



System Functionality (Functional Requirements)

Architectural Thinking for Intelligent Systems

Winter 2020/2021

Prof. Dr. habil. Jana Koehler

Agenda

- Functional requirements from an architectural perspective
 - The importance of negotiating requirements
- User stories vs. use cases
- Defining goal hierarchies
- Measurable acceptance criteria
- Tipps for writing good user stories & use cases

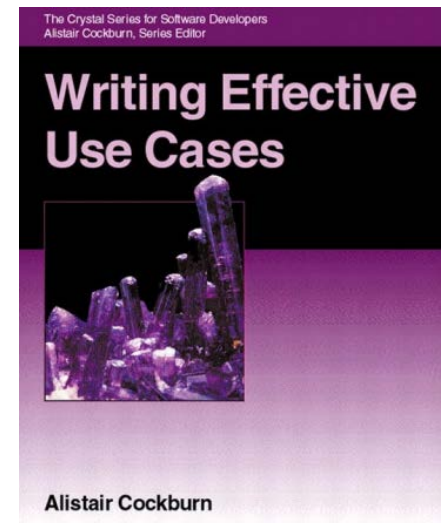
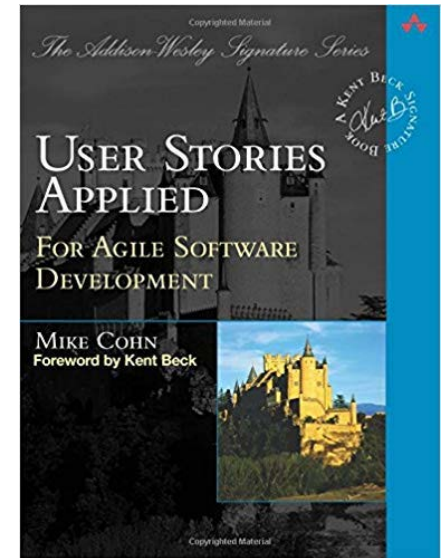
Tutorial Assignment 4:

- We dive deeper into the functional requirements our system has to meet and formulate user stories and use cases.
- We analyze functional dependencies by creating a goal hierarchy.
- Remember to make your requirements SMART
 - Write them as **specific** as possible
 - Make them **measurable** by formulating quantifiable properties
 - Think about **realization** and **attainability**
 - Think about how they will be **traceable** throughout the development process

Recommended Reading

- Cohn: User Stories Applied
 - Chapter 1 (pages 3-15)
 - Chapter 2 (pages 17-28)
 - Chapter 6 (pages 67-73)

- Cockburn: Writing Effective Use Cases
 - Chapter 1.1-1.3 (pages 1-12)
 - Chapter 2 (pages 23-33)
 - Chapter 5.1-5.5 (pages 61-69)



User Stories Revisited



<https://agileinpills.wordpress.com/2013/07/18/learning-by-teaching-user-stories/>

What is a User Story? In comparison to a Use Case?

It's often best to think of the written part as a pointer to the real requirement.

<http://www.mountangoatsoftware.com/agile/user-stories>

**My one liner is that a story is a promise to have a conversation and a use case is the record of the conversation. If you think you need one.
Jim Standley**

**A Use Case is a way of describing requirements.
A User Story is a way of prioritizing work. Tim Wright**



User Story is simply, a user's story. It is business people's version of describing the world, their way of "starting an idea" basically starting a conversation (requirements elicitation) of whether their idea (to get some business benefit) is feasible.

<http://alistair.cockburn.us/A+user+story+is+to+a+use+case+as+a+gazelle+is+to+a+gazebo>

Good User Stories

“As a [role] I can [function] so that [benefit].”

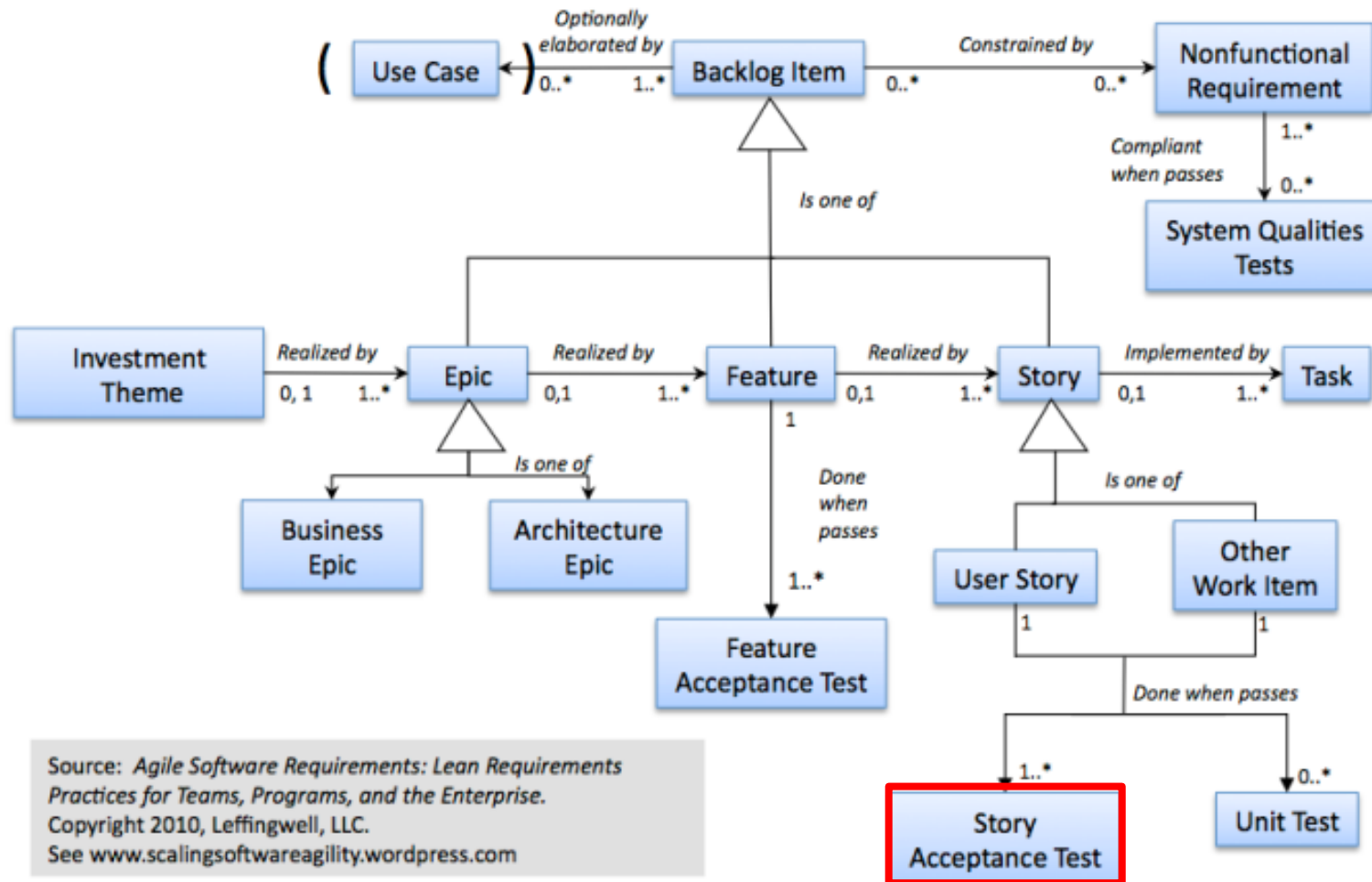
As a student, I can find my grades online so that I don't have to wait until I am notified by the professor to find out whether I passed the exam.

Bill Wake's INVEST acronym

Independent	We want to be able to develop in any sequence.
Negotiable	Avoid too much detail; keep them flexible so the team can adjust how much of the story to implement.
Valuable	Users or customers get some value from the story.
Estimatable	The team must be able to use them for planning.
Small	Large stories are harder to estimate and plan. By the time of iteration planning, the story should be able to be designed, coded, and tested within the iteration.
Testable	Document acceptance criteria, or the definition of done for the story, which lead to test cases.

<https://help.rallydev.com/writing-great-user-story>

Agile Enterprise Backlog Model



Epic vs. User Story

As a bank, we need mechanisms to authenticate our customers across a variety of interaction channels, e.g. in the bank branch, at the ATM, on the phone...

Because an epic is generally too large for an agile team to complete in one iteration, it is split into multiple smaller user stories before it is worked on.

The epic above could be split into several stories, including these two:

As a bank, we can offer an authentication mechanism on our ATMs where customers use a card and a pin.

As a bank customer, I can change the PIN linked with my card such that I can freely choose a PIN that I can better remember.

Definition in an Incremental 3-Step Process

- Brief description of the need
- Conversations to solidify the details (happening during backlog grooming and iteration planning)
- Tests that confirm the story's satisfactory completion
- Acceptance criteria = Definition of Done for the story
 1. Product owner should list as many as possible to clarify the intent of the story
 2. Team should have a conversation about them and adjust the acceptance criteria - capture the critical details in acceptance criteria
 3. Once an iteration has begun, testers can formalize acceptance criteria into acceptance tests

<https://help.rallydev.com/writing-great-user-story>

Frequent Mistakes when defining Acceptance Criteria

- **One-sided and incomplete exploration**
 - Acceptance is only looked at from one angle
 - If possible, consider all facets of acceptance
 - System quality for the user: efficient, simple, intuitive, informative, flexible

- **Discuss, discuss, discuss ...**
 - What are the really critical features for the user?
 - How do we make them measurable?

- **The acceptance test forgets the user**
 - Too much focus on technical system details
 - Customer needs are not taken seriously
 - Measurability remains vague

How do you assess quality of description in these user stories?

- 1) “I want the brochure to be colorful.”
- 2) “As product owner, I want a list of highly-rated restaurants on the brochure.”
- 3) “As a bank customer, I can change the PIN linked with my card such that I can freely choose a PIN that I can better remember.”

I	Independent
N	Negotiable
V	Valuable
E	Estimatable
S	Scalable (small sized)
T	Testable

“As Product Owner, I want a list of highly-rated restaurants on the brochure.”

- User perspective or SCRUM development role?
- Better: “As a gourmet tourist, I want a list of highly-rated restaurants on the brochure.”
- Even Better: “As a gourmet tourist, I want a list of highly-rated restaurants on the brochure with a characterization of the food type they offer and a summary of their rating such that I can easily find a restaurant that fits my needs.”

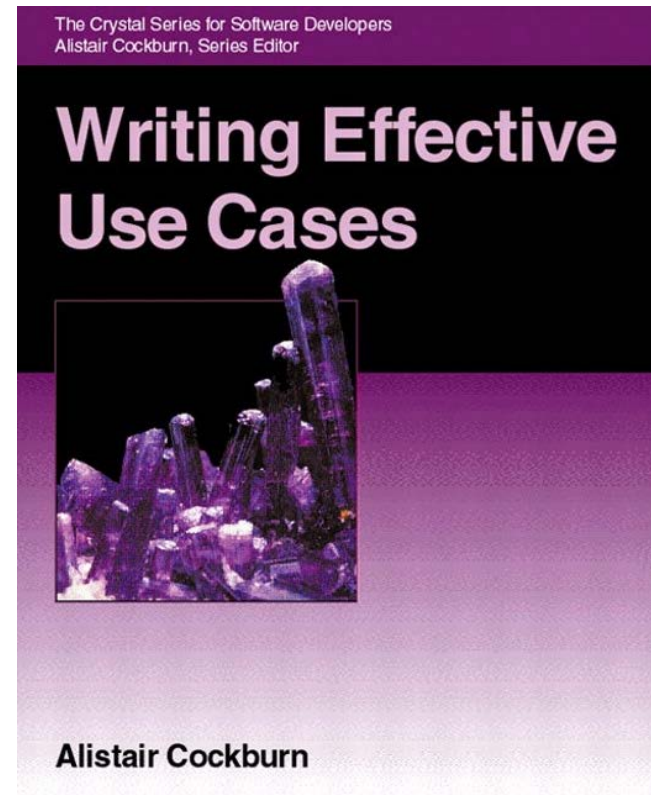
Exercise: Let us define some Acceptance Criteria

- **“As a bank customer, I can change the PIN linked with my card such that I can freely choose a PIN that I can better remember.”**
 - Acceptance Criteria:

- **As a student, I can find my grades online so that I don't have to wait until I receive a notification to know whether I passed.**
 - Acceptance Criteria:

<http://blogs.collab.net/agile/user-story-examples-and-counterexamples>

Writing Effective Use Cases



Use Cases Reduce our Decision Space

- How will a system work precisely in order to realize the agreed user story?
 - *“As a bank customer, I can change the PIN linked with my card such that I can freely choose a PIN that I can better remember.”*

- How will the function be implemented in the system?
 - First insert card?
 - Or choose a change option from main menu first?
 - What can go wrong during the change process?
 - What conditions hold before or after the change process?

Attribute	Description
Short description / goal of the Use Case	Descriptive text of just a few sentences outlining the main purpose of the use case.
Actor(s)	See section 2.3 for possible actors
Preconditions	Conditions that must be fulfilled in order to perform the use case. Example: { The user is authorized } AND { Client Profile is available }
Basic flow	Describes the main control flow of the use case. Example: 1. <i>The user selects the ice cream's flavour from one of {chocolate, vanilla, strawberry}.</i> 2. <i>The system indicates the price of one portion of the desired ice cream.</i> 3. <i>The user inserts coins up to the amount indicated by the system.</i> 4. ...
Alternative flow 1	Describes the first alternative flow. Example: Steps 1 through 2 as in the basic flow. 3. <i>The user cancels the action.</i>
Alternative flow n	Describes the n th alternative flow.
Postconditions	Assertions that hold true after the use case has been performed.
Use Case-specific non-functional requirements	Example: <i>This use case must support up to 20 users concurrently.</i>
Special considerations	E. g. restrictions on the data to be entered.
Creation date	Format: DD.MM.YYYY
Modification	Version history; format: DD.MM.YYYY Description of the change(s)

A possible Use Case Template

Use Case for Changing the PIN at the ATM

Use Case 1	Change of Card PIN by Customer at ATM
Short description	Customer changes the PIN of its ATM card at the ATM to a number of her/his choice
Actors	Customer (interacting with ATM Machine Software x.x, Core banking system)
Preconditions	Main Menu of ATM is displayed and ATM is in active state, core banking system available for change of PIN
Basic Flow <i>In an alternative flow, the customer could also authenticate first, then select the option from the menu</i>	<ol style="list-style-type: none"> 1. Customer selects “change PIN” from Main Menu 2. ATM displays “change PIN” screen and asks customer to insert card, card slot opens 3. Customer inserts card 4. Card is accepted by ATM 5. ATM asks for current PIN and new pin (twice) 6. Customer inserts current pin, then enters new pin twice 7. ATM confirms change of pin, spits out card 8. Customer takes card 9. ATM returns to main menu
Alternative flows	Card is not accepted, pin entered by customer is invalid, customer enters two different new pins, change of pin not successful at backend system, customer forgets to take card out (details need to be worked out ...)
Postconditions	Old PIN is inactive, new pin is active
NFR	Answering and reply times, clarity of user menu, availability time of new pin
Author/Date/Revision History	JK, 26.09.2020. Version 1.0

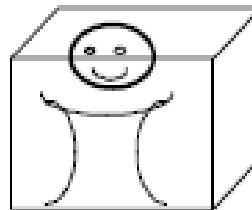
«An Actor with a goal calls on the responsibilities of another»

Primary Actor
person or system
with goal for SuD



- Goal 1
- Goal 2
- ... action 1
- ⋮
- backup goal
for Goal 2

System under design
could be any system



(Interaction 1)

Responsibility

- Goal 1
- ...action 1

Secondary Actor
other system against
which SuD has a goal



(Interaction 2)

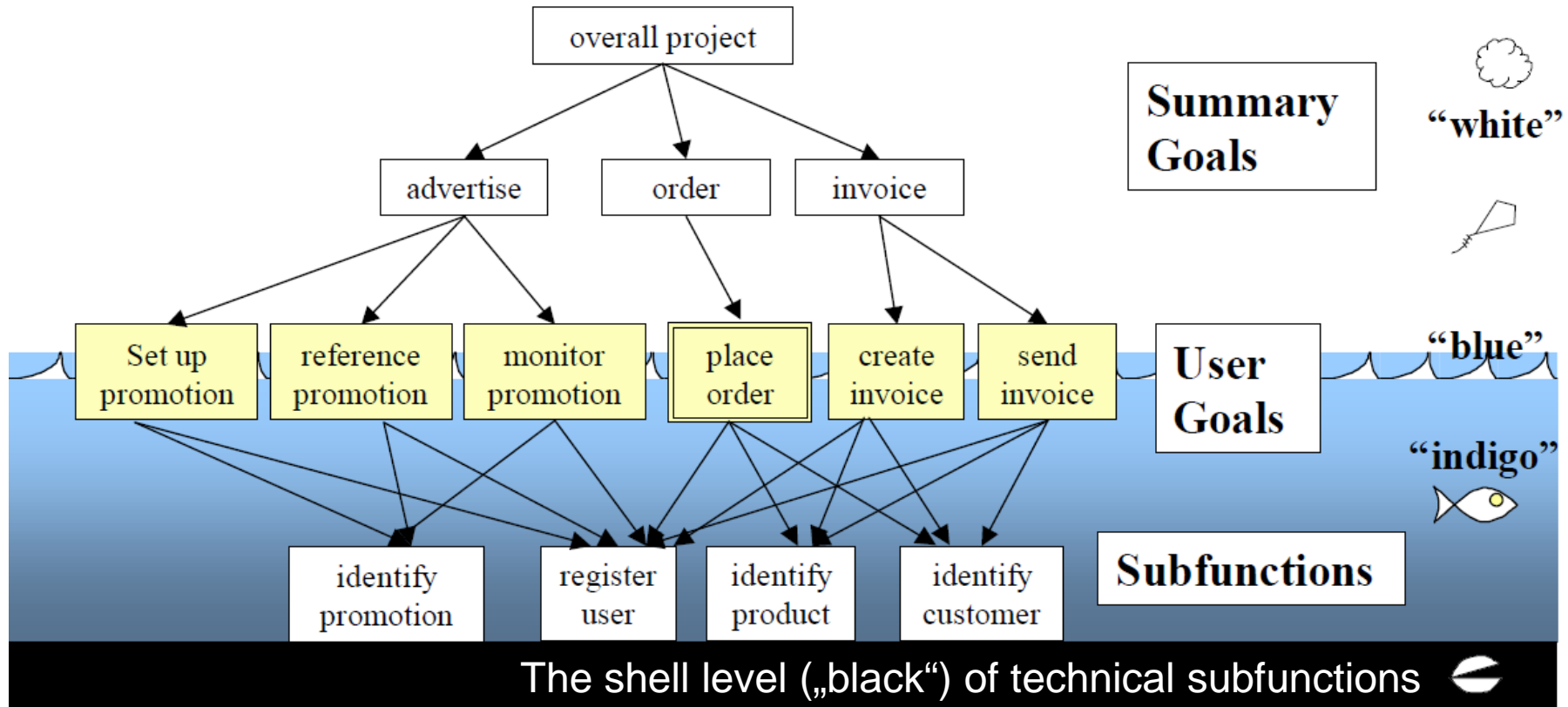
Responsibility

SuD – system under discussion

Cockburn, S. 24



3 Main Levels of Goals: White – Blue – Indigo



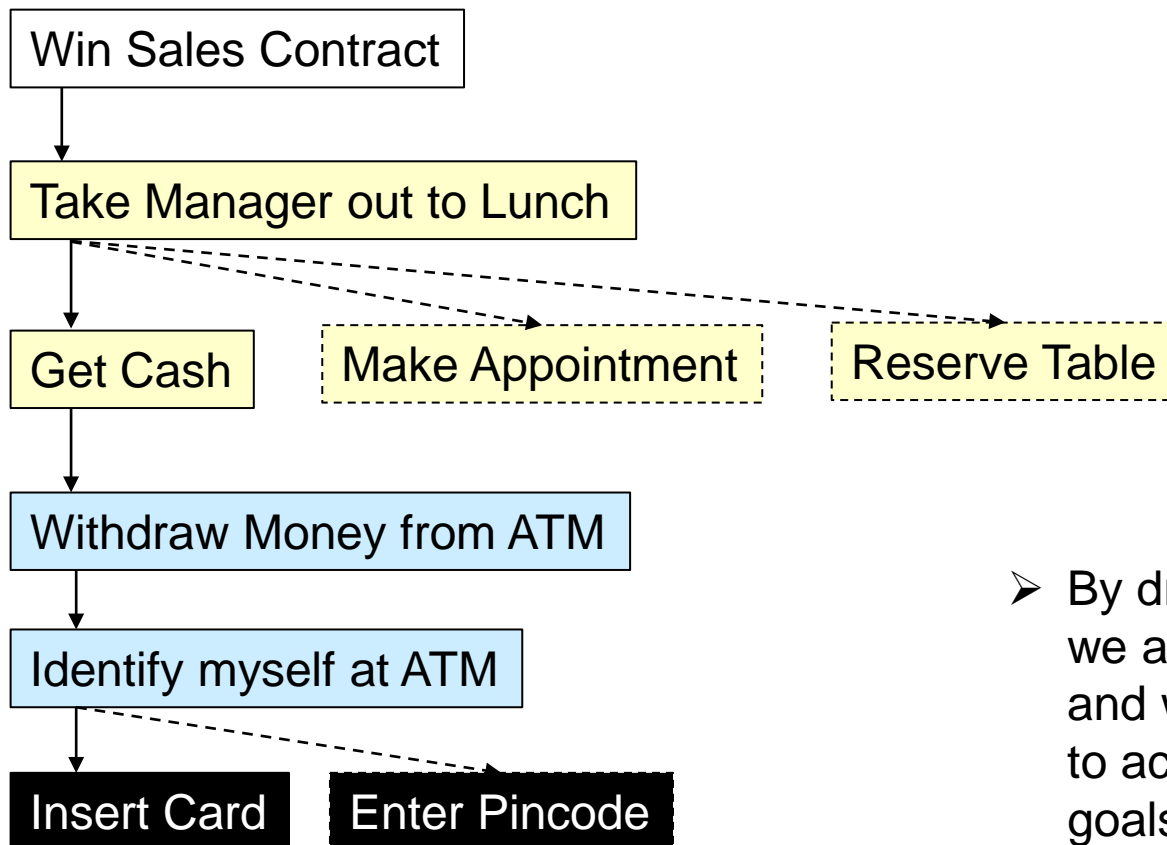
Formulate Causal Chains to Identify Goal Hierarchies

1. *"I want this sales contract. To do that I have to take this manager out to lunch. To do that I have to get some cash. To do that I have to withdraw money from this ATM. To do that I have to get it to accept my identity. To do that I have to get it to read my ATM card. To do that I have to find the card slot."*
2. *"I want to find the tab key so I can get the cursor into the address field, so I can put in my address, so I can get my personal information into this quote software, so I can get a quote, so I can buy a car insurance policy, so I can get my car licensed, so I can drive."*

Cockburn, S. 61

The Resulting Goal Hierarchy

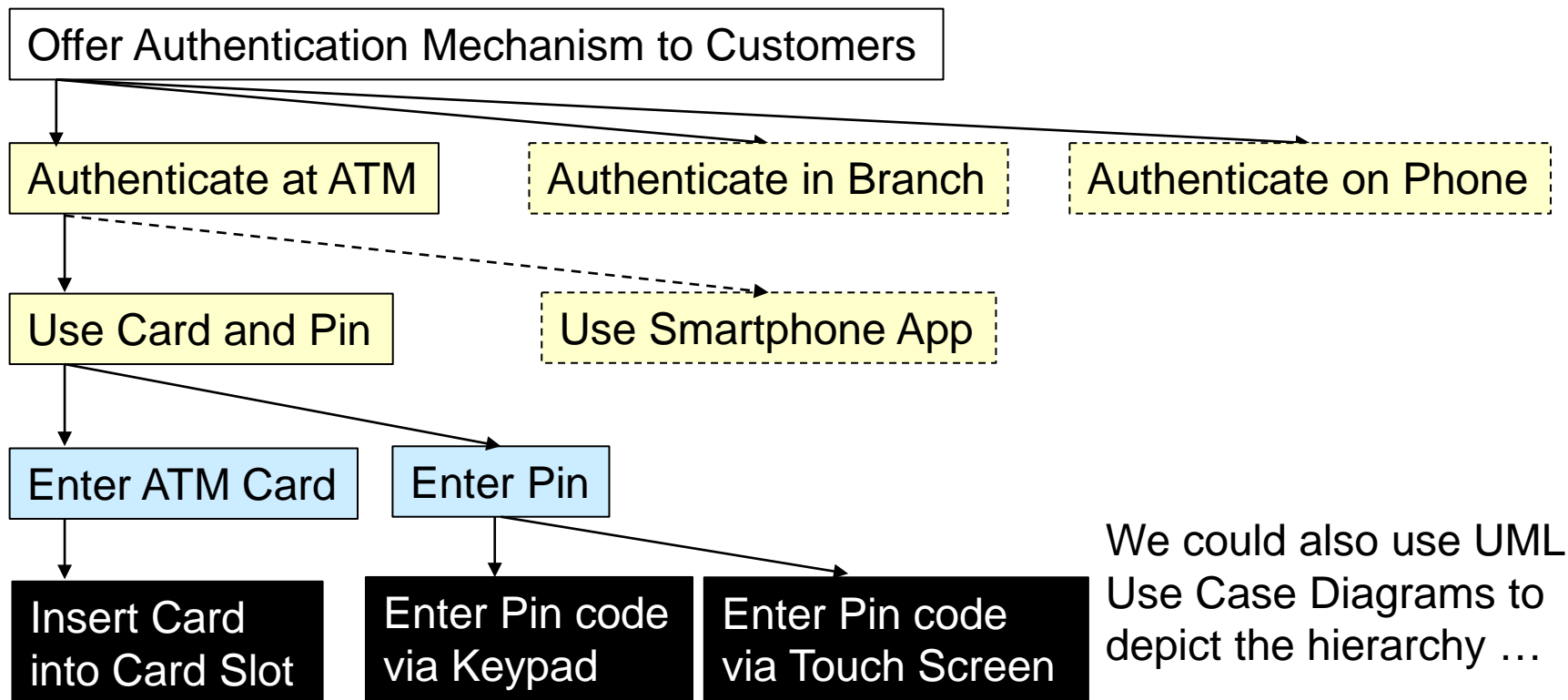
"I want this sales contract. To do that I have to take this manager out to lunch. To do that I have to get some cash. To do that I have to withdraw money from this ATM. To do that I have to get it to accept my identity. To do that I have to get it to read my ATM card. To do that I have to find the card slot."



- By drawing the goal hierarchy we also notice missing goals and we get ideas about options to achieve goals, alternative goals, optional goals,

Goal Hierarchy for Bank Authentication

As a bank, we need mechanisms to authenticate our customers across a variety of interaction channels, e.g. in the bank branch, at the ATM, on the phone...



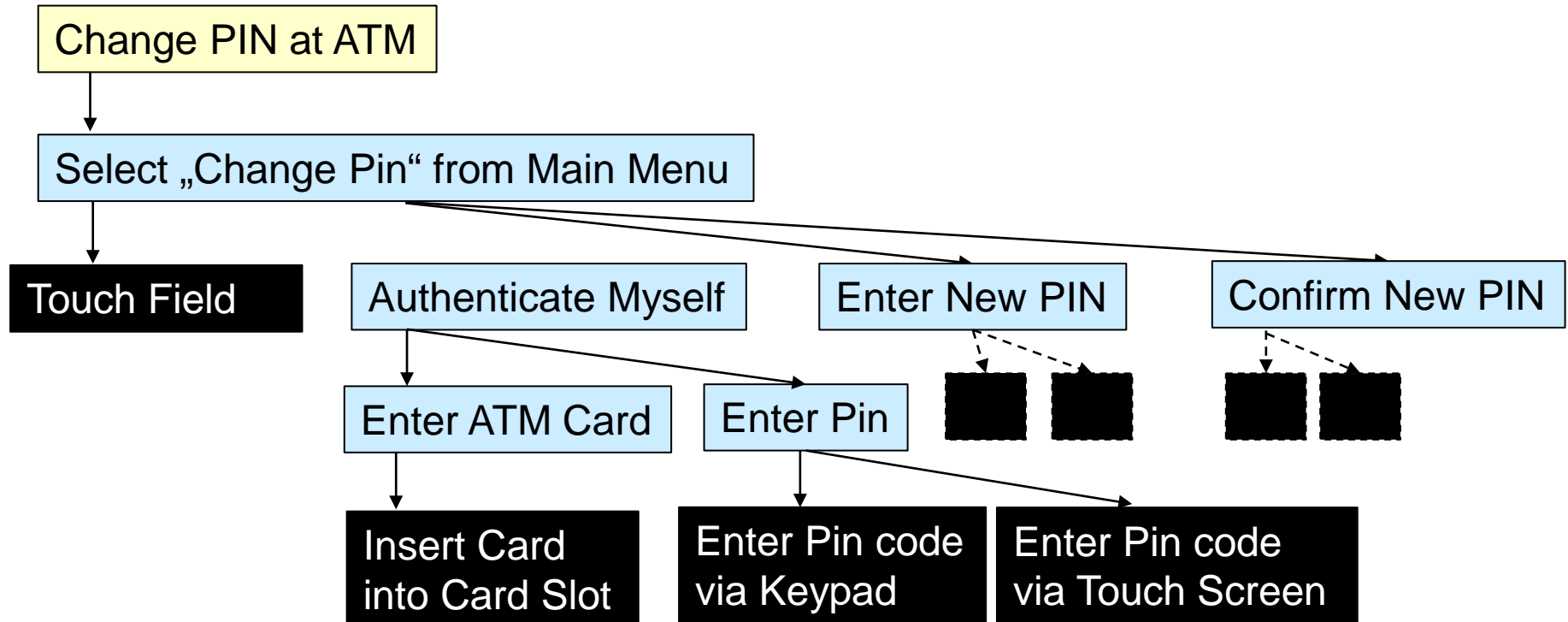
Goal Hierarchy for PIN Change

“I want to change my pin at the ATM. To do so, I need to select “change pin” from the main menu displayed at the machine. To do so, I need to touch the corresponding field on the touch screen.

Then, I need to authenticate myself. To do so, I need to, ...

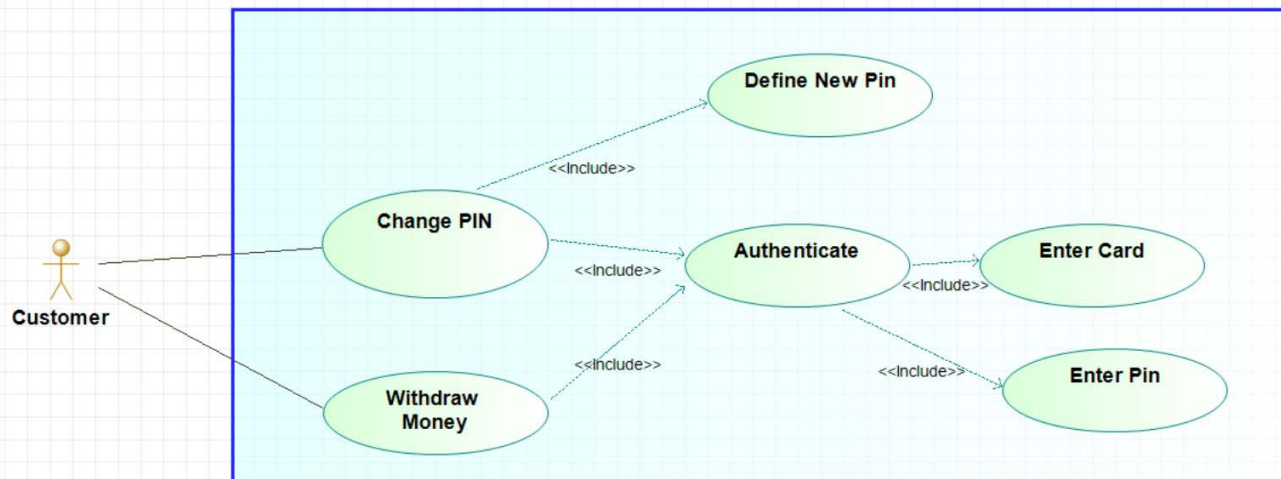
Then, I need to enter the new pin. To do so, I ...

Then I need to confirm the new pin. To do so, I ...”

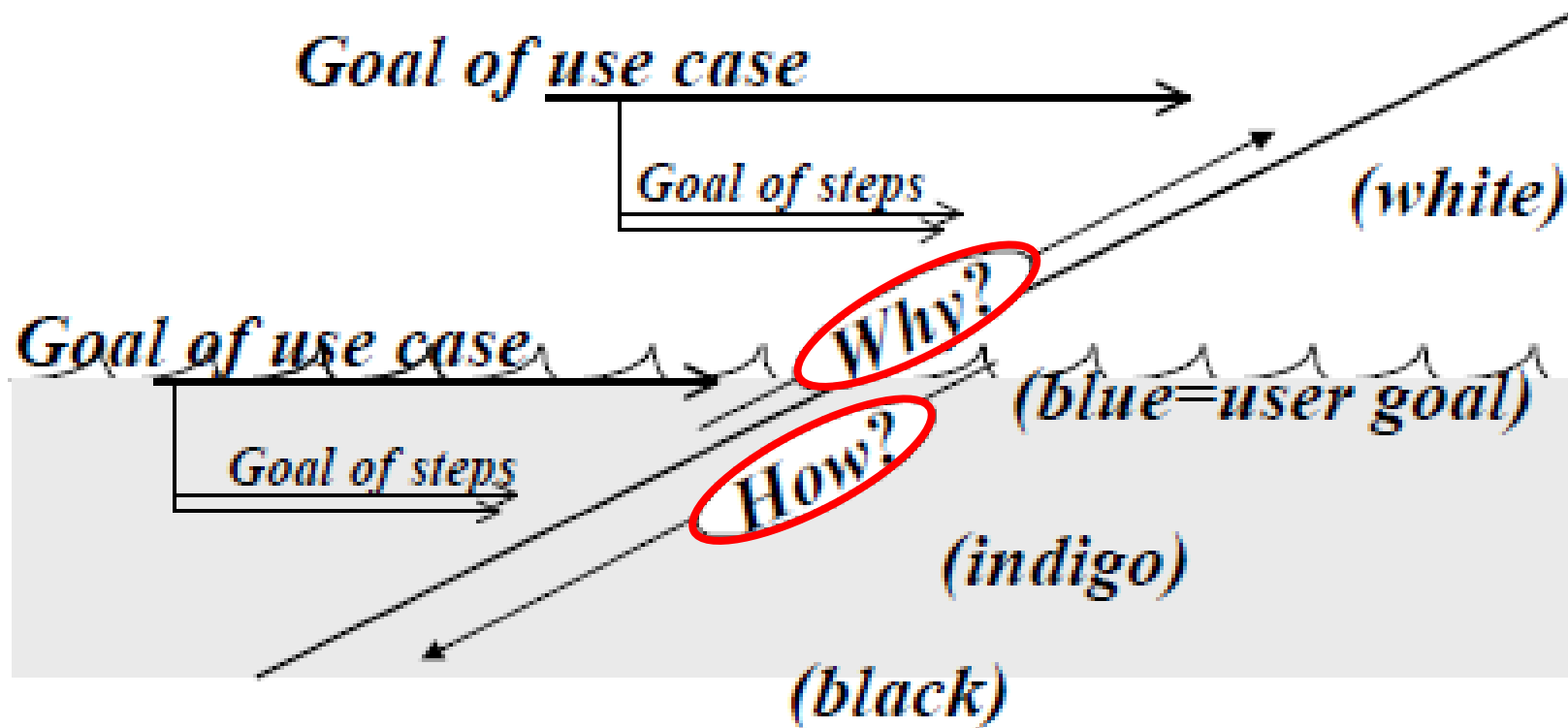


Goal Hierarchy in a UML Use Case Diagram

- The inclusion in use case diagrams can be used to show hierarchical relationships between use cases
- In the example, the authentication could be defined as a separate use case as it occurs in various interactions of the customer with the ATM
- However, entering card or pin or defining a new pin does not seem to be not meaningful as a separate use case as they are too fine-granular and only an intermediate step to achieve a user goal - however, if the processes of entering card and pin involve sophisticated flows that occur several times in the system, we could still consider to make them separate use cases to ensure the reusability and documentation of these technical functions



The use case goal is higher level than the steps. They sit on a gradient.



Cockburn, S. 69



© JK

Use Case Example on Level «White»

Use Case 18 Operate an Insurance Policy+

Primary Actor: The customer

Scope: The insurance company ("MyInsCo")

Level: Summary ("white")

Steps:

1. Customer gets a quote for a policy.
2. Customer buys a policy.
3. Customer makes a claim against the policy.
4. Customer closes the policy.

«White» vs. «Blue»

Use Case 6 Add New Service (Enterprise)

Primary Actor: Customer

Scope: MyTelCo

Level: Summary

1. Customer calls MyTelCo, requests new service . . .
2. MyTelCo delivers . . . etc. . . .

Use Case 7 Add New Service (Acura)

Primary Actor: Clerk for external customer

Scope: Acura

Level: User goal

1. Customer calls in, clerk discusses request with customer.
2. Clerk finds customer in Acura.
3. Acura presents customer's current service package . . . etc. . . .

Use Case Example on Level «Blue»

Use Case 13 Serialize Access to a Resource

Primary Actor: Service Client object

Scope: Concurrency Service Framework (CSF)

Level: User goal

Main Success Scenario:

1. Service Client asks a Resource Lock to give it specified access.
2. The Resource Lock returns control to the Service Client so that it may use the Resource.
3. Service Client uses the Resource.
4. Service Client informs the Resource Lock that it is finished with the Resource.
5. Resource Lock cleans up after the Service Client.

Extensions:

- 2a. Resource Lock finds that Service Client already has access to the resource:
 - 2a1. Resource Lock applies a lock conversion policy (Use Case 14) to the request.
- 2b. Resource Lock finds that the resource is already in use:
 - 2b1. The Resource Lock applies a compatibility policy (Use Case 15) to grant access to the Service Client.
- 2c. Resource Locking Holding time limit is nonzero:
 - 2c1. Resource Lock starts the holding timer.

...

Use Case Example on Level «Indigo»

Use Case 14 Apply a Lock Conversion Policy

Primary Actor: Client object

Scope: Concurrency Service Framework (CSF)

Level: Subfunction

Main Success Scenario:











1. Resource Lock verifies that request is for exclusive access.
2. Resource Lock verifies that Service Client already has shared access.
3. Resource Lock verifies that there is no Service Client waiting to upgrade access.
4. Resource Lock verifies that there are no other Service Clients sharing the resource.
5. Resource Lock grants Service Client exclusive access to the resource.
6. Resource Lock increments Service Client lock count.

Extensions:

- 1a. Resource Lock finds that the request is for shared access:
 - 1a1. Resource Lock increments lock count on Service Client.
 - 1a2. Success!
- 2a. Resource Lock finds that the Service Client already has exclusive access:
 - 2a1. Resource Lock increments lock count on Service Client.
 - 2a2. Success!

Notation

Icons

Design Scope		Goal Level	
	Organization (black-box)		Very high summary
	Organization (white-box)		Summary
	System (black box)		User-goal
	System (white box)		Subfunction
	Component		Too low

For Goal Level, alternatively, append one of these characters to the use case name:

Append “+” to summary use case names.

Append “!” or nothing to user-goal use case names.

Append “–” to subfunction use case names.

How to Identify Use Cases?

1. Name the system scope and boundaries.
Track changes to this initial context diagram with the in/out list.
2. Brainstorm and list the primary actors.
Find every human and non-human primary actor, over the life of the system.
3. Brainstorm and exhaustively list user goals for the system.
The initial Actor-Goal List is now available.
4. Capture the outermost summary use cases to see who really cares.
Check for an outermost use case for each primary actor.
5. Reconsider and revise the summary use cases. Add, subtract, or merge goals.
Double-check for time-based triggers and other events at the system boundary.
6. Select one use case to expand.
Consider writing a narrative to learn the material.

7. Capture stakeholders and interests, preconditions and guarantees.
The system will ensure the preconditions and guarantee the interests.
8. Write the main success scenario (MSS).
Use 3 to 9 steps to meet all interests and guarantees.
9. Brainstorm and exhaustively list the extension conditions.
Include all that the system can detect and must handle.
10. Write the extension-handling steps.
Each will end back in the MSS, at a separate success exit, or in failure.
11. Extract complex flows to sub use cases; merge trivial sub use cases.
Extracting a sub use case is easy, but it adds cost to the project.
12. Readjust the set: add, subtract, merge, as needed.
Check for readability, completeness, and meeting stakeholders' interests.

Use Case: Login

Exercise: Login Use Case

Your new
colleague
sends you the
Use Case
Login for
evaluation.

Give him
feedback and
corrections.

This use case describes the process by which users log in to the order-processing system. It also sets up access permissions for various categories of users.

Flow of Events:

Basic Path:

1. The use case starts when the user starts the application.
2. The system will display the Login screen.
3. The user enters a username and password.
4. The system will verify the information.
5. The system will set access permissions.
6. The system will display the Main screen.
7. The user will select a function.
8. While the user does not select Exit loop
9. If the user selects Place Order, Use Place Order.
10. If the user selects Return Product, Use Return Product.
11. If the user selects Cancel Order, Use Cancel Order.
12. If the user selects Get Status on Order, Use Get Status.
13. If the user selects Send Catalog, Use Send Catalog.
14. If the user selects Register Complaint, Use Register Complaint.
15. If the user selects Run Sales Report, Use Run Sales Report.
- end if
16. The user will select a function.
- end loop
17. The use case ends.

Exercise: Cash withdrawals from ATMs

1. Develop a goal hierarchy and define 3 use cases at the levels "white-blue-indigo".
2. What are the major errors when doing a cash withdrawal at an ATM considered at level "blue" ?
3. Define possible acceptance criteria and tests for the goals and use cases defined above.

How to Proceed (Cockburn, p. 90 ff.)

- Use Simply formulated sentences to describe active actions
 - Subject – verb – object: The System deducts the amount from the ...
- Subject = «Who is on it?»

What is the turn of the subject? How does it transfer control to another subject?
or Finish the job and «clean up the dirt"
- Take bird's eye view - not a system's view
 - Bad: „Get ATM card and pin number“
 - Good: „The customer puts in the ATM Card and PIN“
- Goals correspond to intentions
 - Bad: „System asks for name. User enters name...“
 - Good: „User enters name and address (to specify delivery address)“
- Demonstrate how the process is progressing towards the goal of the use case.

How to Proceed (continued)

- Make the purpose of validations clear
 - Bad: „System checks the password. If ... then ... else.“
 - Good: „The system verifies that the password is correct.“
- Describe repeating and branching activities precisely (do not use pseudo code)
 - 3a. Holding Timer expires before the Client informs the Resource Lock that it is finished:
 - 3a1. Resource Lock sends an Exception to the Client's process.
 - 3a2. Fail!
 - 4a. Resource Lock finds nonzero lock count on Service Client:
 - 4a1. Resource Lock decrements the reference count of the request.
 - 4a2. Success!
 - 5a. Resource Lock finds that the resource is currently not in use:
 - 5a1. Resource Lock applies an access selection policy (Use Case 16) to grant access to any suspended service clients.
 - 5b. Holding Timer is still running:
 - 5b1. Resource Lock cancels Holding Timer.

Best Practices

Write something readable.

Casual, readable use cases are still useful, whereas unreadable use cases won't get read.

Work breadth-first, from lower precision to higher precision.

Precision Level 1: Primary actor's name and goal

Precision Level 2: The use case brief, or the main success scenario

Precision Level 3: The extension conditions

Precision Level 4: The extension handling steps

For each step:

Show a goal succeeding.

Capture the actor's intention, not the user interface details.

Have an actor pass information, validate a condition, or update state.

Write between-step commentary to indicate step sequencing (or lack of).

Ask "why" to find a next-higher level goal.

For data descriptions (only put Precision Level 1 into the use case text):

Precision Level 1: Data nickname

Precision Level 2: Data fields associated with the nickname

Precision Level 3: Field types, lengths, and validations

When will we have captured all use cases?

- ◆ You have named *all the primary actors and all the user goals* with respect to the system.
- ◆ You have captured *all trigger conditions* to the system either as use case triggers or as extension conditions.
- ◆ You have written all the user-goal use cases, along with the *summary and sub-function use cases* needed to support them.
- ◆ Each use case is written clearly enough that
 - The sponsors *agree* that they will be able to tell whether or not it is actually delivered.
 - The users *agree* that it is what they want or can accept as the system's behavior.
 - The developers *agree* that they can actually develop that functionality.
- ◆ The sponsors *agree* that the use case set covers all they want (for now).

Consistency of User Story and Use Case

Use Case: *Titel*

Description : *Summary in one sentence*

Level: High Level Summary, Summary, User Goal,
Sub-Function, Low Level

Main Actuator: Actuator

Preconditions: *Preconditions*

Postconditions: *Postconditions (Success, Minimal)*

Main scenario

1. *Describe Step 1*
2. *Describe Step 2*

Alternative Scenarios

As a [role]

I can [function]

so that [rationale]

+ Acceptance Criteria

Summary

- Use cases and user stories address different aspects of the system (how vs. what/why)
- Focus on uniform terminology, make yourself aware of levels of abstraction and write uniformly within and across levels
 - Goal hierarchies make levels explicit and help to sort the set of use cases
- Precisely formulate use cases and use stories and if both are used, align them correctly with each other
 - Make them SMART
 - Formulate measurable acceptance criteria

Working Questions

1. By which means can we capture requirements?
2. What is your personal stake on the relationship between user stories and use cases?
3. What helps in finding the right level of detail/abstraction in a user story?
4. Why are measurable acceptance criteria so important when using user stories?
5. Why should use cases be formulated at different levels of detail? Which levels are useful?
6. How are goals and steps related in a use case?