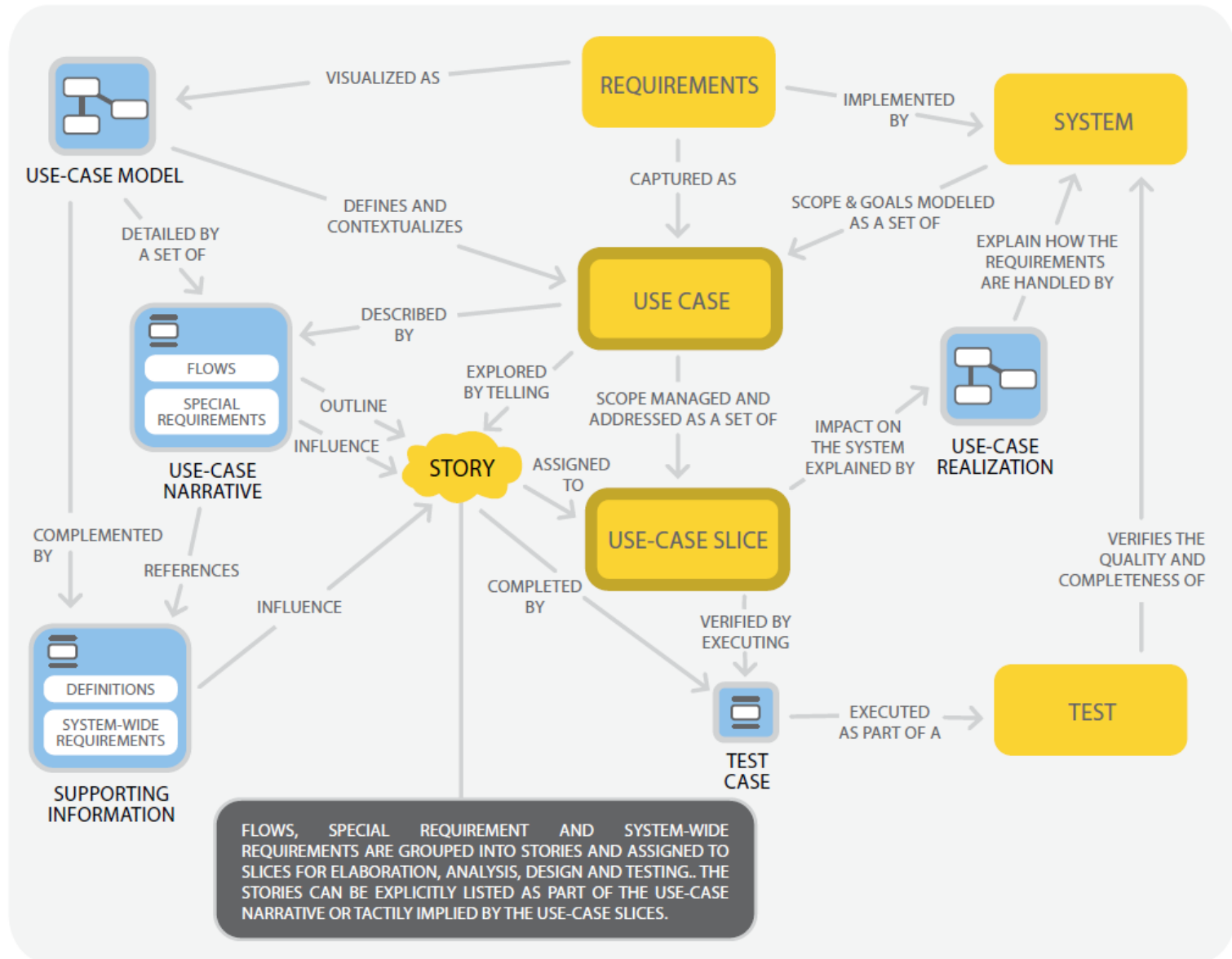
5

SOME SLICES
FROM THE USE CASE
CAPTURED ON THEIR
OWN POST-IT NOTES

Use Case 2.0 concept map



Work Products



Work product level of detail



- All of the work products are defined with a number of levels of detail.
- Choose the level of detail you need

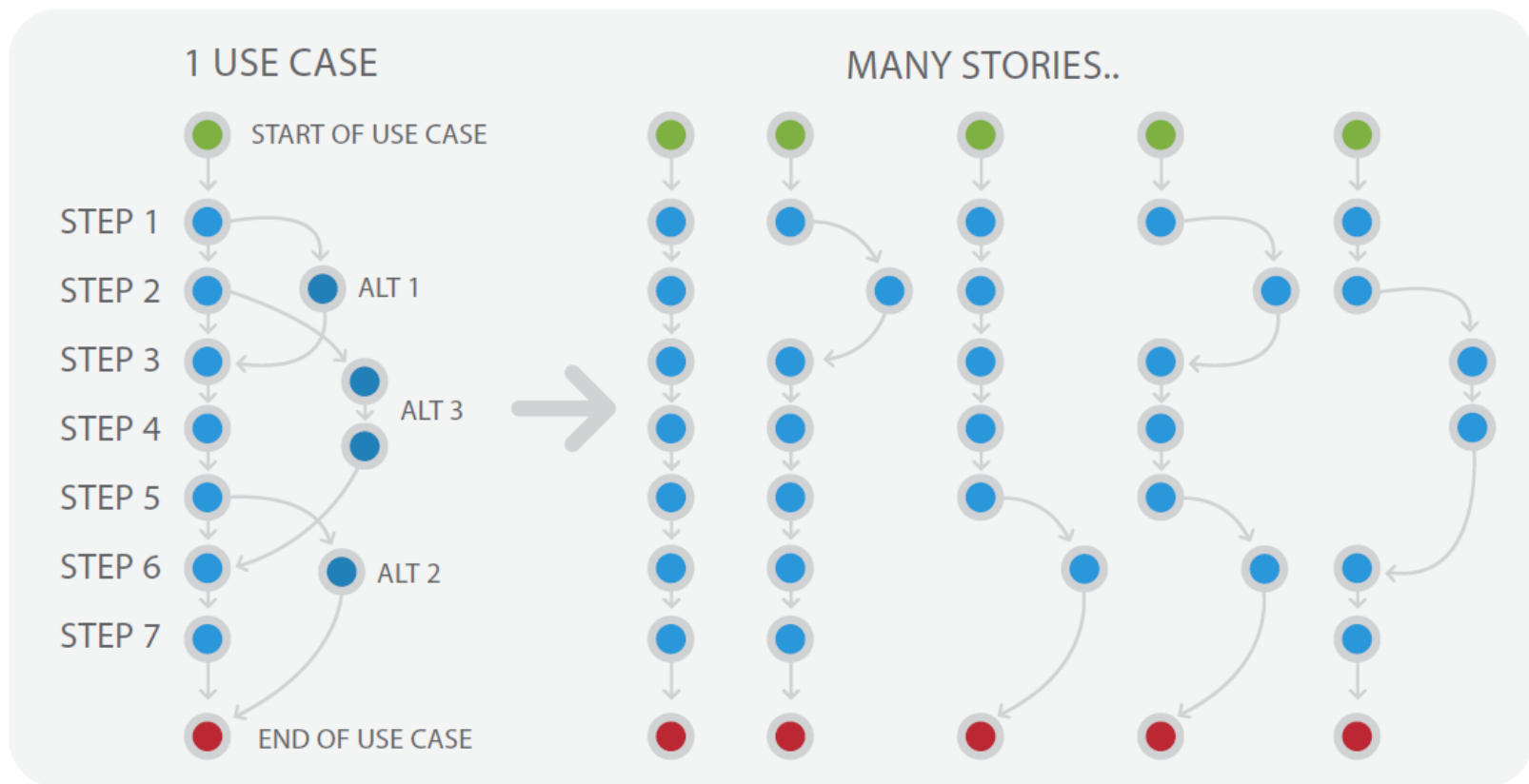
	USE-CASE MODEL	USE-CASE NARRATIVE	USE--CASE REALIZATION	TEST CASE	SUPPORTING INFORMATION
SKETCH:		BRIEFLY DESCRIBED		TEST IDEAS FORMULATED	OUTLINED
BARE ESSENTIALS:	VALUE ESTABLISHED	BULLETED OUTLINE	IMPLEMENTATION ELEMENTS IDENTIFIED	SCENARIO CHOSEN	SIMPLY DEFINED
ENHANCED:	SYSTEM BOUNDARY ESTABLISHED	ESSENTIAL OUTLINE	RESPONSIBILITIES ALLOCATED	VARIABLES IDENTIFIED	MODELLED & ILLUSTRATED
EXPANDED:	STRUCTURED	FULLY DESCRIBED	INTERACTION DEFINED	VARIABLES SET	COMPREHENSIBLY DEFINED
FURTHER EXPANDED:				SCRIPTED /AUTOMATED	

As lightweight as you want, as scalable as you need.

Story telling with use cases



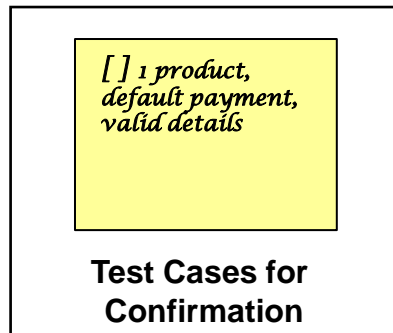
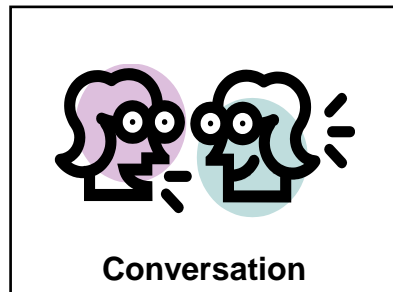
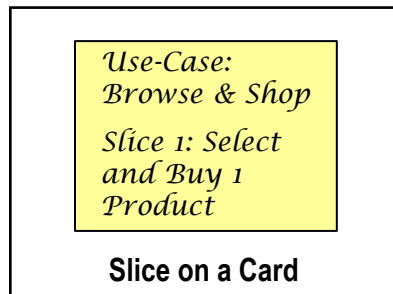
- A story is described by part of the use-case narrative, one or more flows and special requirements, and one or more test cases.



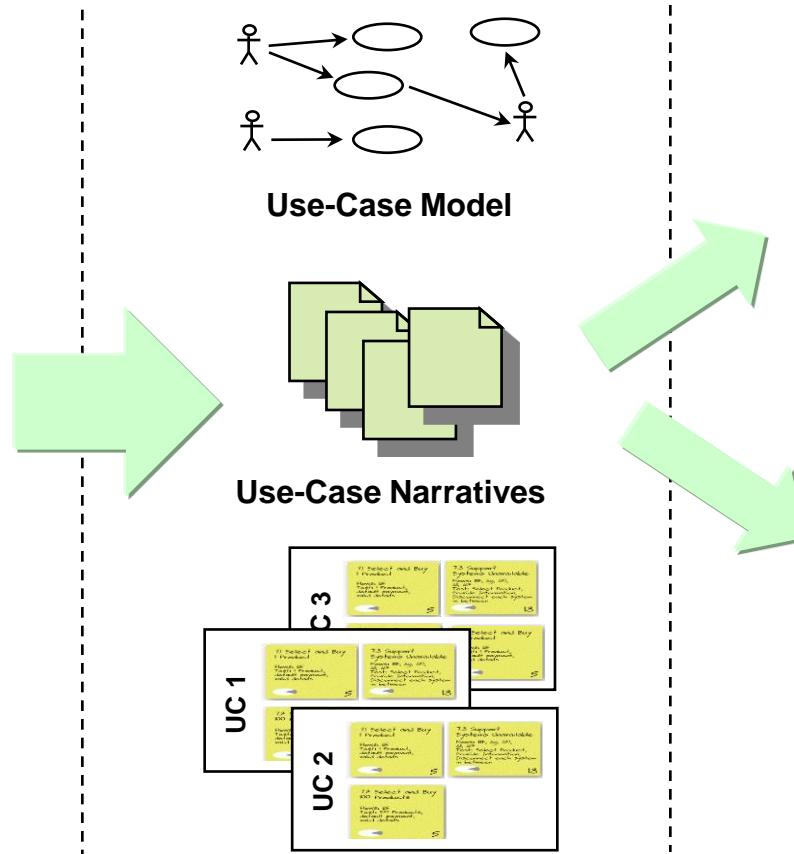


- Name
 - Brief Description
 - Basic Flows
 - Alternate Flows
 - Sub Flows
 - Pre-Post Conditions
 - Special Requirements
 - Public Extension Points
- From use-case model**
- Flow of events**

A truly scalable solution



Start with Simple Cards

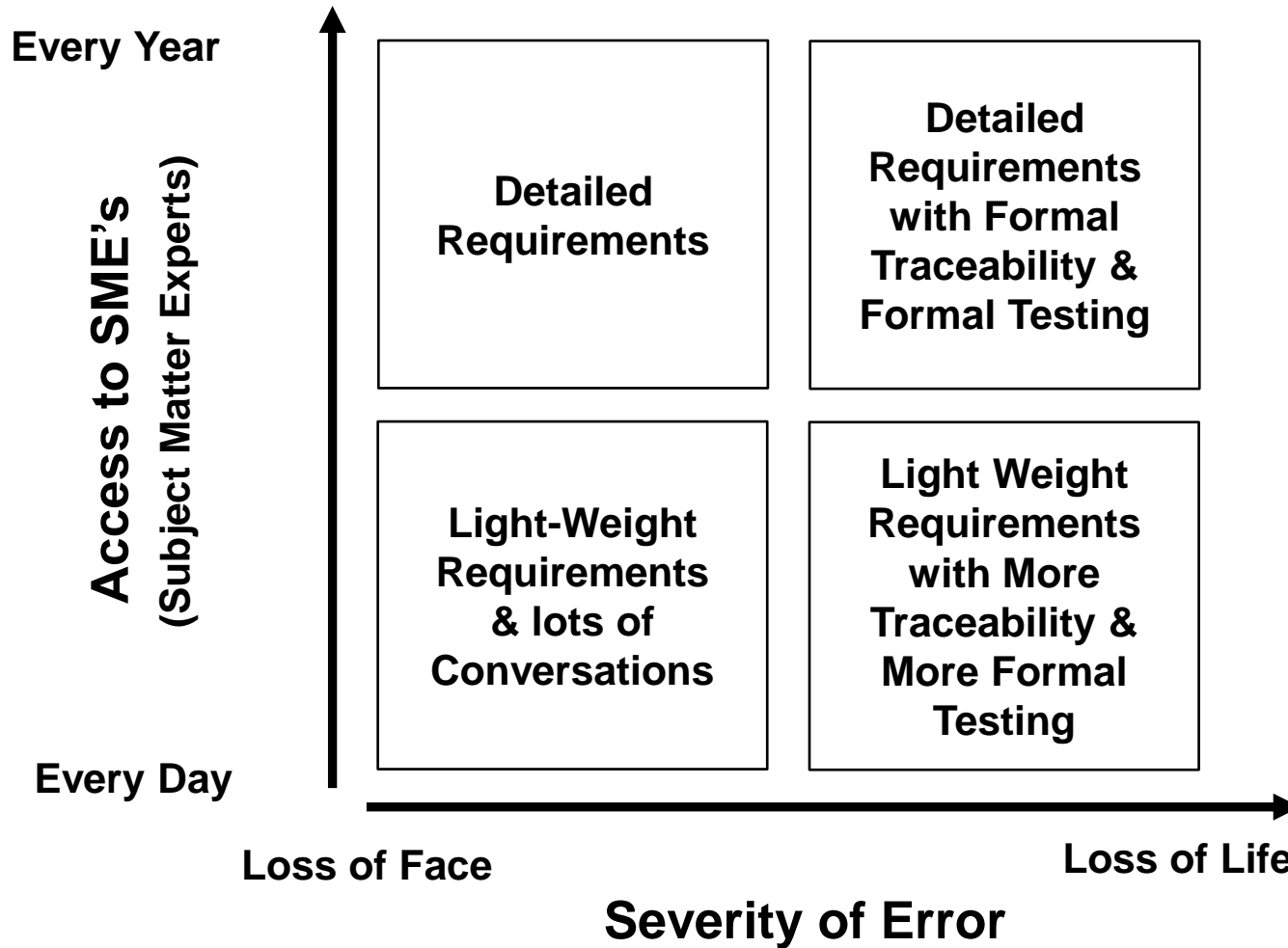


**Handle larger systems with
documented Use-Case Narratives to
complement the Slices**

**Handle distributed teams with
fully described use cases**

**Handle systems-of-
interconnected systems with
recursive use-case models**

Different Situations Require Different Approaches





Stakeholder Analysis

- Stakeholder Mapping
- Stakeholder Needs
- Stakeholder Types
- Persona Modeling

Business Analysis

- Business Process Modeling
- Business Object Modeling
- Business Use-Case Modeling
- Business Rules
- Opportunity Analysis



- Interviews
- Workshops
- Model Storming
- Focus Groups
- Demonstration
- Simulation
- Observation

- Story Boarding
- Prototyping
- Use-Case Realization
- Information Modeling
- Wireframes

Facilitation and Elicitation

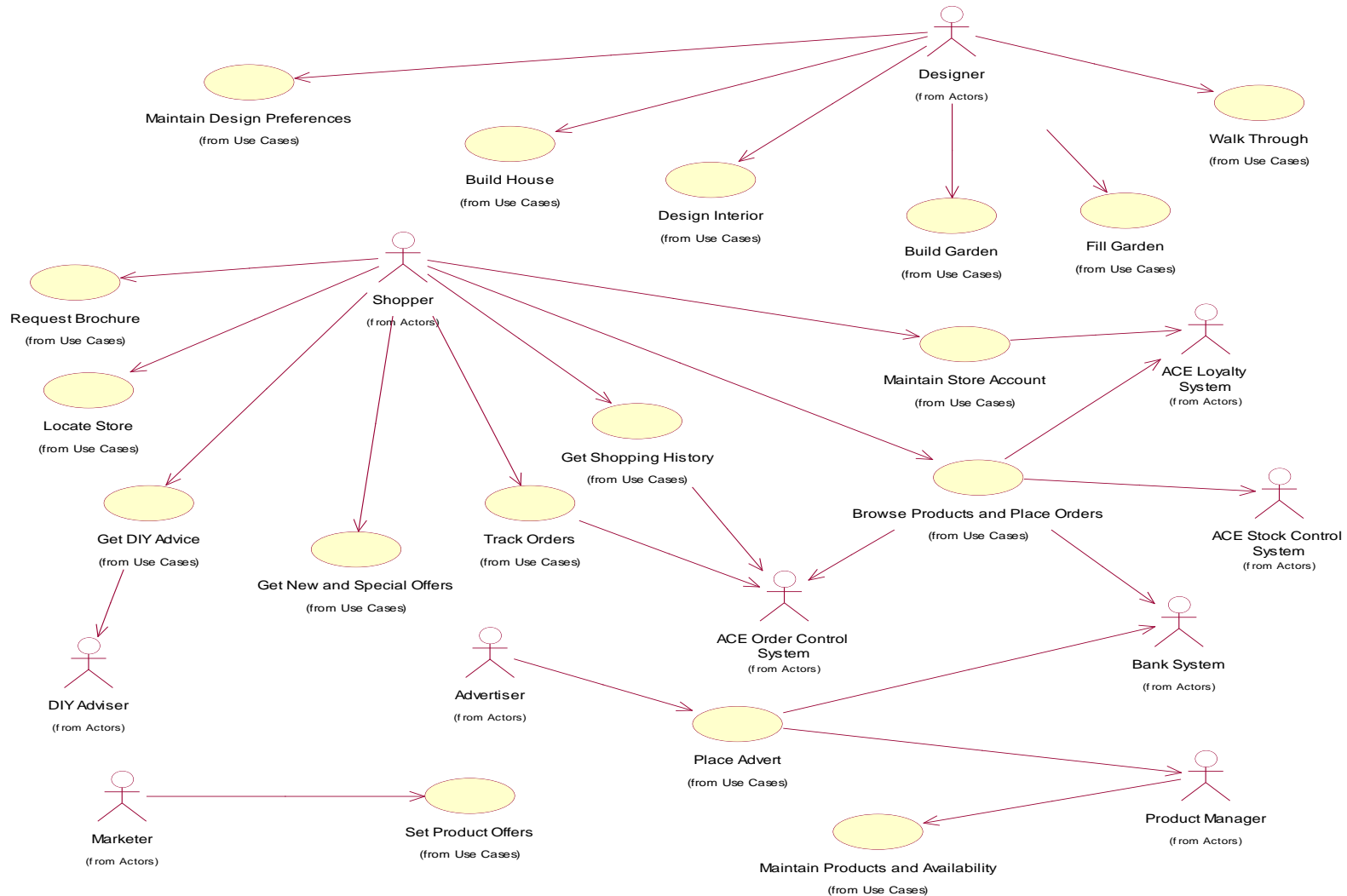
Specification By Example

Use Cases enable agility



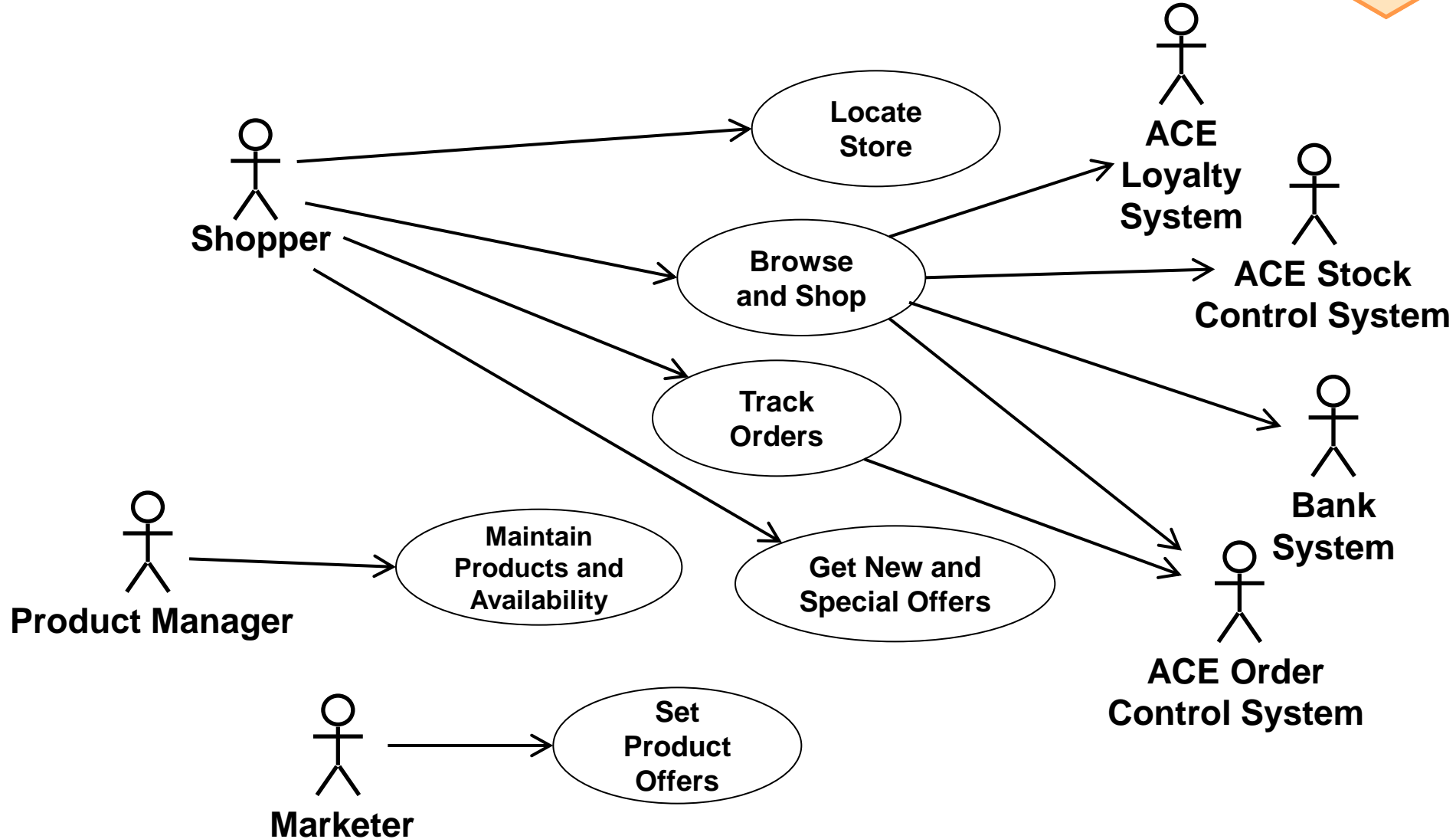
Level of detail	Primary Purpose	Supports
Briefly Described	Identify the use case and summarize its purpose.	<ul style="list-style-type: none">• Basic scope management• Discussions about requirements
Bulleted Outline	Summarize the shape and extent of the use case.	<ul style="list-style-type: none">• Scope management• Low fidelity estimation.• Collaborative test definition• Impact analysis and prototyping.• Component identification
Essential Outline	Summarize the essence of the use case.	<ul style="list-style-type: none">• User Interface design.• Prototyping.• Collaborative, creative analysis and design• Collaborative test definition• High fidelity estimation
Fully Described	Provide a full, detailed requirements specification for the use case.	<ul style="list-style-type: none">• Analysis and design• Implementation and testing• Creation of user documentation.• High fidelity estimation

What is the Big Picture?



What is needed for a basic on-line store?

Agree Scope



ACE DIY Online System

Planning development with use case slices



- The use cases and the use-case slices should also be ordered so that the most important ones are addressed first.





- We don't just slice the use cases.
- The system should be built in slices, each of which has clear value to the users.

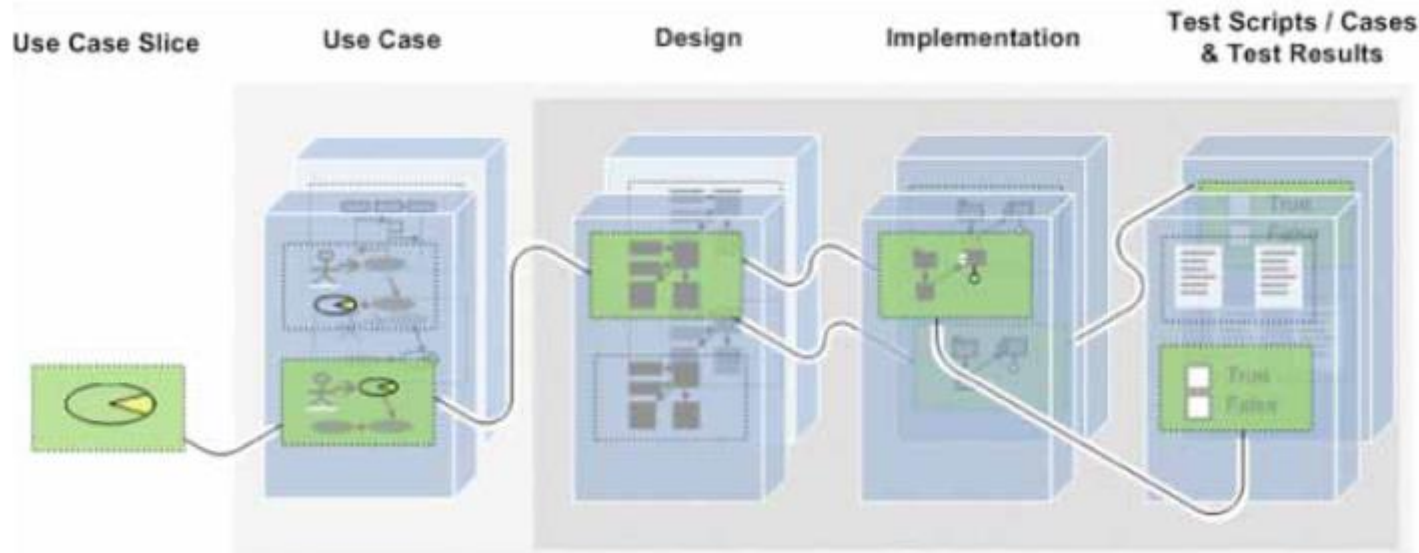
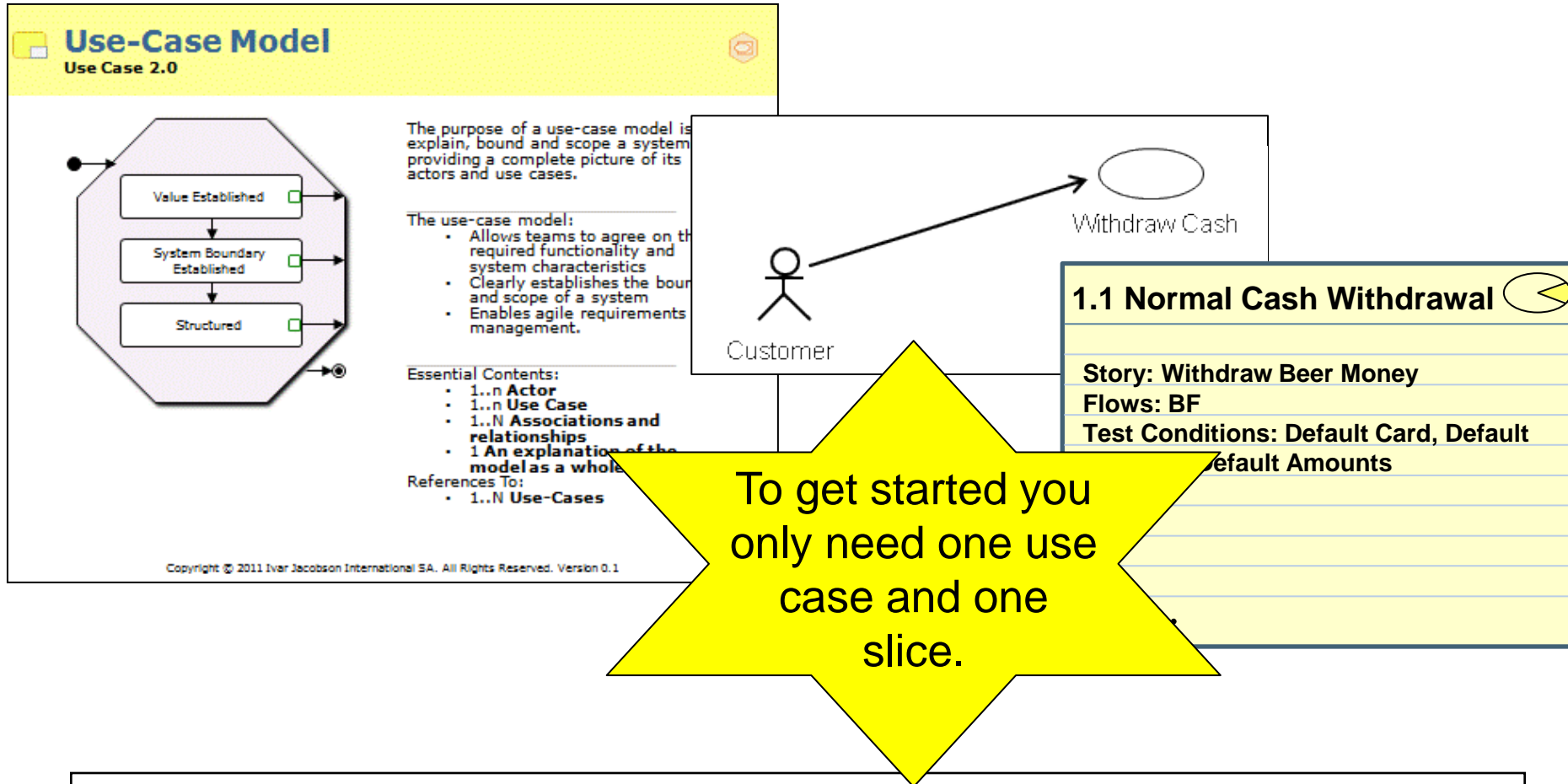


FIGURE 3: A USE-CASE SLICE IS MORE THAN JUST A SLICE OF THE REQUIREMENTS

Getting started is easy.

You only need to model what is important to you



The use-case model provides the big picture needed for effective scope management and release planning.

Summary: Introducing Use-Case 2.0



- Use cases are still use cases
- They provide context for our requirements
- We only model what is important
- We slice our use cases to drive the development
- We eliminate waste by using the lightest level of detail
- We include test cases (as part of the use case) to define done
- We use cards and backlogs to support agile ways-of-working
- We add detail to cope with out-sourcing and off-shoring
- We apply the techniques recursively to handle large projects, programs and business change

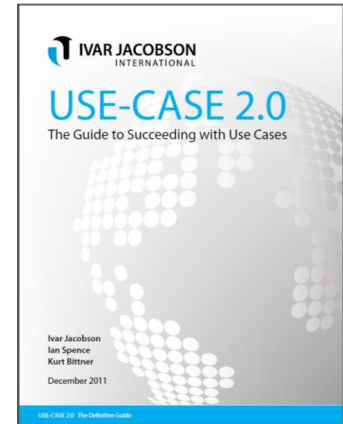
Use-Case 2.0

Lightweight • Scalable • Versatile • Easy to use

Summary: Use Case 2.0 -- Distinctive Features



- It helps you quickly understand the big picture
- As light as you want it to be
- Enabling incremental delivery
- It's not just about requirements, it's for the whole lifecycle
- It's also for non-functional requirements
- It's also for embedded software
- It's not just for software development – it's for business development as well
- Scaling to meet your needs – zooming in, scaling out and scaling up



Ebook:
<http://bit.ly/1aSJD0M>

An effective way to capture and manage your requirements.