IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Use-Case 2.0

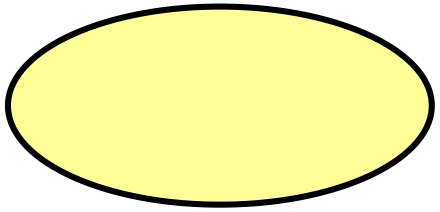
Module 3 – Outlining and Slicing Use Cases

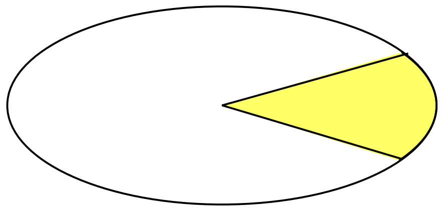
www.ivarjacobson.com

Objectives

- Understand how we progress and complete the use cases
- Understand how use cases drive the development activities
- Understand the role of use-case slices and how to use them
- Understand the importance of use-case outlines

Slicing up use cases to drive the development


A Use Case


A Use-Case Slice

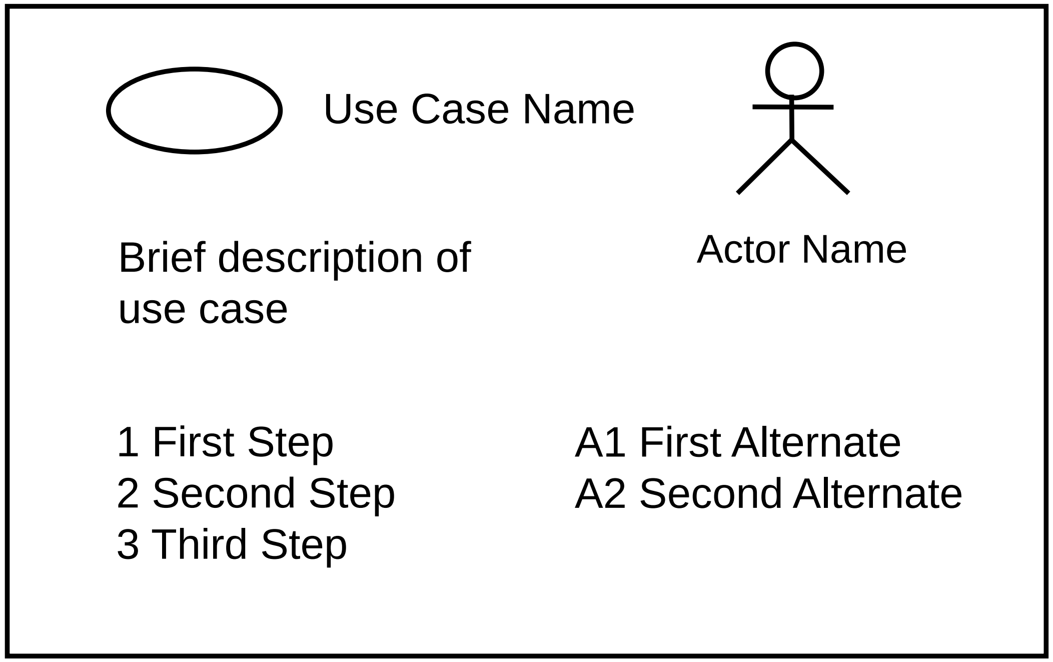
- Is described by a set of structured stories in the form of:
  - A use-case narrative containing flows and special requirements
  - And a set of matching Test Cases

- 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.

First outline the Use Cases

- To get a better idea of the complexity and scale of a use case you should create an outline to complement the brief description

Outlining a Use Case:



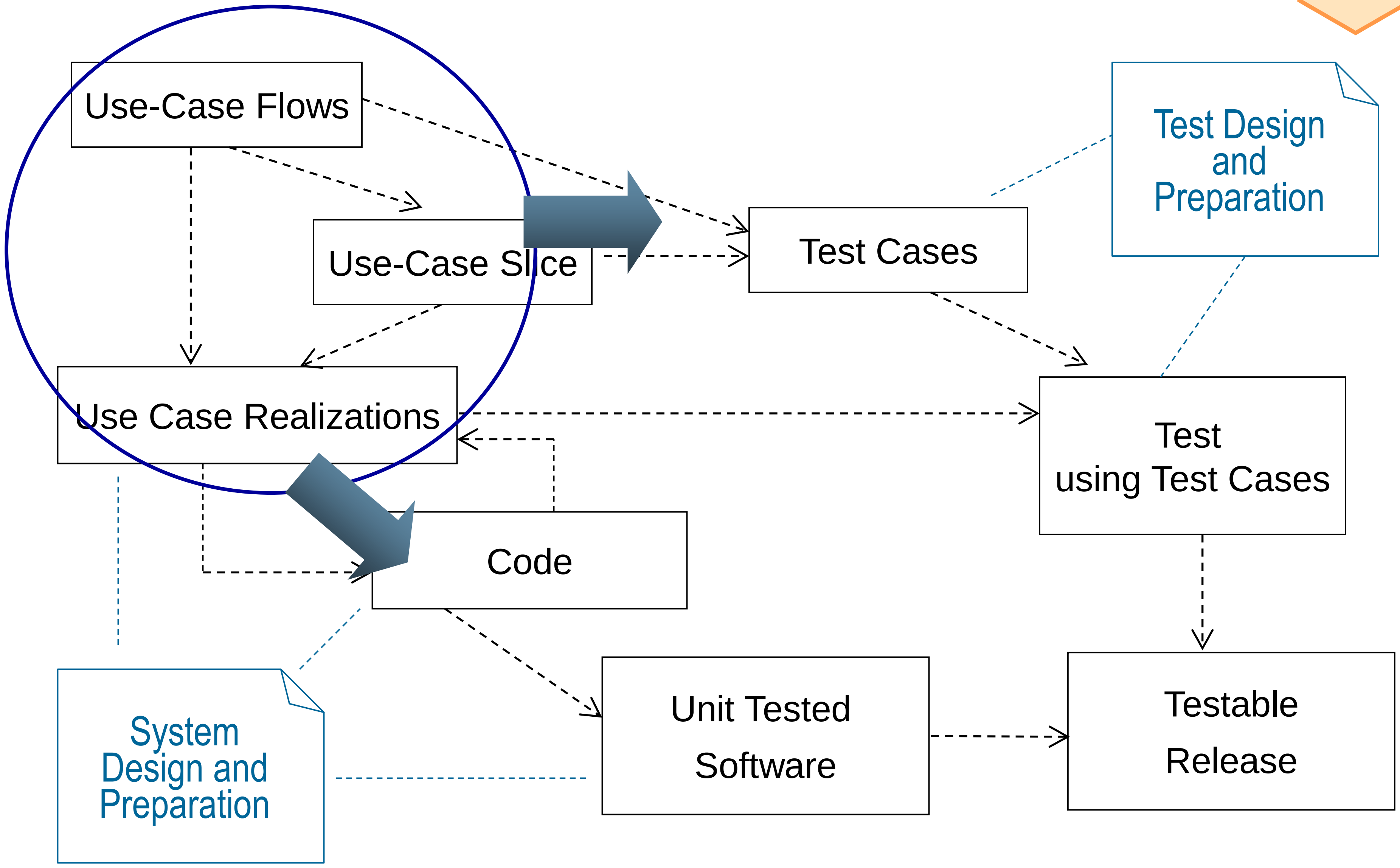
- List the steps in the basic flow
- Write down the actions in order
- Enumerate the steps
- Walk through, name and enumerate the main alternatives

"This is the first step in the evolution of the use-case narrative"

Exercise 3.1: Use-Case Outlines

- In your groups
  - Each select one of the use cases discovered and briefly described in your group exercise
  - Create a use-case outline on flip-chart paper
  - Identify and name at least 3 Alternative flows
  - Each person briefly presents the use-case outline back to the group

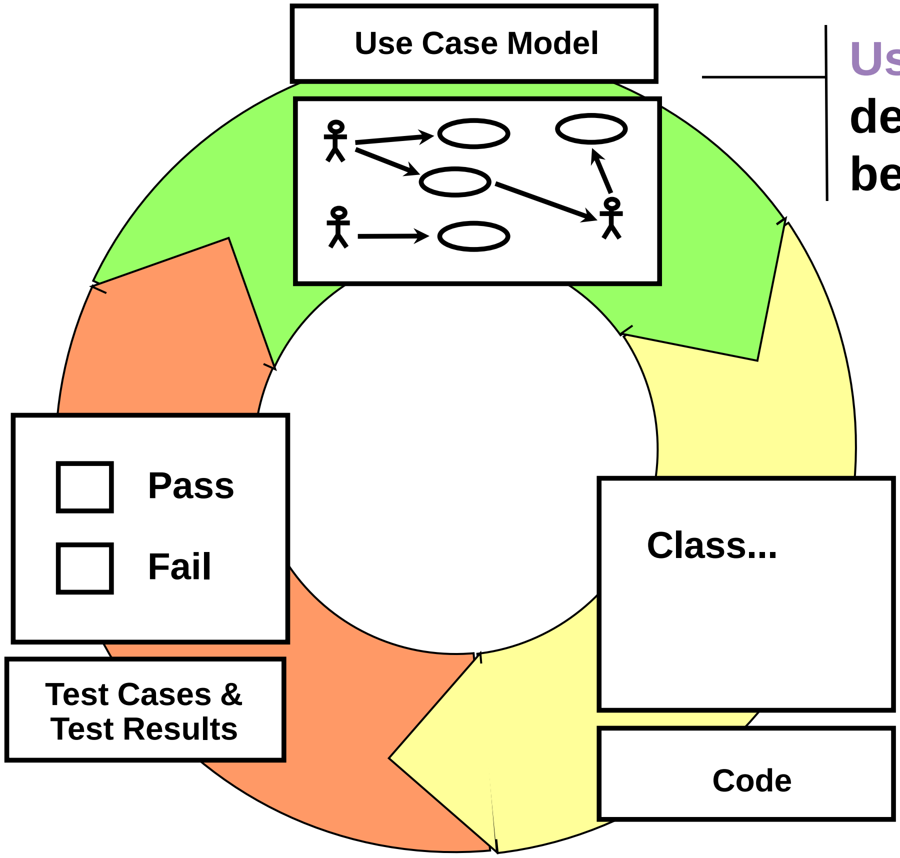
Use Cases Drive Both Development and Testing



IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Use cases and test cases: knowing what to do and when it's done



Use-Case Narratives define what should be implemented

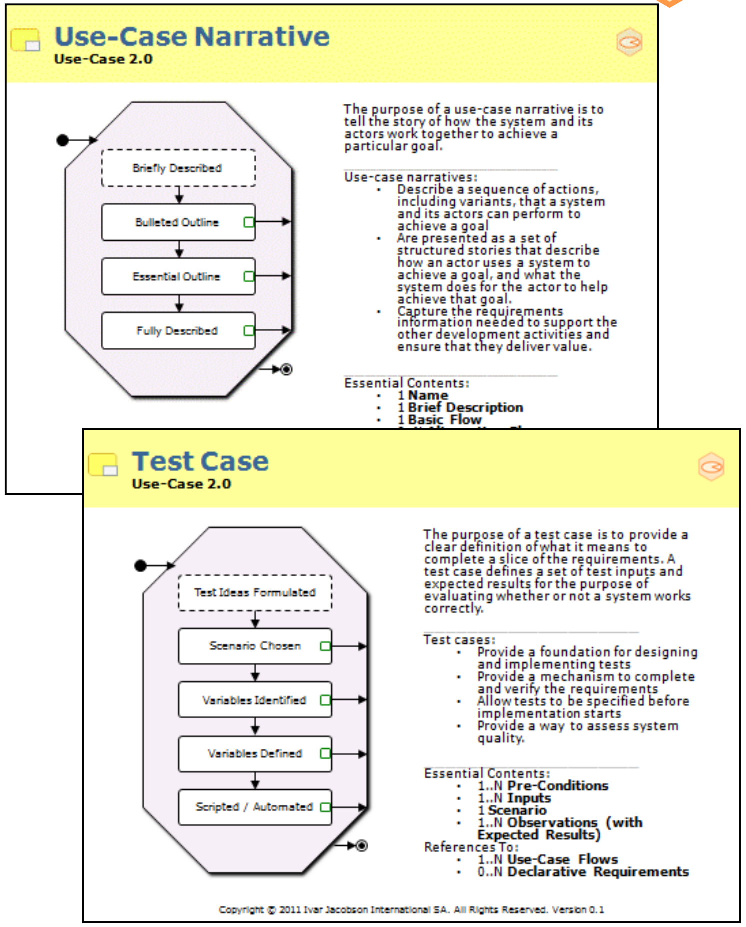
Test Cases verify that the system works as specified by the Requirements

Code implements the requirements

Testing - closing the loop and defining what "done" is.

Start by defining "done"

- Find a meaningful sub-set of the requirements
  - These should tell an end-to-end story and provide value to the users
  - These should be stable and unaffected by the addition of more requirements
  - These should be a small enough set to implement and test within a few days
- Define the test cases
  - These should be scenario based
  - These should be repeatable
  - These should define what's done


Briefly Described
Bulleted Outline
Essential Outline
Fully Described
Test Ideas Formulated
Scenario Chosen
Variables Identified
Variables Defined
Scripted / Automated

Use-Case Narrative
Use-Case 2.0

The purpose of a use-case narrative is to tell the story of how the system and its actors work together to achieve a particular goal.

Use-case narratives:
- Describe a sequence of actions, including variants, that a system and its actors can perform to achieve a goal
- Are presented as a set of structured stories that describe how an actor uses a system to achieve a goal, and what the system does for the actor to help achieve that goal.
- Capture the requirements information needed to support the other development activities and ensure that they deliver value.

Essential Contents:
- 1 **Name**
- 1 **Brief Description**
- 1 **Basic Flow**

Test Case
Use-Case 2.0

The purpose of a test case is to provide a clear definition of what it means to complete a slice of the requirements. A test case defines a set of test inputs and expected results for the purpose of evaluating whether or not a system works correctly.

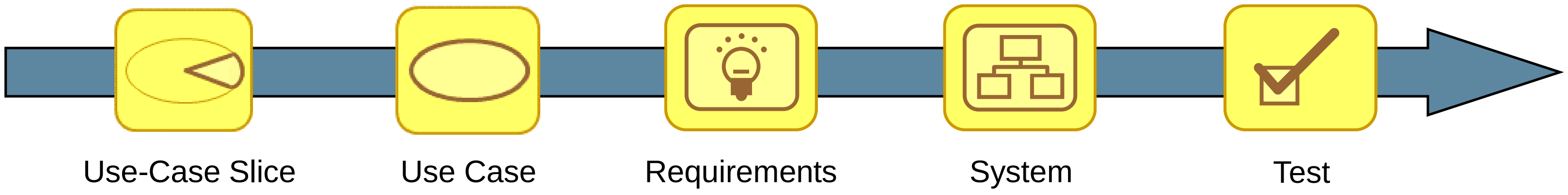
Test cases:
- Provide a foundation for designing and implementing tests
- Provide a mechanism to complete and verify the requirements
- Allow tests to be specified before implementation starts
- Provide a way to assess system quality.

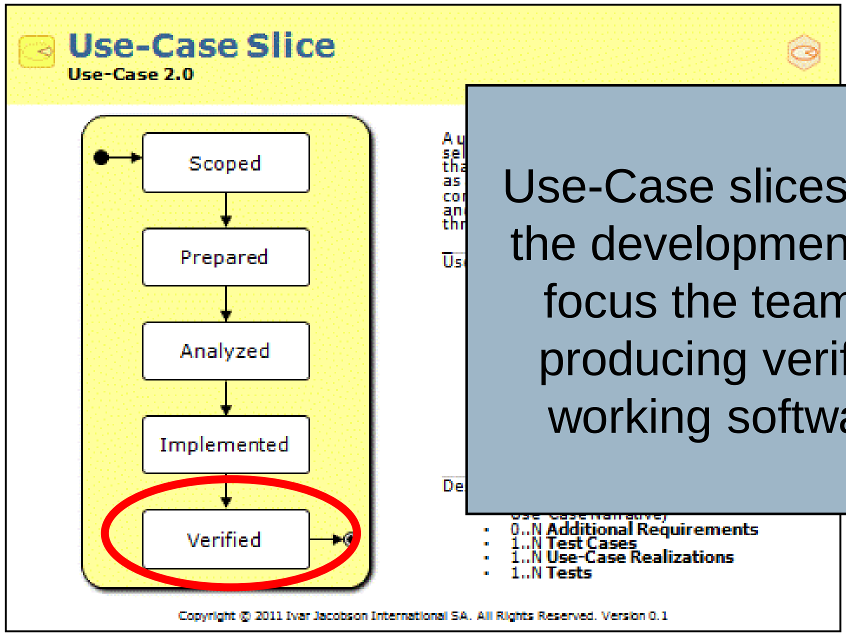
Essential Contents:
- 1..N **Pre-Conditions**
- 1..N **Inputs**
- 1 **Scenario**
- 1..N **Observations (with Expected Results)**

References To:
- 1..N **Use-Case Flows**
- 0..N **Declarative Requirements**

It takes more than just a set of requirements.

Use-Case Driven Development

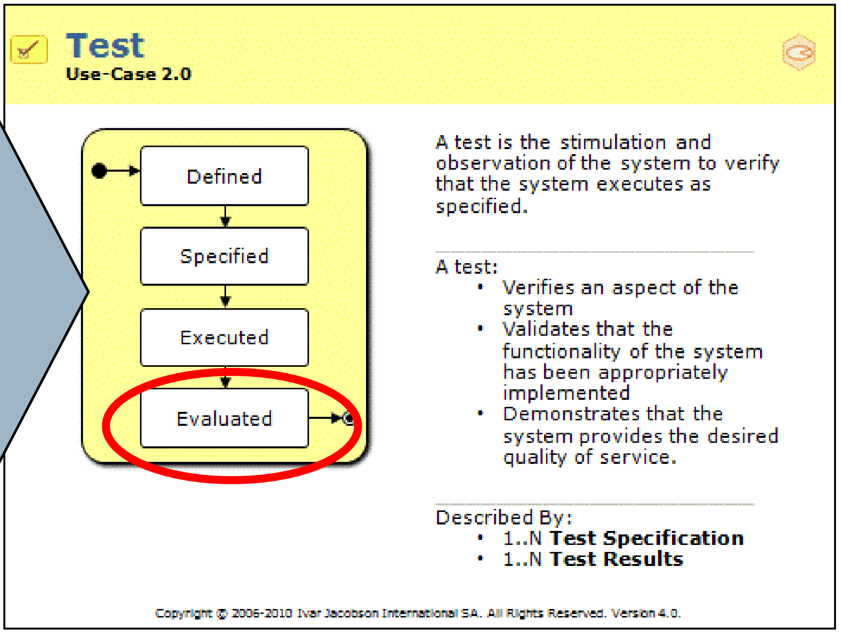



Use-Case Slice
Use-Case 2.0
Scoped
Prepared
Analyzed
Implemented
Verified

0..N Additional Requirements
1..N Test Cases
1..N Use-Case Realizations
1..N Tests

Copyright © 2011 Ivar Jacobson International SA. All Rights Reserved. Version 0.1

Use-Case slices drive the development and focus the team on producing verified, working software.


Test
Use-Case 2.0
Defined
Specified
Executed
Evaluated

A test is the stimulation and observation of the system to verify that the system executes as specified.

A test:
- Verifies an aspect of the system
- Validates that the functionality of the system has been appropriately implemented
- Demonstrates that the system provides the desired quality of service.

Described By:
- 1..N **Test Specification**
- 1..N **Test Results**

Copyright © 2006-2010 Ivar Jacobson International SA. All Rights Reserved. Version 4.0.

Use-Case Slices – They're more than just the stories.

Measuring what's done….

- By implementing and testing the use cases we can measure what's been done
  - Use cases help us find the value and define the right tests
- Bad things to measure:
  - Functions
  - Progress through the disciplines
  - Proportional earned value
  - The number of tasks completed

Good things to measure:

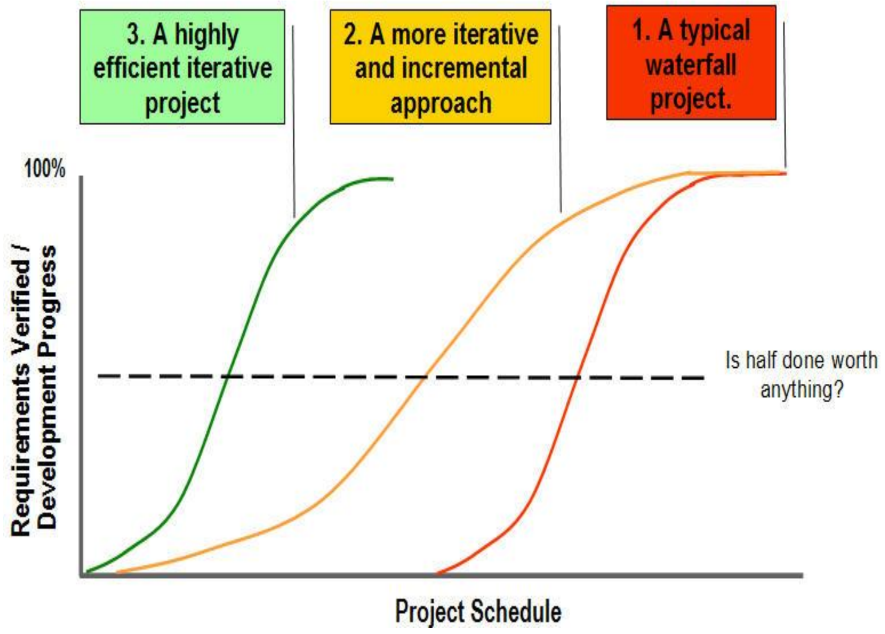
Slices completed

Value to the users

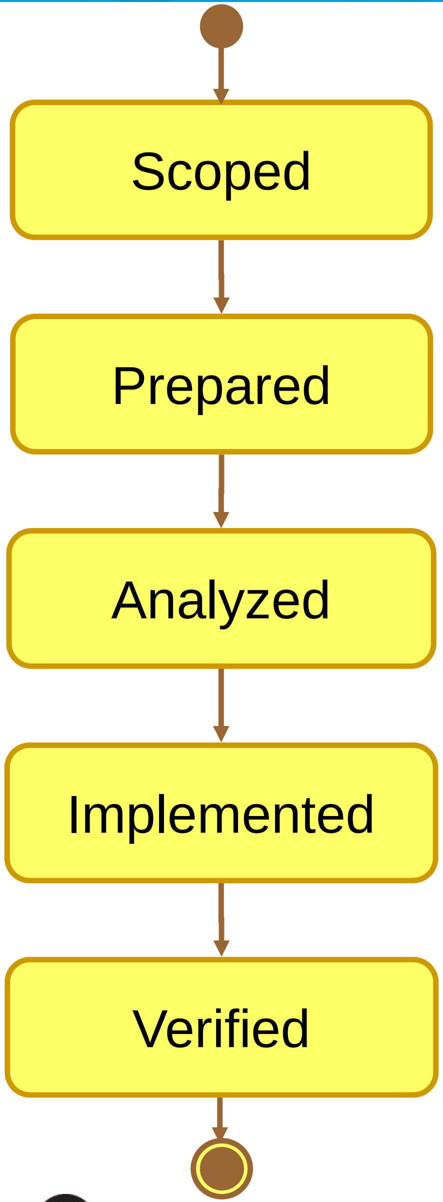
Quality

Tests passed

The only real measure of progress is working software ….and we only know it's working once we've tested it.

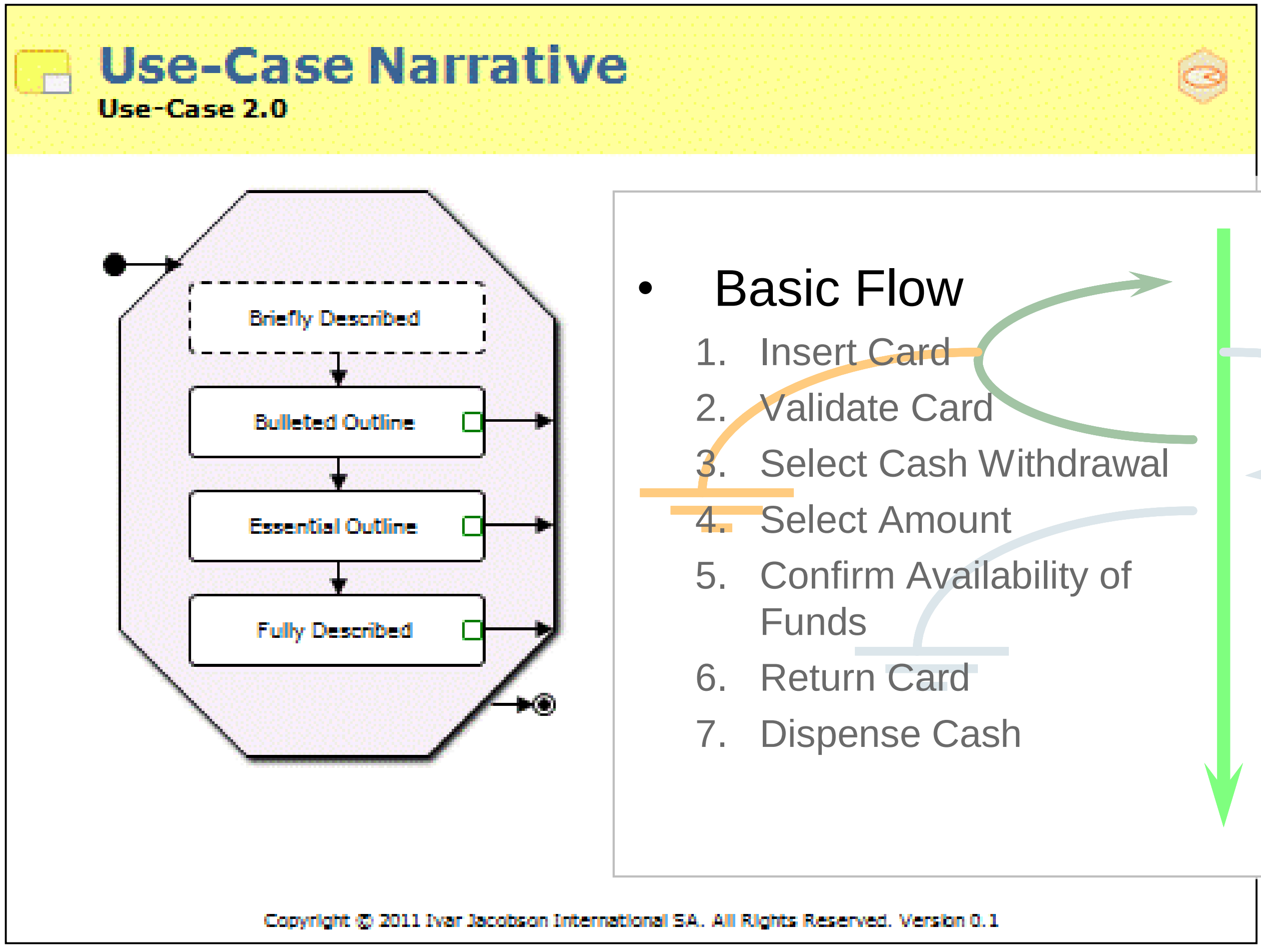


How do we progress and complete the use-cases?



- Scope the use-case slice
  - Identify the flows to be described and implemented
- Prepare the slice
  - Write the use-case narrative
  - Write the test cases
  - Remember to cover special requirements and supporting definitions
- Analyze the slice
  - Allocate the requirements described by the flow of events to the system's implementation elements
  - Understand the impact of implementing the new requirements
- Implement software to deliver the use case
  - Amend the affected implementation elements
  - Integrate the system
- Verify the system produced
  - Execute tests based on the test cases to verify the system delivers the use cases as specified.

Use-case narratives contain many requirements…



Use-Case Narrative

Use-Case 2.0

Copyright © 2011 Ivar Jacobson International SA. All Rights Reserved. Version 0.1

- Basic Flow
  1. Insert Card
  2. Validate Card
  3. Select Cash Withdrawal
  4. Select Amount
  5. Confirm Availability of Funds
  6. Return Card
  7. Dispense Cash

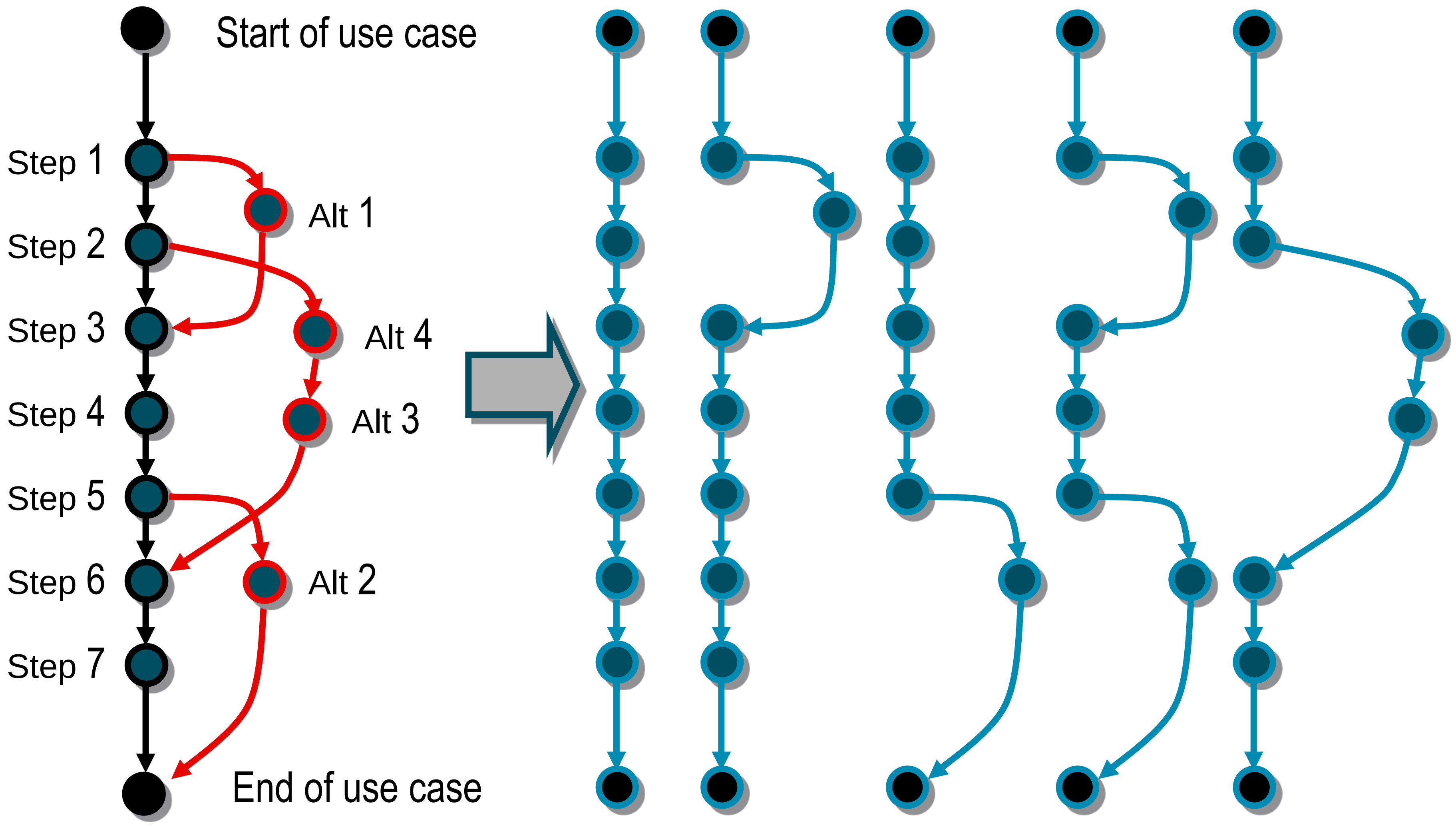
- Alternative Flows
  - A1 Invalid Card
  - A2 Non-Standard Amount
  - A3 Receipt Required
  - A4 Insufficient Funds in ATM
  - A5 Insufficient Funds in Acct
  - A6 Would Cause Overdraft
  - A7 Card Stuck
  - A8 Cash Left Behind
  - Etc…

… often too many to code and test in one go.

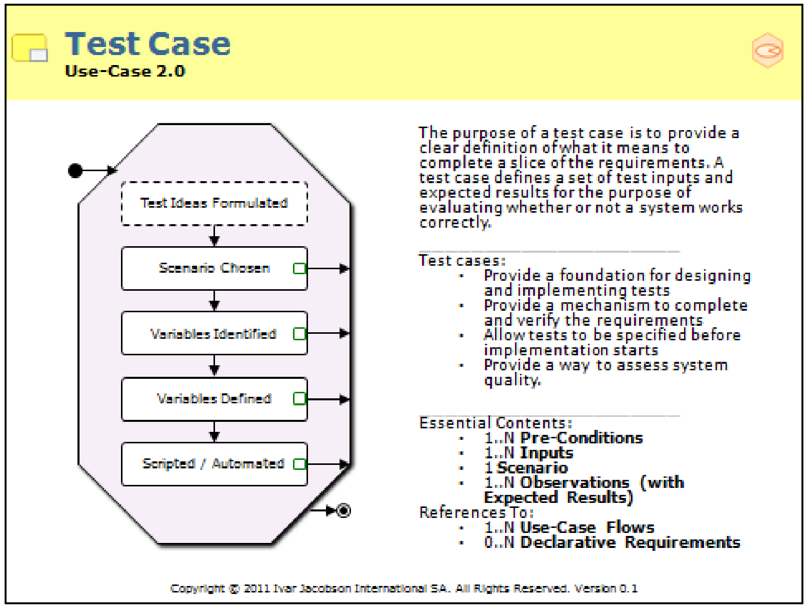
Use-case narratives tell many stories

1 Use Case

Many stories…



Use Case based Test Cases



Test Case
Use-Case 2.0

Test Ideas Formulated
Scenario Chosen
Variables Identified
Variables Defined
Scripted / Automated

The purpose of a test case is to provide a clear definition of what it means to complete a slice of the requirements. A test case defines a set of test inputs and expected results for the purpose of evaluating whether or not a system works correctly.

Test cases:

- Provide a foundation for designing and implementing tests
- Provide a mechanism to complete and verify the requirements
- Allow tests to be specified before implementation starts
- Provide a way to assess system quality.

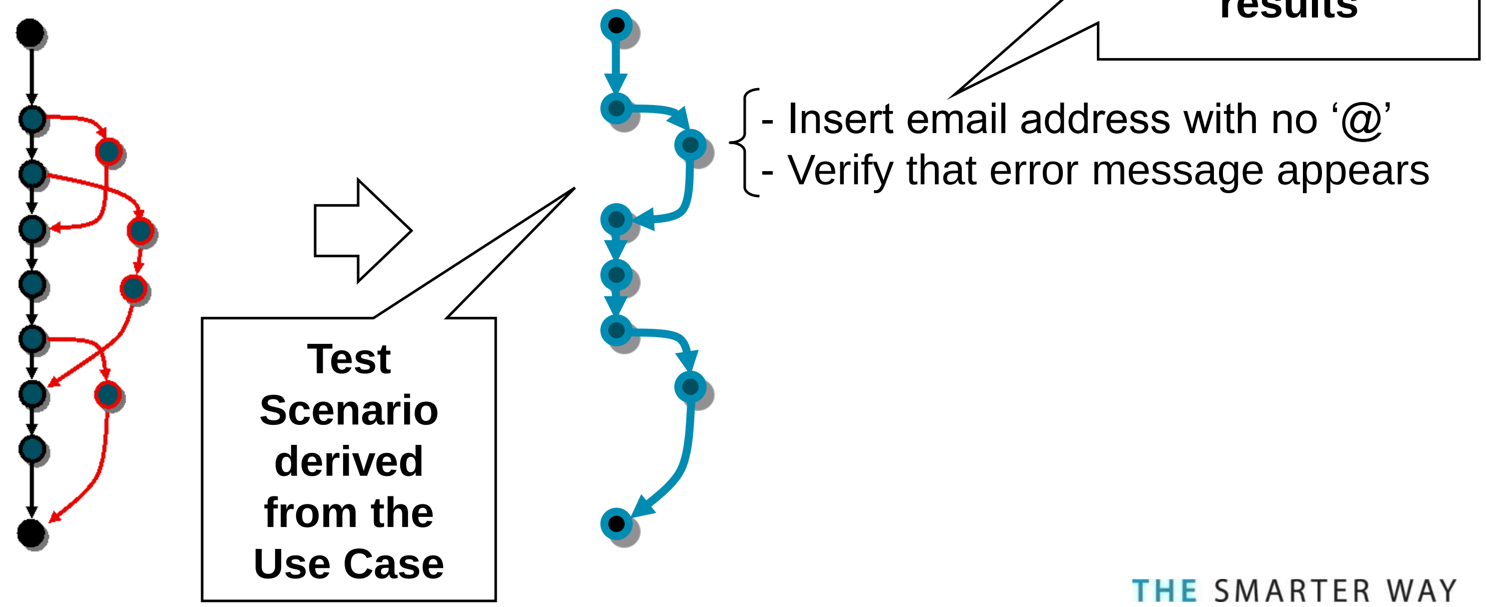
Essential Contents:

- 1..N **Pre-Conditions**
- 1..N **Inputs**
- 1 **Scenario**
- 1..N **Observations (with Expected Results)**

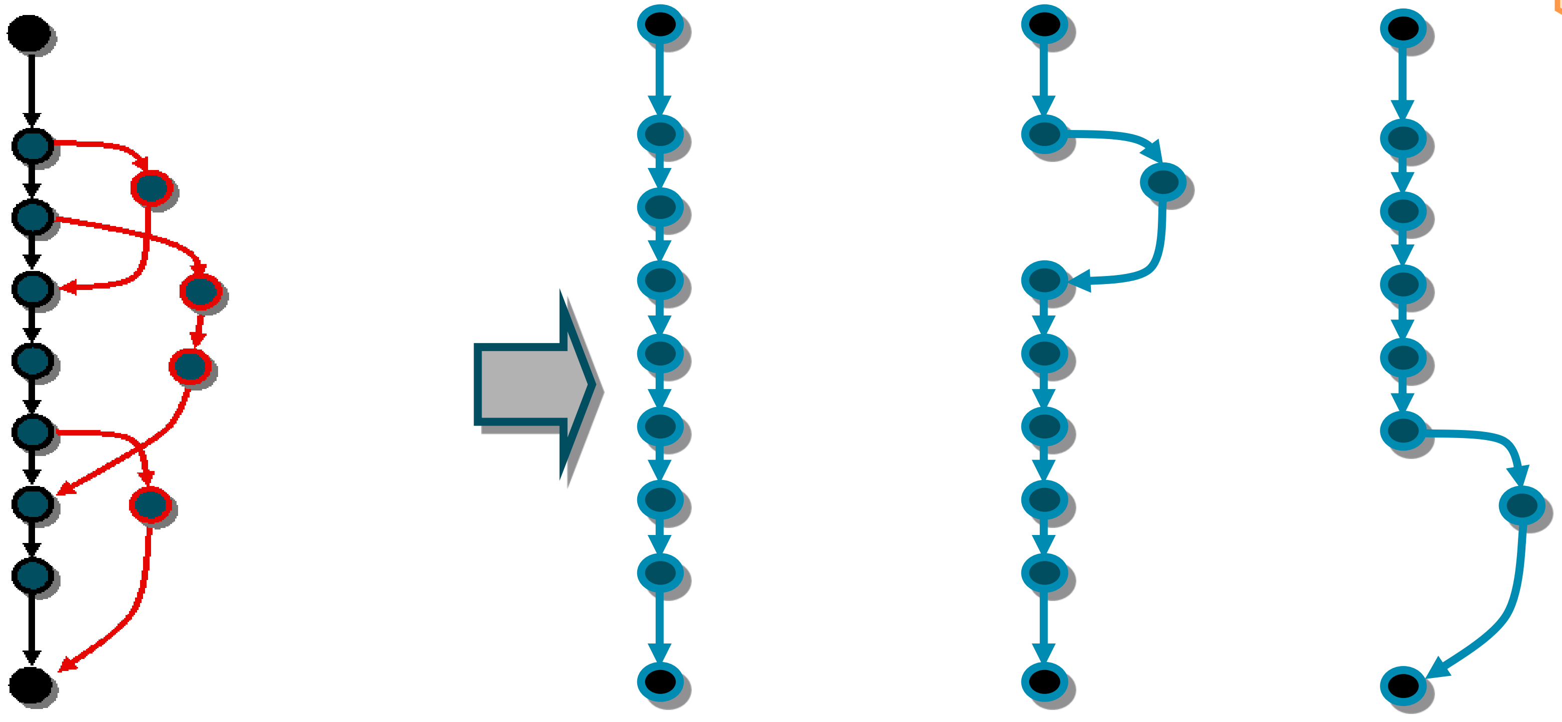
References To:

- 1..N **Use-Case Flows**
- 0..N **Declarative Requirements**

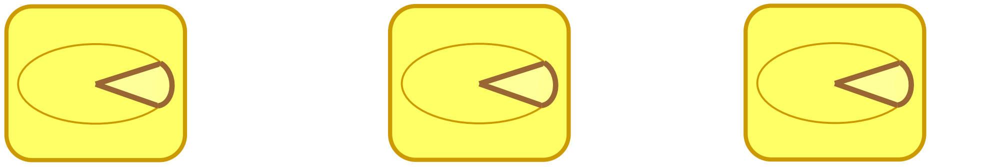
- Set of **inputs** and **expected results** for the purpose of evaluating whether or not a system correctly implements a specific **test scenario**
- Allow tests to be specified before implementation starts


Inputs and expected results
- Insert email address with no '@'
- Verify that error message appears
Test Scenario derived from the Use Case

Managing Scope Using Slices



Use-Case 1



Use-Case Slice 1.1

Priority 1

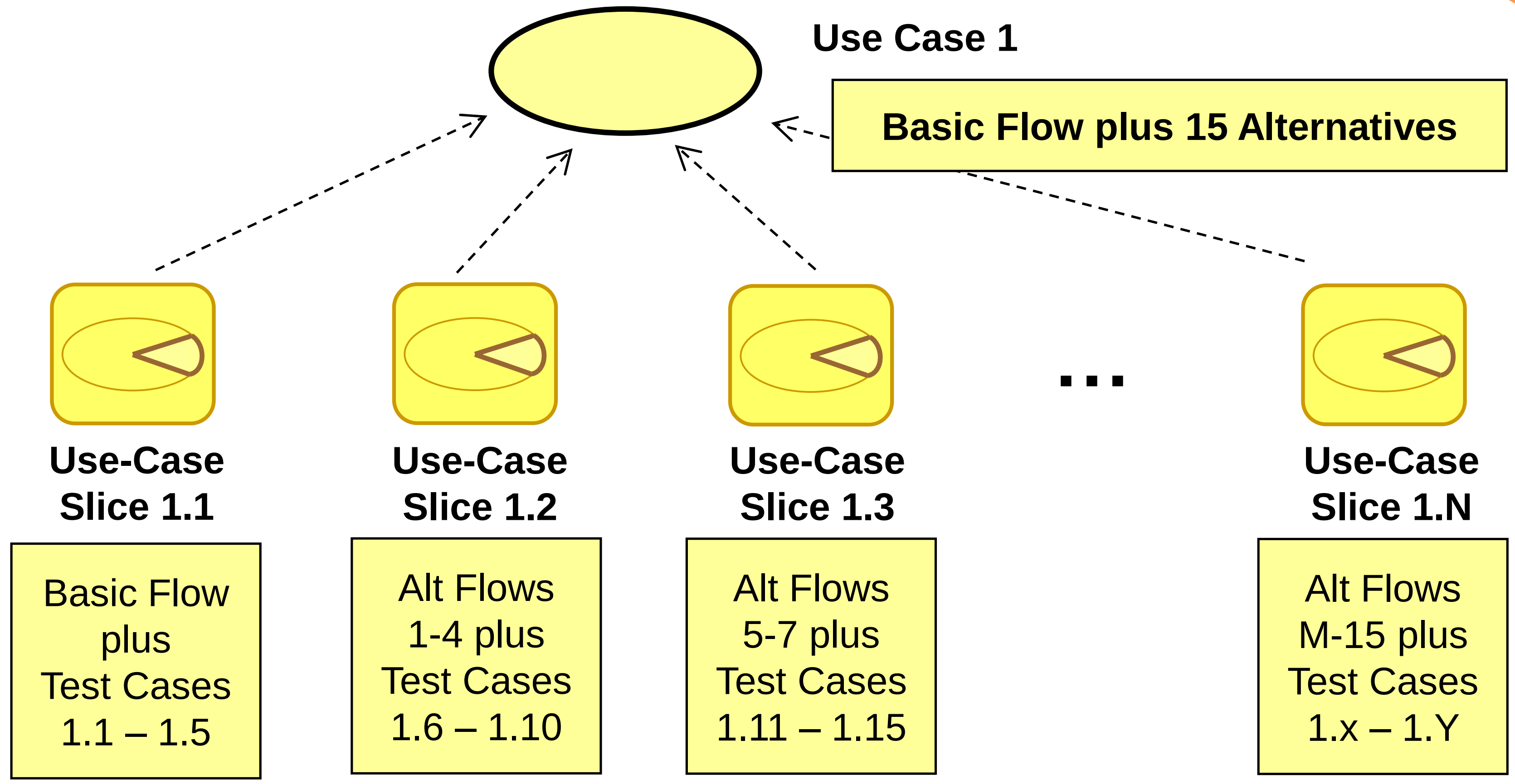
Use-Case Slice 1.2

Delayed until next release

Use-Case Slice 1.3

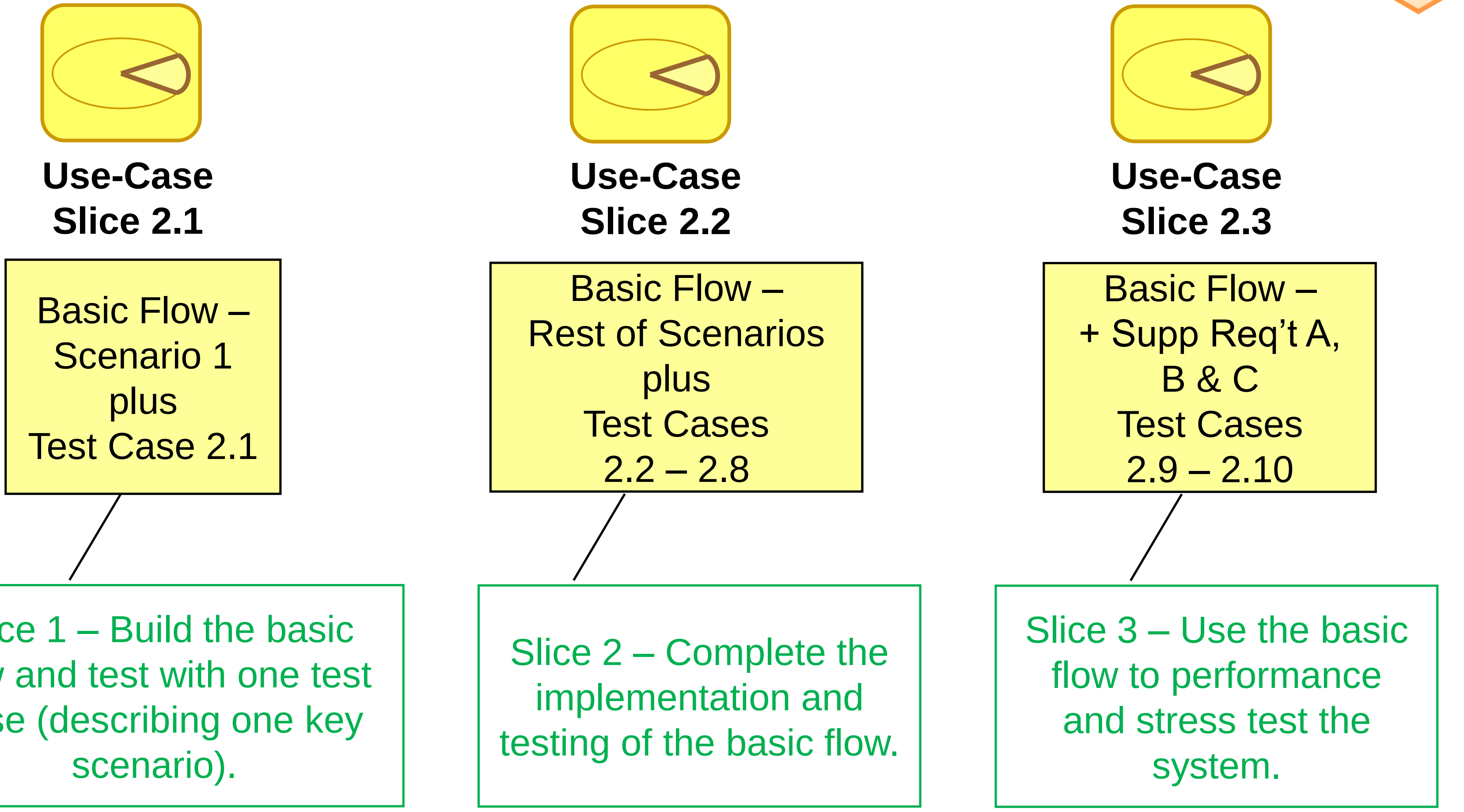
Priority 2

Use Cases and Use-Case Slices


Use Case 1
Basic Flow plus 15 Alternatives
Use-Case Slice 1.1
Basic Flow plus Test Cases 1.1 – 1.5
Use-Case Slice 1.2
Alt Flows 1-4 plus Test Cases 1.6 – 1.10
Use-Case Slice 1.3
Alt Flows 5-7 plus Test Cases 1.11 – 1.15
…
Use-Case Slice 1.N
Alt Flows M-15 plus Test Cases 1.x – 1.Y

The Use-Case Slices split the use case up into a number of smaller, separately deliverable parts

Handling non-functional and other requirements

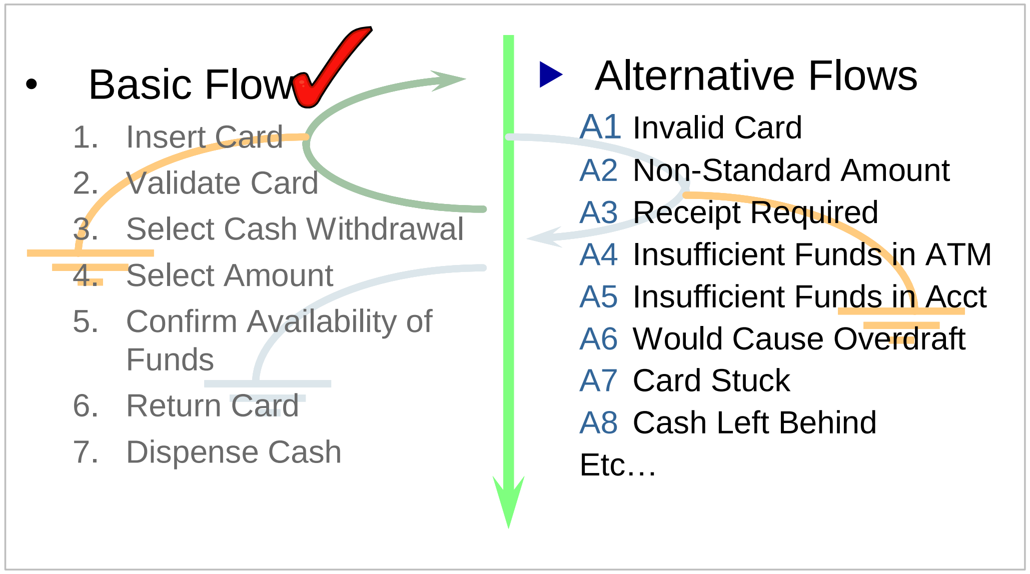

Use-Case Slice 2.1
Use-Case Slice 2.2
Use-Case Slice 2.3
Basic Flow – Scenario 1 plus Test Case 2.1
Basic Flow – Rest of Scenarios plus Test Cases 2.2 – 2.8
Basic Flow – + Supp Req't A, B & C Test Cases 2.9 – 2.10
Slice 1 – Build the basic flow and test with one test case (describing one key scenario).
Slice 2 – Complete the implementation and testing of the basic flow.
Slice 3 – Use the basic flow to performance and stress test the system.

The slices can include test cases to address the non-functional as well as the functional requirements.

Finding Use-Case Slices

- **Think about your risks and identify the most important stories**
- Think about the natural groups of flows
- Think about testing and proving the system
- Think work items and driving the development

Risks			
4	It might be harder than we think (estimates)	Very High	Build Withdraw Cash
5	Reliability of the O/S platform	Very High	Build Withdraw Cash
6	Scalability of J2EE Infrastructure	Very High	Build Withdraw Cash



- Basic Flow ✓
  1. Insert Card
  2. Validate Card
  3. Select Cash Withdrawal
  4. Select Amount
  5. Confirm Availability of Funds
  6. Return Card
  7. Dispense Cash

- Alternative Flows
  - A1 Invalid Card
  - A2 Non-Standard Amount
  - A3 Receipt Required
  - A4 Insufficient Funds in ATM
  - A5 Insufficient Funds in Acct
  - A6 Would Cause Overdraft
  - A7 Card Stuck
  - A8 Cash Left Behind
  - Etc…

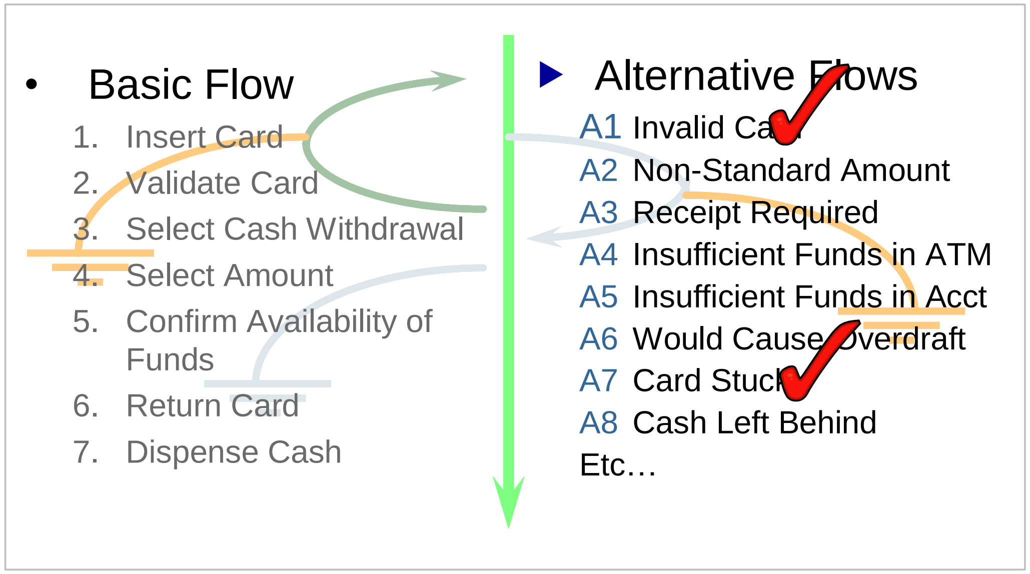
Build a simple cash withdrawal based on the Basic Flow

UCS 1.1 - Build a simple cash withdrawal based on the basic Flow

- One test case one account / one amount.

Finding Use-Case Slices

- Think about your risks and identify the most important stories
- **Think about the natural groups of flows**
- Think about testing and proving the system
- Think work items and driving the development



- Basic Flow
  1. Insert Card
  2. Validate Card
  3. Select Cash Withdrawal
  4. Select Amount
  5. Confirm Availability of Funds
  6. Return Card
  7. Dispense Cash

- Alternative Flows
  - A1 Invalid Card ✓
  - A2 Non-Standard Amount
  - A3 Receipt Required
  - A4 Insufficient Funds in ATM
  - A5 Insufficient Funds in Acct
  - A6 Would Cause Overdraft
  - A7 Card Stuck ✓
  - A8 Cash Left Behind
  - Etc…

There are a number of flows about card handling?

Wouldn't you implement them all at the same time?

UCS 1.2 – Card handling during cash withdrawal

A 1.1 Handle Invalid Card

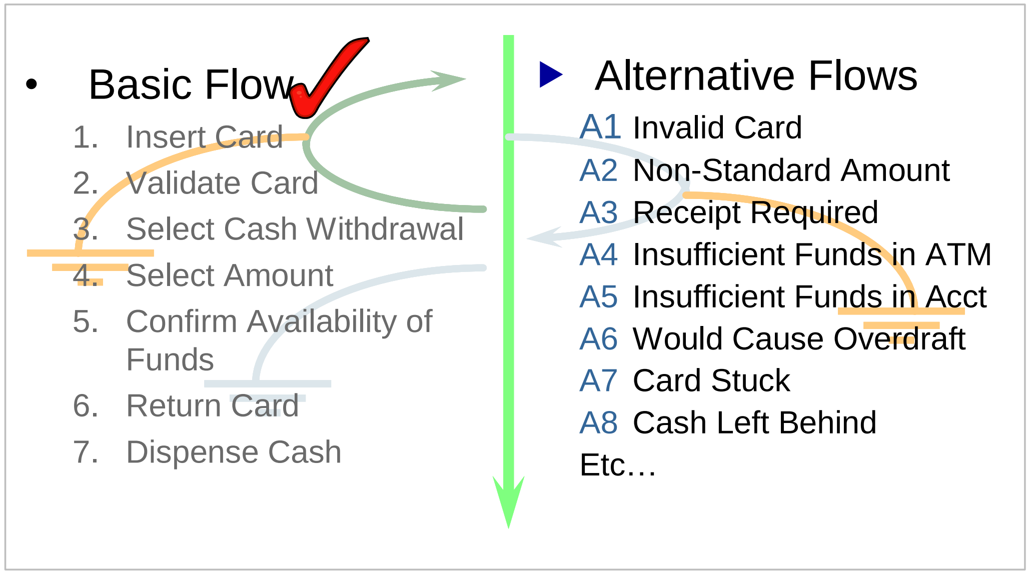
A 1.2 Handle Unreadable Card

A 1.3 Handle Card Jam

Numerous test cases

Finding Use-Case Slices

- Think about your risks and identify the most important stories
- Think about the natural groups of flows
- **Think about testing and proving the system**
- Think work items and driving the development



How can we address the supplementary requirements?

How will we know when we're done?

Performance 1.1: Peak Loading

Performance 1.2: Transaction Service Levels

UCS 1.3 – Peak Load Testing

Basic Flow

P 1.1 Peak Loading

P 1.2 Service Levels

Numerous orchestrated test cases.

Finding Use-Case Slices

- Think about your risks and identify the most important stories
- Think about the natural groups of flows
- Think about testing and proving the system
- **Think work items and driving the development**

What are we going to do in the next iteration – 2 weeks.

Well we can't do the whole use case?

UCS 1.A – Handle Security Breaches

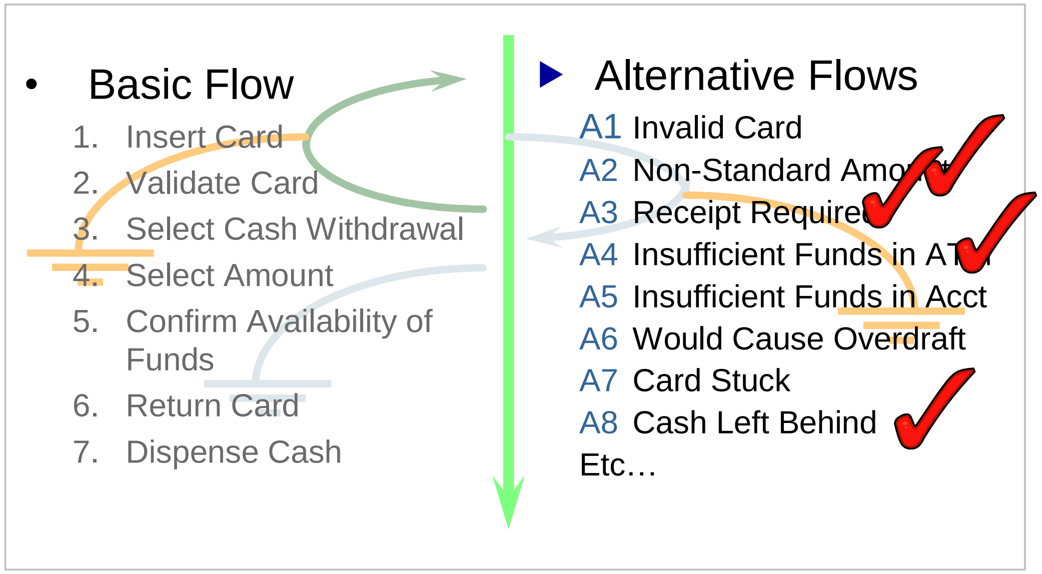
UCS 1.B – Handle Loss of Critical Resources

UCS 1.C – Forgetful Customer

UCS 1.D – Non-Standard Amounts

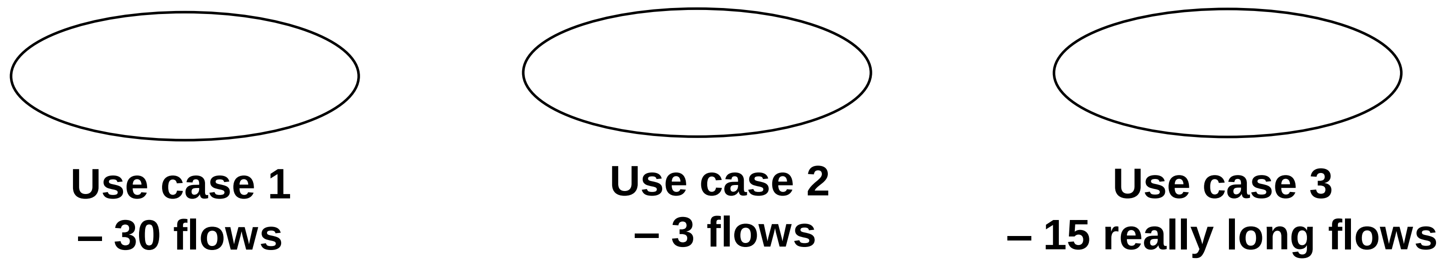
UCS 1.E – Receipt Handling

…..



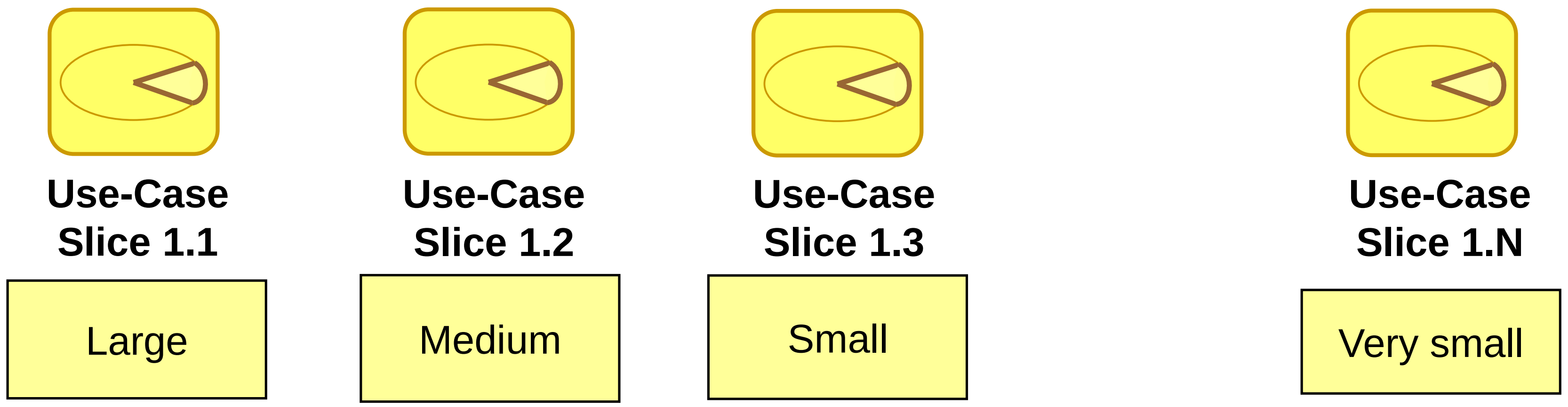
Sizing the work to be done

- Use cases can be any size



  – And are often to big to describe, size, estimate or deliver in one go

- Use-case slices can be split up or combined to create sensibly sized work items



Prioritizing and Ordering The Work To Be Done

Use-Case Slice 1.1

Priority 1

Use-Case Slice 2.1

Priority 1

Use-Case Slice 1.3

Priority 2

Use-Case Slice 3.3

Priority 2

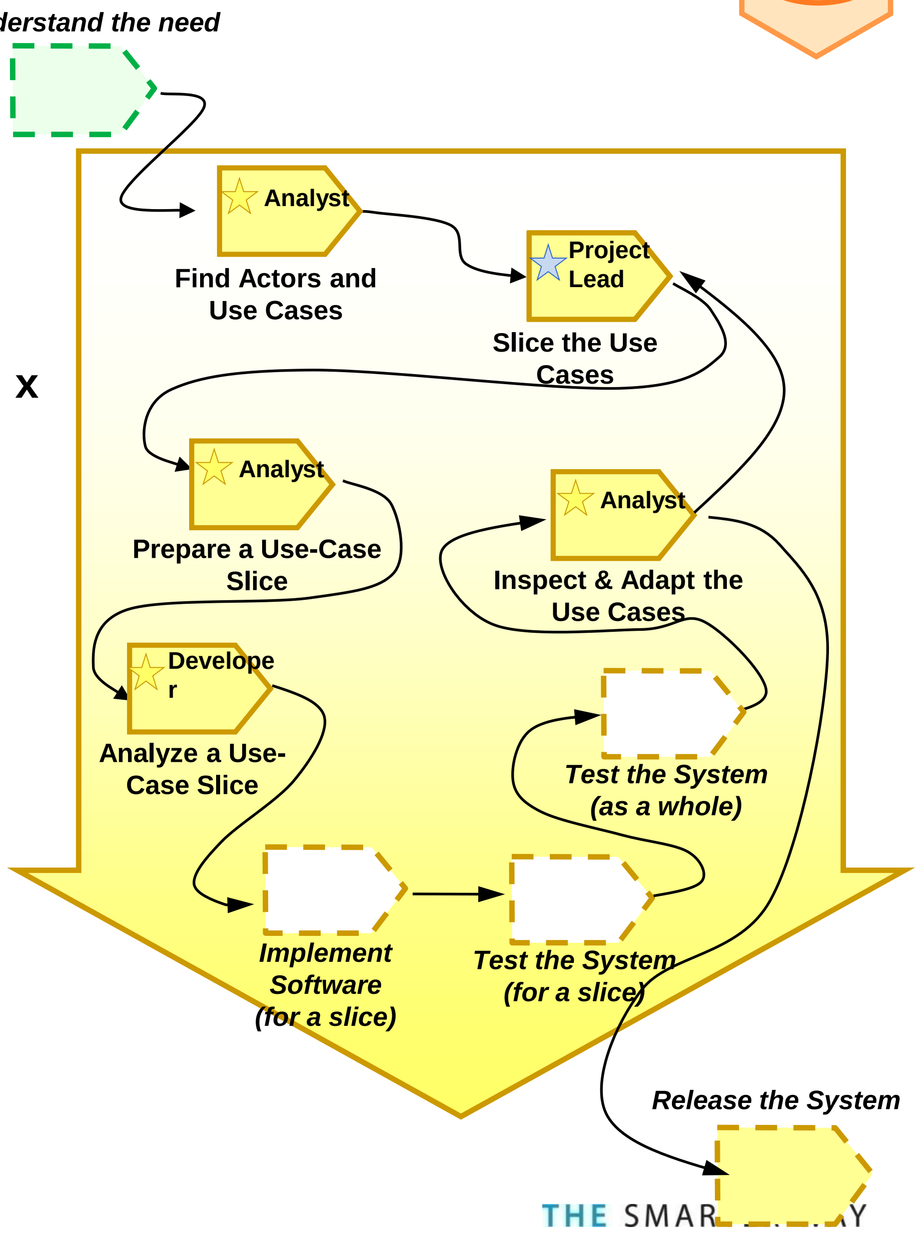
For Iteration x

Use-Case Slice 1.2

Delayed until next release

Use-Case Slice 3.2

Delayed until next release



Exercise 3.2: Creating Use-Case Slices

- Based on your outlines prioritize your flows of events and chunk up them up into a set of use-case slices
  - Estimate the size of each use-case slice using the scale very small, small, medium, large
    - What is in your largest use-case slice?
  - See if you can estimate the complexity of the implementation required using the scale trivial, easy, hard, very hard
    - What is you most complex use-case slice?
  - See if you can prioritize your use-case slices
    - Classify your slices using the MoSCoW rules
    - Place the slices you identify into a forced ranking

Estimating What You Can Do

Size \ Complexity	Trivial	Easy	Hard	Very Hard
Large	2	4	6	9
Medium	1	2	4	6
Small	0.5	1	2	4
Very Small	0.25	0.5	1	2

Will a system with a complex design, have a complex use-case model?

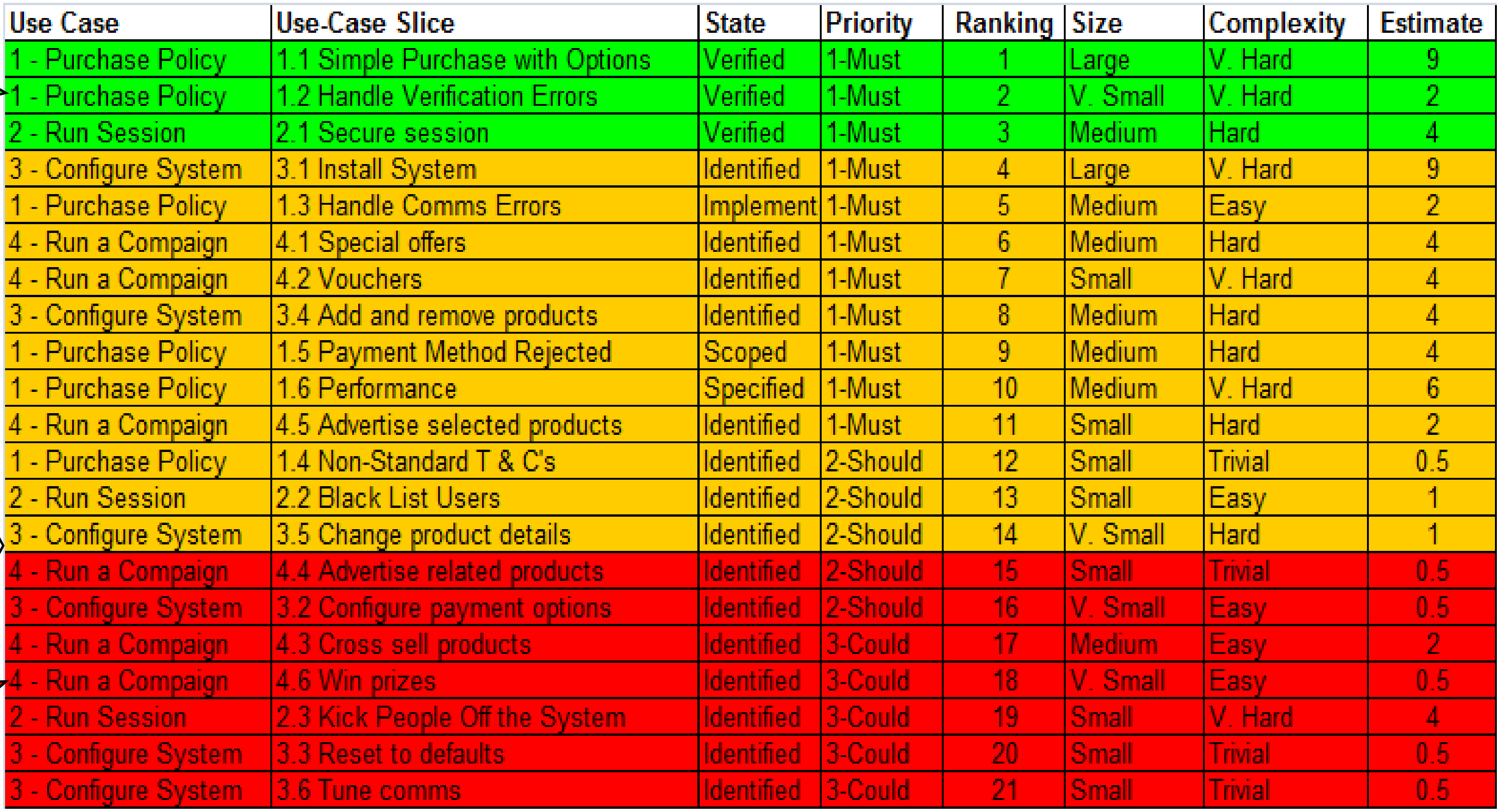
Tracking Done

Done

Doable

Where time runs out.

At risk



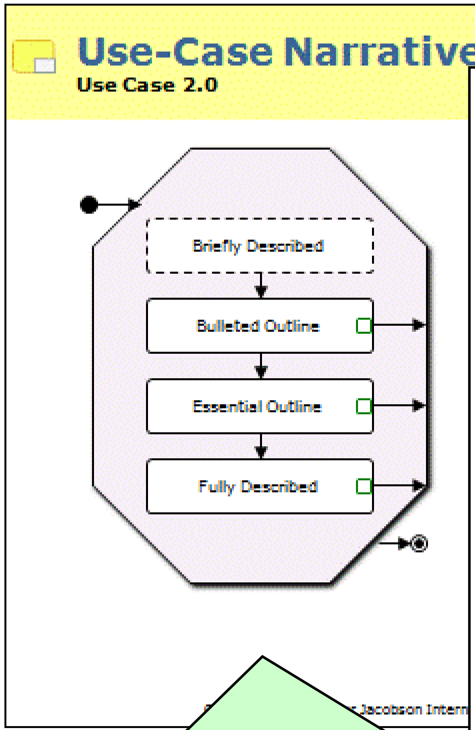
Use Case	Use-Case Slice	State	Priority	Ranking	Size	Complexity	Estimate
1 - Purchase Policy	1.1 Simple Purchase with Options	Verified	1-Must	1	Large	V. Hard	9
1 - Purchase Policy	1.2 Handle Verification Errors	Verified	1-Must	2	V. Small	V. Hard	2
2 - Run Session	2.1 Secure session	Verified	1-Must	3	Medium	Hard	4
3 - Configure System	3.1 Install System	Identified	1-Must	4	Large	V. Hard	9
1 - Purchase Policy	1.3 Handle Comms Errors	Implement	1-Must	5	Medium	Easy	2
4 - Run a Compaign	4.1 Special offers	Identified	1-Must	6	Medium	Hard	4
4 - Run a Compaign	4.2 Vouchers	Identified	1-Must	7	Small	V. Hard	4
3 - Configure System	3.4 Add and remove products	Identified	1-Must	8	Medium	Hard	4
1 - Purchase Policy	1.5 Payment Method Rejected	Scoped	1-Must	9	Medium	Hard	4
1 - Purchase Policy	1.6 Performance	Specified	1-Must	10	Medium	V. Hard	6
4 - Run a Compaign	4.5 Advertise selected products	Identified	1-Must	11	Small	Hard	2
1 - Purchase Policy	1.4 Non-Standard T & C's	Identified	2-Should	12	Small	Trivial	0.5
2 - Run Session	2.2 Black List Users	Identified	2-Should	13	Small	Easy	1
3 - Configure System	3.5 Change product details	Identified	2-Should	14	V. Small	Hard	1
4 - Run a Compaign	4.4 Advertise related products	Identified	2-Should	15	Small	Trivial	0.5
3 - Configure System	3.2 Configure payment options	Identified	2-Should	16	V. Small	Easy	0.5
4 - Run a Compaign	4.3 Cross sell products	Identified	3-Could	17	Medium	Easy	2
4 - Run a Compaign	4.6 Win prizes	Identified	3-Could	18	V. Small	Easy	0.5
2 - Run Session	2.3 Kick People Off the System	Identified	3-Could	19	Small	V. Hard	4
3 - Configure System	3.3 Reset to defaults	Identified	3-Could	20	Small	Trivial	0.5
3 - Configure System	3.6 Tune comms	Identified	3-Could	21	Small	Trivial	0.5

…and knowing how much more you can do.

IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

You only need to outline the flows to create the slices



Use-Case 1: Withdraw Cash

- Basic Flow
  1. Insert Card
  2. Validate Card
  3. Select Cash Withdrawal
  4. Select Amount
  5. Confirm Availability of Funds
  6. Return Card
  7. Dispense Cash

1.3 Card Handling Errors

Story: Wrong type of card.
Flows: BF, A1
Test Conditions: Invalid card

Story: C

Test Cond
Acc

Points:

1.2 Full Cash Withdrawal

Story: Empty an Account
Flows: BF
Test Conditions: Card 001, Account 001 containing £200
Story: Withdraw Daily Limit

ccount 002

1.1 Normal Cash Withdrawal

Story: Withdraw Beer Money
Flows: BF
Test Conditions: Default Card, Default Account, Default Amounts

Points:

Use case narratives can be described at different levels of detail.
They start very lightweight by outlining the flows.

This is enough detail to identify our stories and define our slices.

IVAR JACOBSON INTERNATIONAL

THE SMARTER WAY

Summary

- The use-case slices allow us to drive the creation of working software based upon the flows of events defined by the use cases
- Use Cases have many forms and states and evolve during the software development lifecycle
- Use Cases underpin all areas of the software development lifecycle ("use-case driven"), not just requirements
- The authoring of use cases to increasing levels of completeness addresses differing areas of project risk