System Functionality

Architectural Thinking for Intelligent Systems

Winter 2019/2020

Prof. Dr. habil.Jana Koehler

Agenda

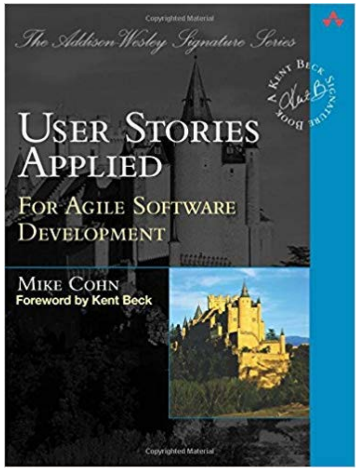
- Functional requirements from an architectural perspective
  - The importance of negotiation
- User stories vs. use cases
- Techniques for writing good user stories & use cases
- Defining goal hierarchies
- Measurable acceptance criteria

Tutorial Assignment 4:

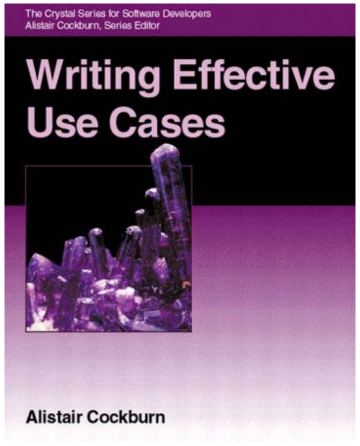
- We dive deeper into the functional requirements our system has to meet and formulate user stories and use cases.
- We analyze functional dependencies by creating a goal hierarchy.
- Remember to make your requirements SMART.

Two established Methods to capture Functional Requirements

- User stories


The Addison-Wesley Signature Series
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck
A Kent Beck Signature Book

- Use cases


The Crystal Series for Software Developers
Alistair Cockburn, Series Editor
Writing Effective Use Cases
Alistair Cockburn

User Stories Revisited

AGILE - SCRUM
I CAN'T GIVE YOU ALL OF THESE FEATURES IN THE FIRST VERSION.
scottadams@aol.com
www.dilbert.com
AND EACH FEATURE NEEDS TO HAVE WHAT WE CALL A "USER STORY."
© 2002 United Feature Syndicate, Inc.
1/10/03
OKAY, HERE'S A STORY: YOU GIVE ME ALL OF MY FEATURES OR I'LL RUIN YOUR LIFE.

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.mountaingoatsoftware.com/agile/user-stories

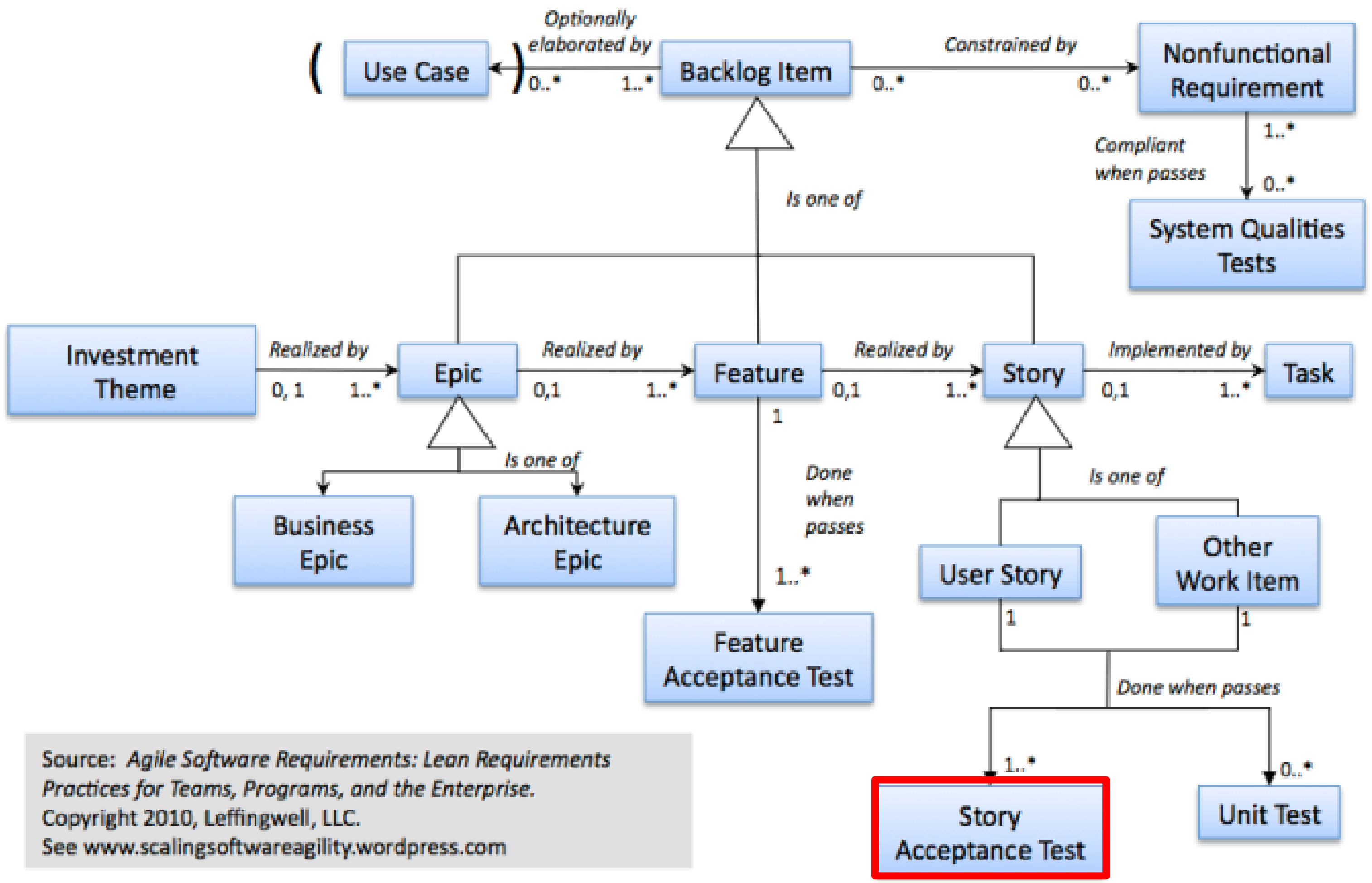
My one liner is that a story is a promise to have a conversation and a use case is the record of the conversation. If you think you need one. Jim Standley

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

Agile Enterprise Backlog Model



Source: *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise.* Copyright 2010, Leffingwell, LLC. See www.scalingsoftwareagility.wordpress.com

Epic vs. User Story

As a user, I can backup my entire hard drive.

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 dozens (or possibly hundreds), including these two:

As a power user, I can specify files or folders to backup based on file size, date created and date modified.

As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

http://www.mountaingoatsoftware.com/agile/user-stories

Good User Stories

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

As a student, I can find my grades online so that I don’t have to wait until the next day to know whether I passed.

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

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

Exercise: Which criteria are violated in these “user stories”?

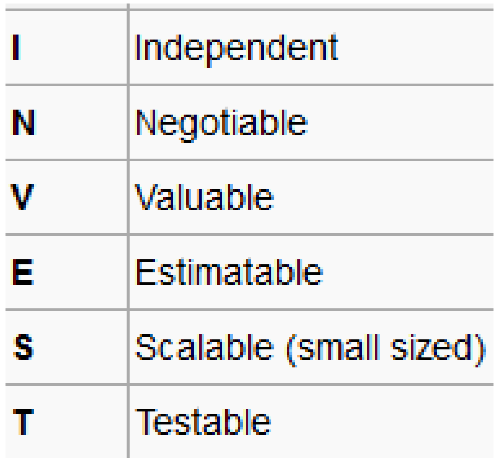
1) “Design brochure layout.”

2) “Write game rules.”

3) “I want the brochure to be colorful.”

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

5) “Play test the game.”



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.”

- Drawbacks: It’s not only about you!
- Better: Focus on your end users and stakeholders. “As a gourmet tourist, I want a list of highly-rated restaurants on the brochure.”
- Better: “As the Chicago Public Health Department, I want warnings about restaurants that serve raw ingredients so that tourists don’t get sick on our dime.”

“I want the brochure to be colorful.”

- Drawbacks: not *Independent*, not *Estimable* (without knowing other features of brochure), not *Small*.
  - This is an easy trap for those of us who grew up with the habit of writing “the JFIDM _shall_ comply with the IEEE-488 interface specification.”
- Better: Use “colorful” and other cross-cutting requirements as *acceptance criteria* on each of the specific features in the backlog they apply to.

“Design brochure layout.”

- Drawbacks: not *Independent*, no business *Value*. This is a *task* representing a horizontal architectural layer or phase. The architecture will be done in a vacuum, possibly contributing to analysis paralysis.

- Better: “As a dog owner, I can find a meal schedule on the brochure so I know whether this doggy day care center is appropriate for my hungry dog.”
  - This will lead to only the necessary amount of design to support this Sprint’s features. The layout might change the next Sprint, but rework is cheaper than no work.

“Write game rules.”

- Drawbacks: not *Independent*, no business *Value*, not *Small*.
- Better: “As a newbie game player, I want to know who goes first so we can start the game.”
- Better: “As a competitive gamer, I want a way to leapfrog my opposing players.”

“Play test the game.”

- Drawbacks: Not *Independent*. Encourages phasewise development.
- Better: Make testing, refactoring, etc. a default acceptance criteria on every Product Backlog Item.
  - But: If you failed to fully test and refactor in previous Sprints, you are in technical debt! You are already working on a legacy product. In this case you may need to make testing and refactoring first-class Product Backlog Items to make up for your sins. This practice is controversial, and technical debt repayment cannot honestly be called a *User* Story. A PBI that’s not a User Story may still be useful as a starting point for a conversation about how to reduce technical debt incrementally while continuing to deliver new value.

Exercise: Let us define some Acceptance Criteria

- **A bank customer can change his PIN.**
  - Acceptance Criteria: ….
- **As a student, I can find my grades online so that I don't have to wait until the next day to know whether I passed.**
  - Acceptance Criteria: ….
- **As a book shopper, I can read reviews of a selected book to help me decide whether to buy it.**
  - Acceptance Criteria: ….
- **As an author, I want the spell checker to ignore words with numbers so that only truly misspelled words are indicated.**
  - Acceptance Criteria: ….

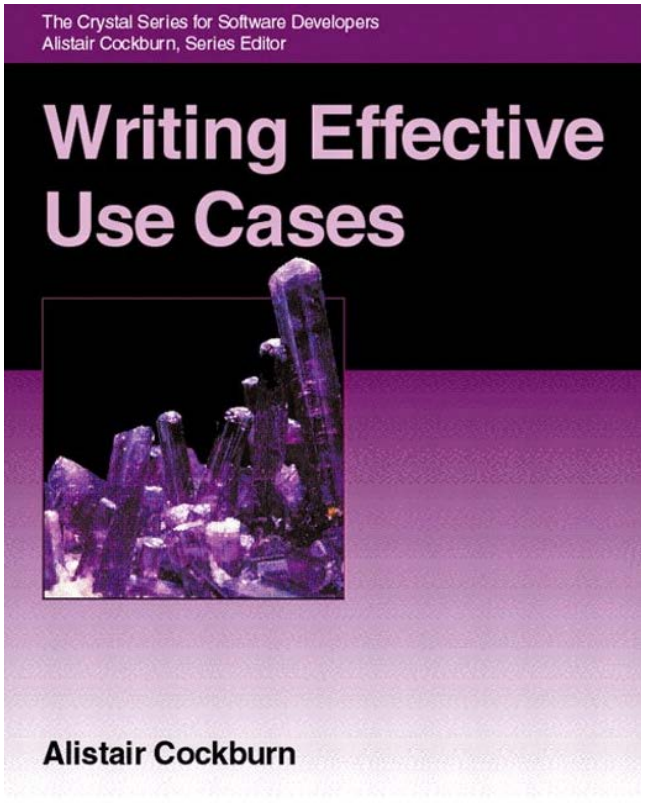
http://blogs.collab.net/agile/user-story-examples-and-counterexamples

Exercise: High or low Level of Detail?

As a <user type>, I want to <function> so that <benefit> .

1) As a consumer, I want the shopping cart functionality to easily purchase items online.
2) As an executive, I want to generate a report to understand which departments need to improve their productivity.
3) A team member can view the iteration status.
4) A team member can view a table of stories with rank, name, size, package, owner, and status.
5) A team member can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status.
6) A team member can view the iteration's stories and their status.
7) A team member can view the current burndown chart on the status page, and can click it for a larger view.
8) A team member can edit a task from the iteration status page.

Writing Effective Use Cases


The Crystal Series for Software Developers
Alistair Cockburn, Series Editor
Writing Effective Use Cases
Alistair Cockburn

Use Cases reduce Decision Space

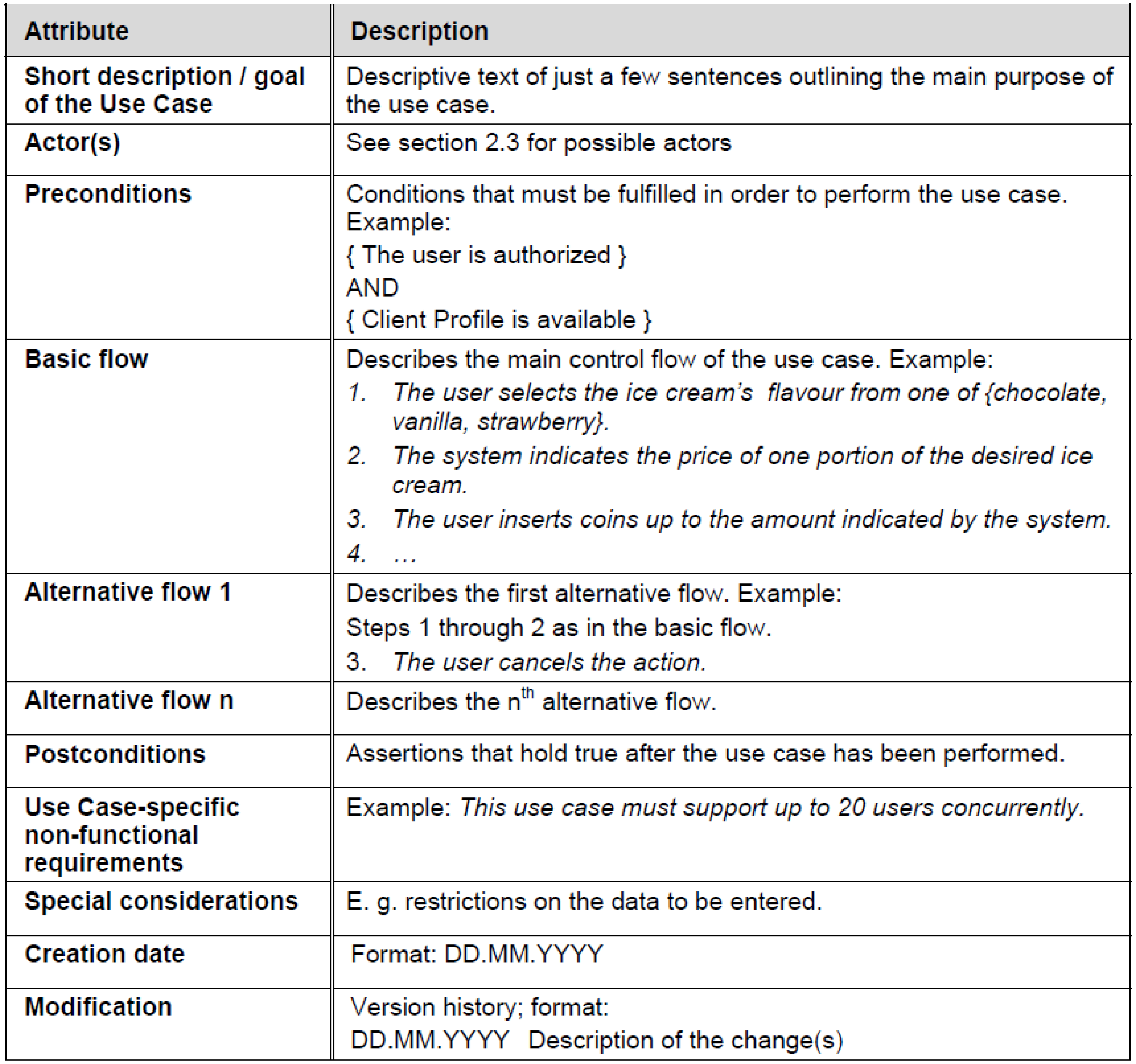
Welcome to the realm of use cases. Systems folks will have to analyze and carry out "thought experiments" and conceive how their system should precisely work in order to realize the agreed story.
Use cases are a very effective tool to do that.

So fundamentally given a concrete system definition, finite actors and business logic rules that do not contradict computation theory, there is a finite (but large) set of possibilities that can occur and they group together as scenarios and use cases.
Use cases are something really fundamental, but only when considered from a modeling perspective.

Nikhil Shah

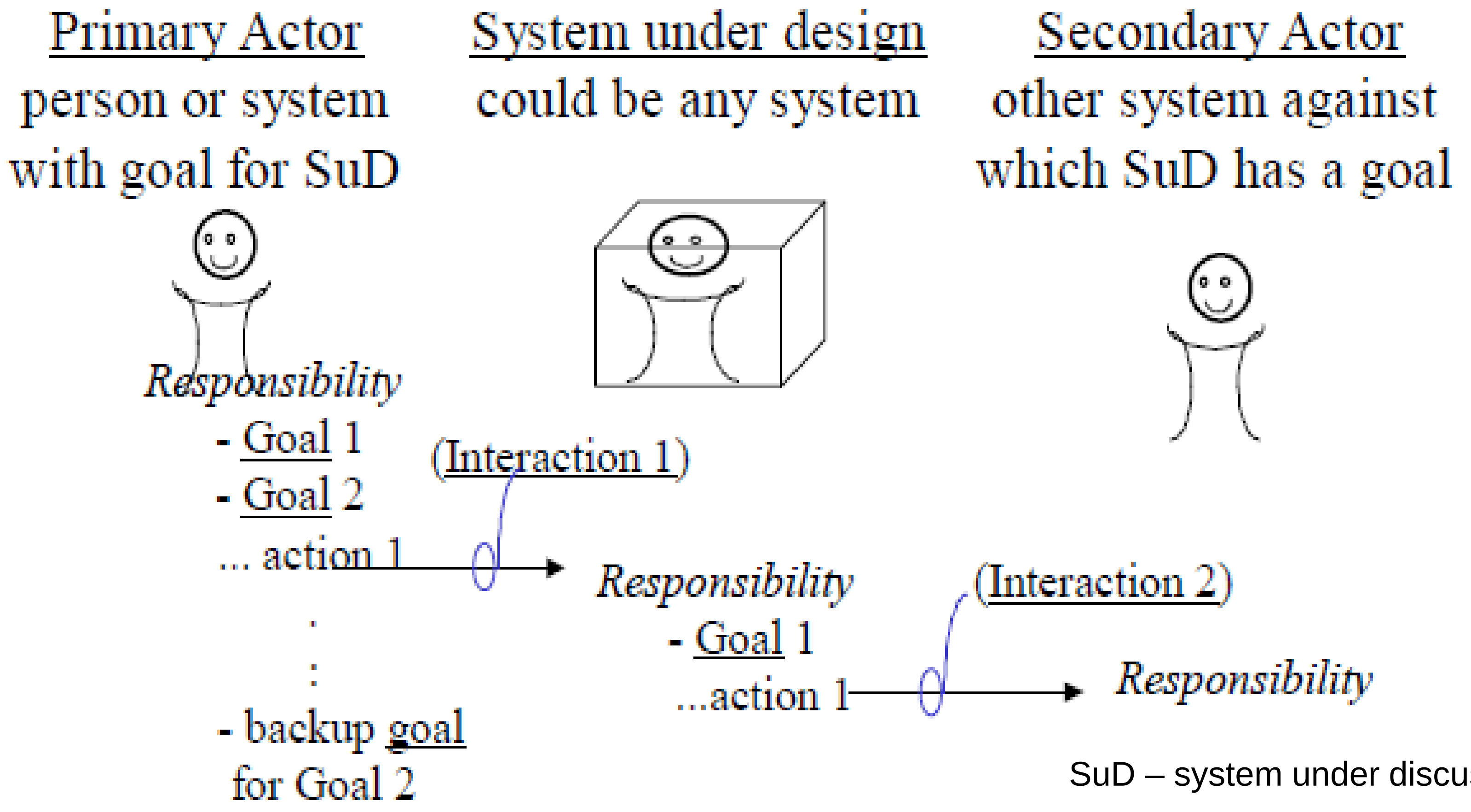
http://alistair.cockburn.us/A+user+story+is+to+a+use+case+as+a+gazelle+is+to+a+gazebo

A possible Use Case Template



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. The user selects the ice cream's flavour from one of {chocolate, vanilla, strawberry}.* *2. The system indicates the price of one portion of the desired ice cream.* *3. The user inserts coins up to the amount indicated by the system.* *4. …*
Alternative flow 1	Describes the first alternative flow. Example: Steps 1 through 2 as in the basic flow. 3. *The user cancels the action.*
Alternative flow n	Describes the nth alternative flow.
Postconditions	Assertions that hold true after the use case has been performed.
Use Case-specific non-functional requirements	Example: *This use case must support up to 20 users concurrently.*
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)

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



SuD – system under discussion

Cockburn, S. 24

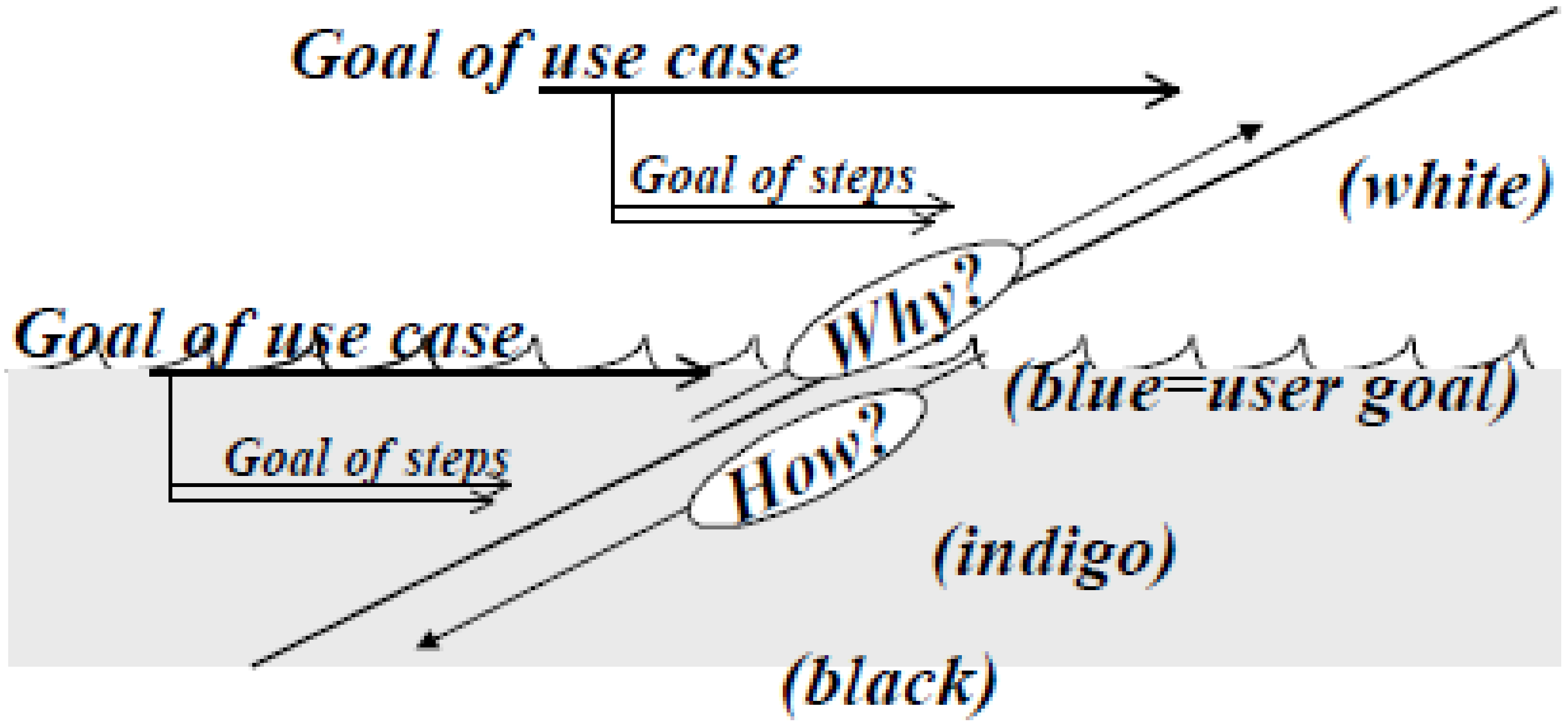
Goal Hierarchies

- *"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."*

- *"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."*

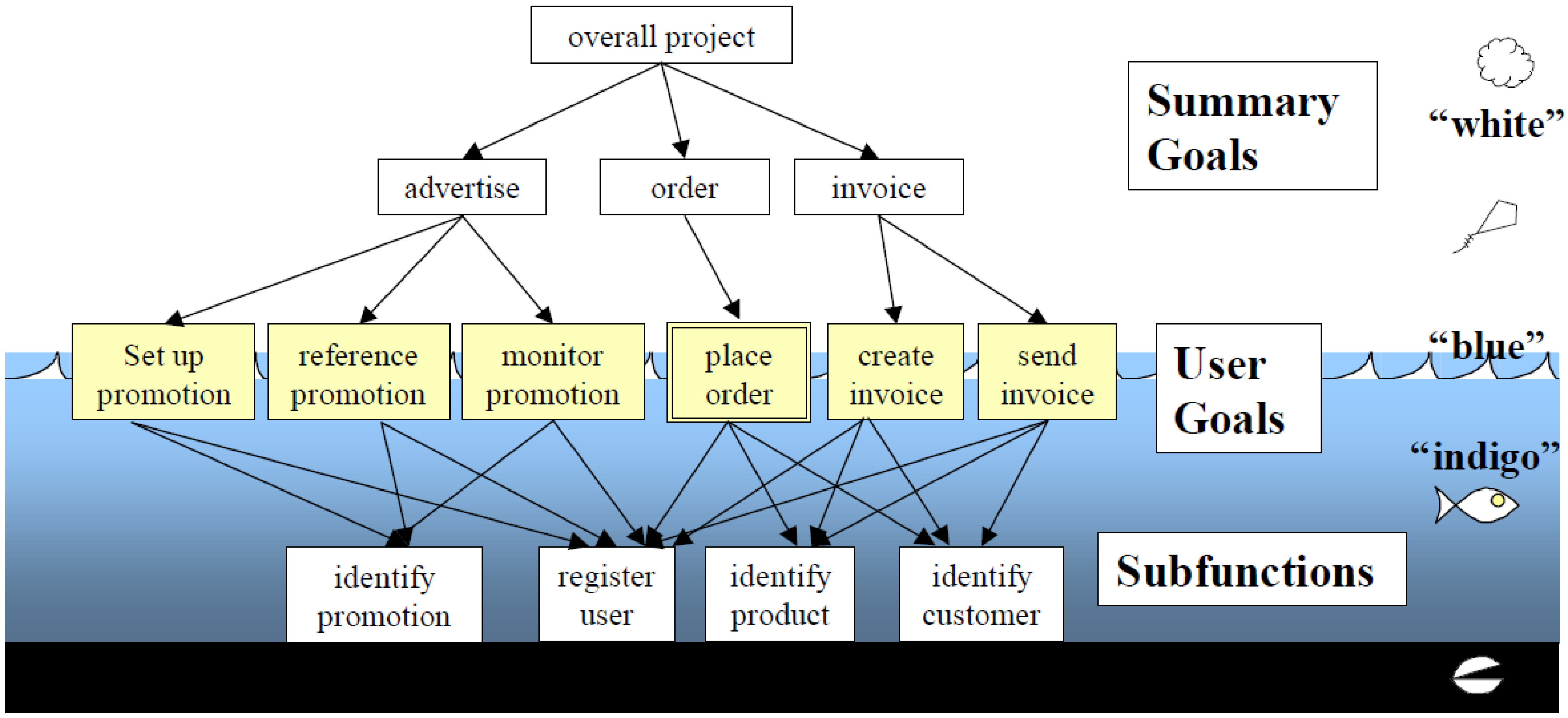
Cockburn, S. 61

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



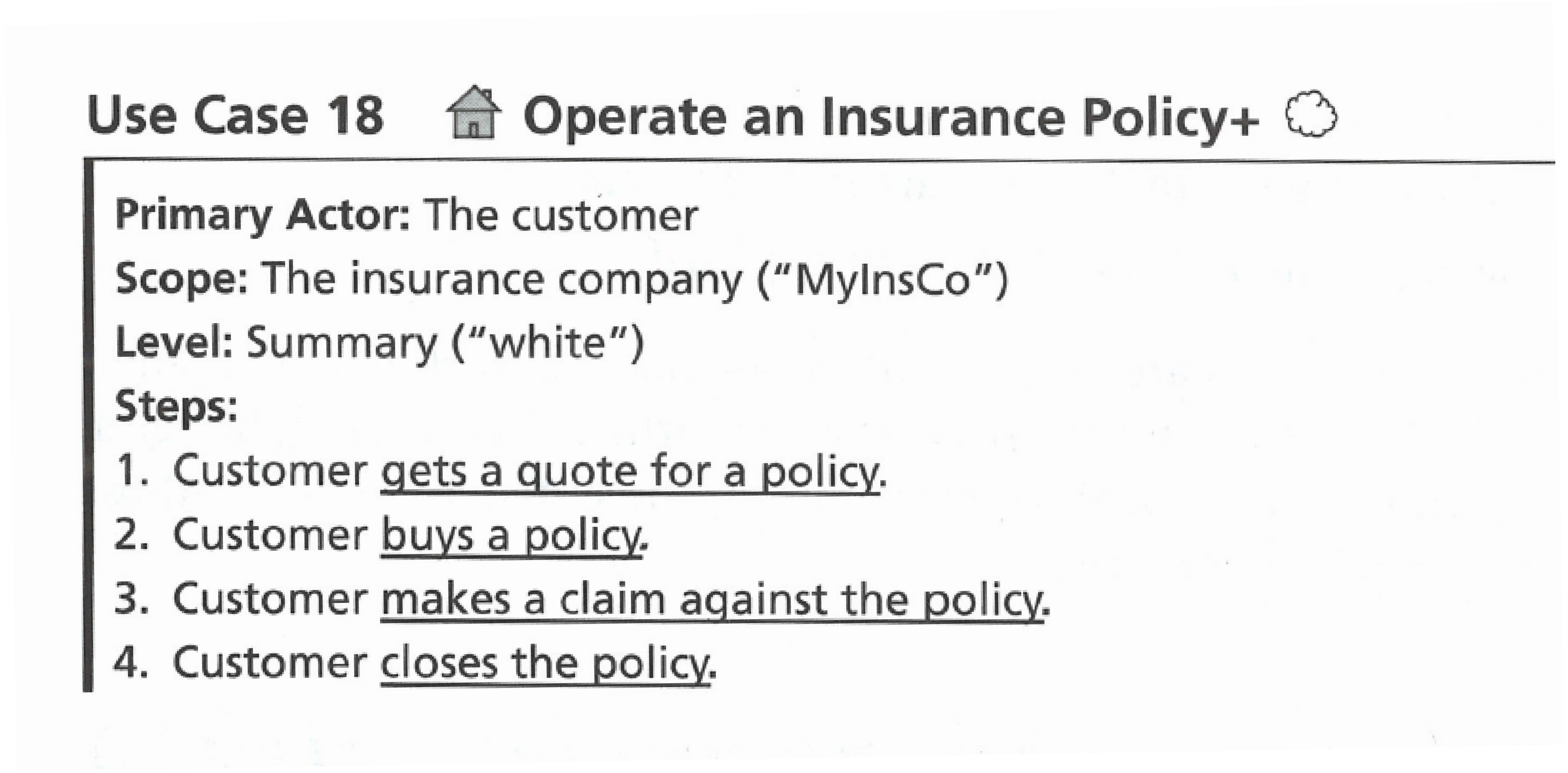
Cockburn, S. 69

3 main Levels of Goals: White – Blue – Indigo



Cockburn, S. 62

Use Case Example on Level «White»



Use Case 18 Operate an Insurance Policy+

Primary Actor: The customer

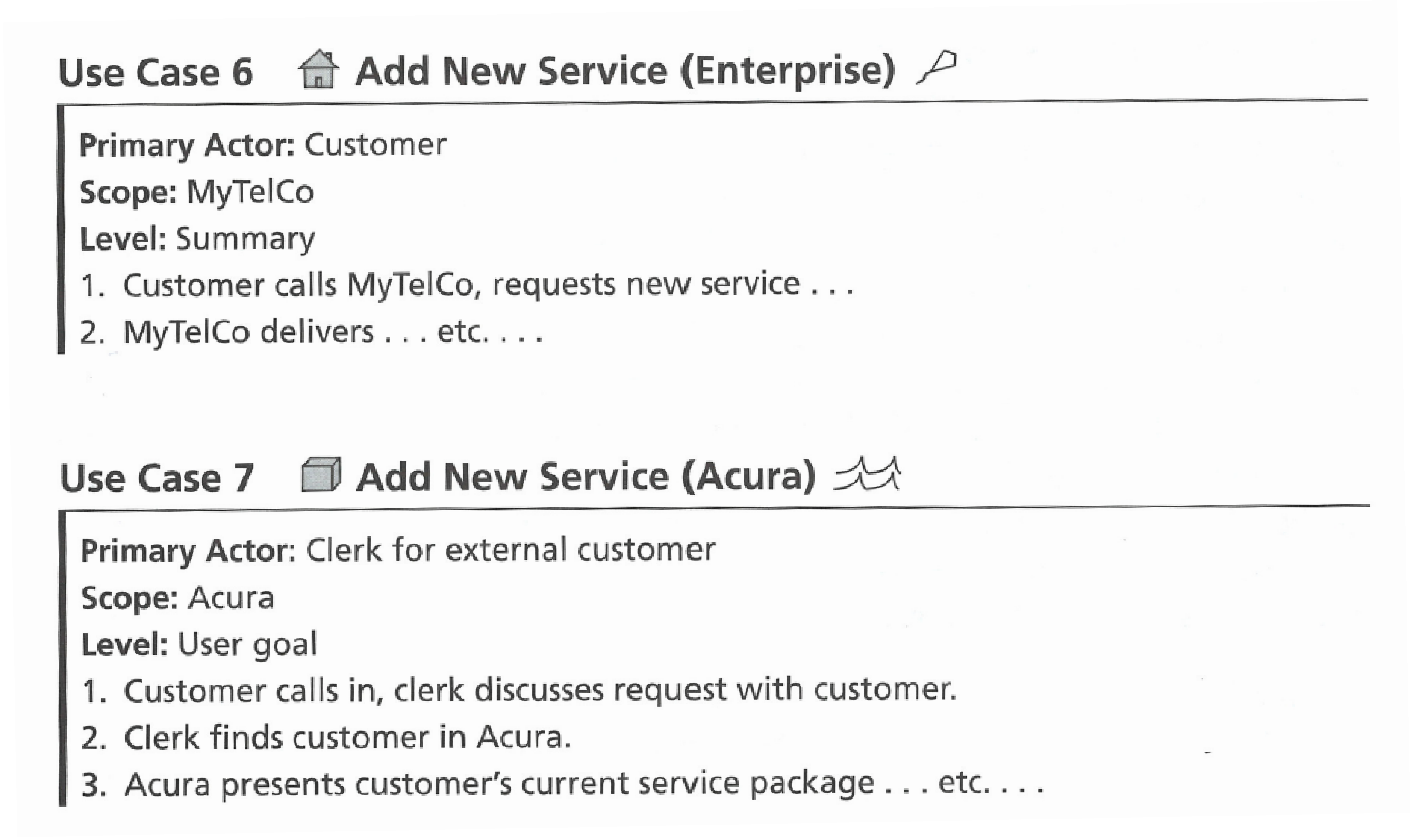
Scope: The insurance company ("MyInsCo")

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

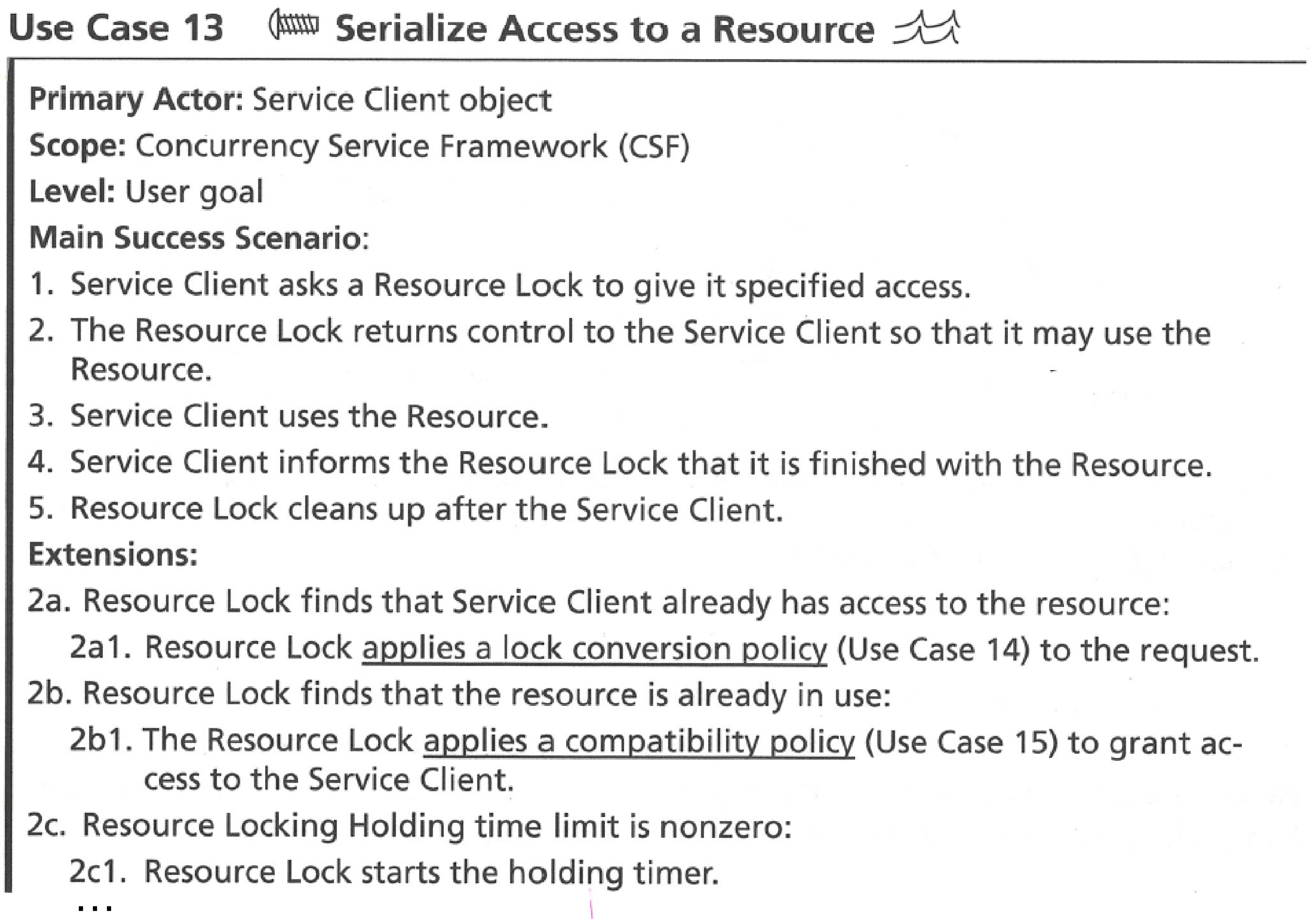
1. Customer calls MyTelCo, requests new service . . .
2. MyTelCo delivers . . . etc. . . .

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:

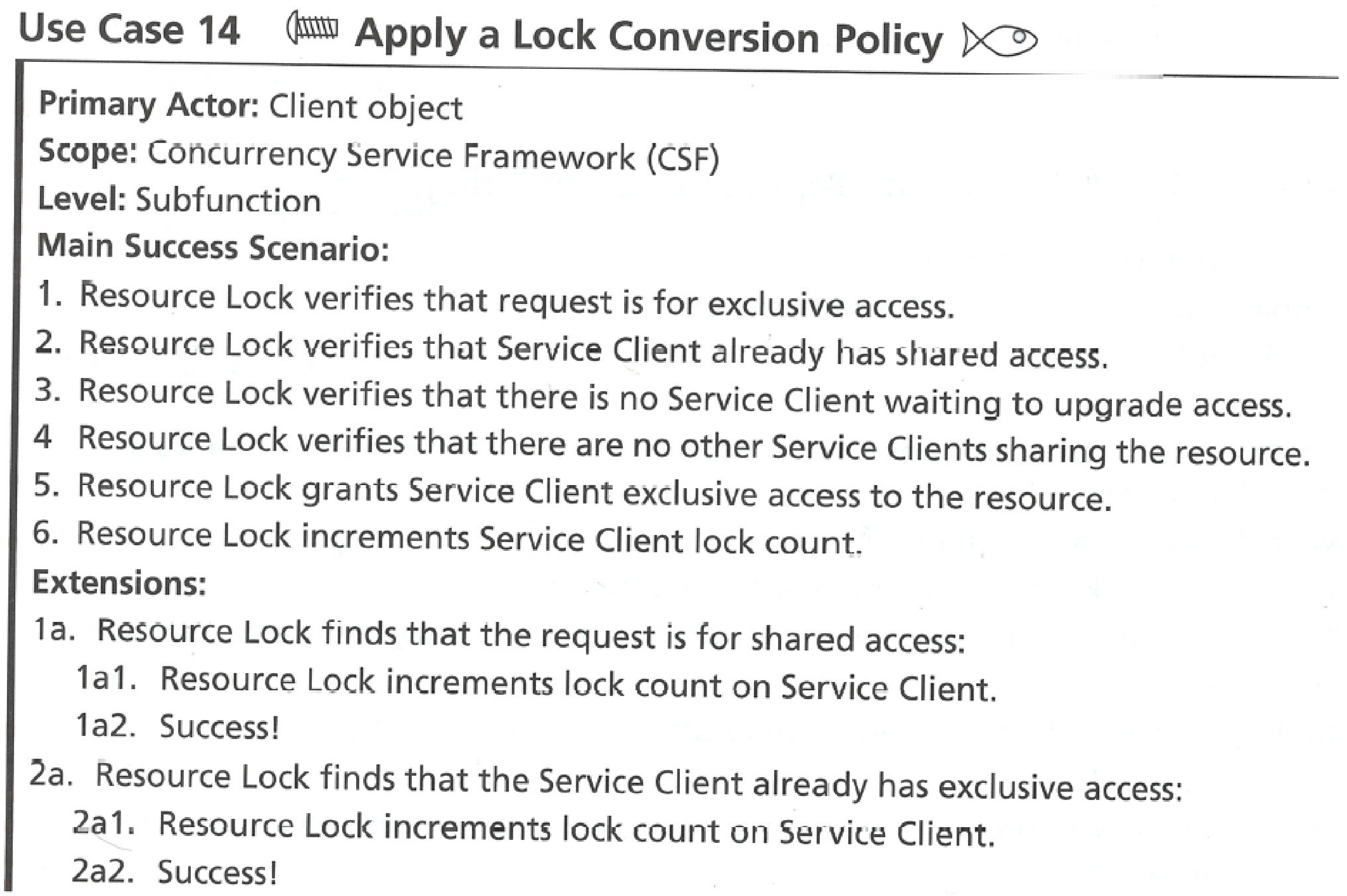
2b1. The Resource Lock applies a compatibility policy (Use Case 15) to grant access to the Service Client.

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:

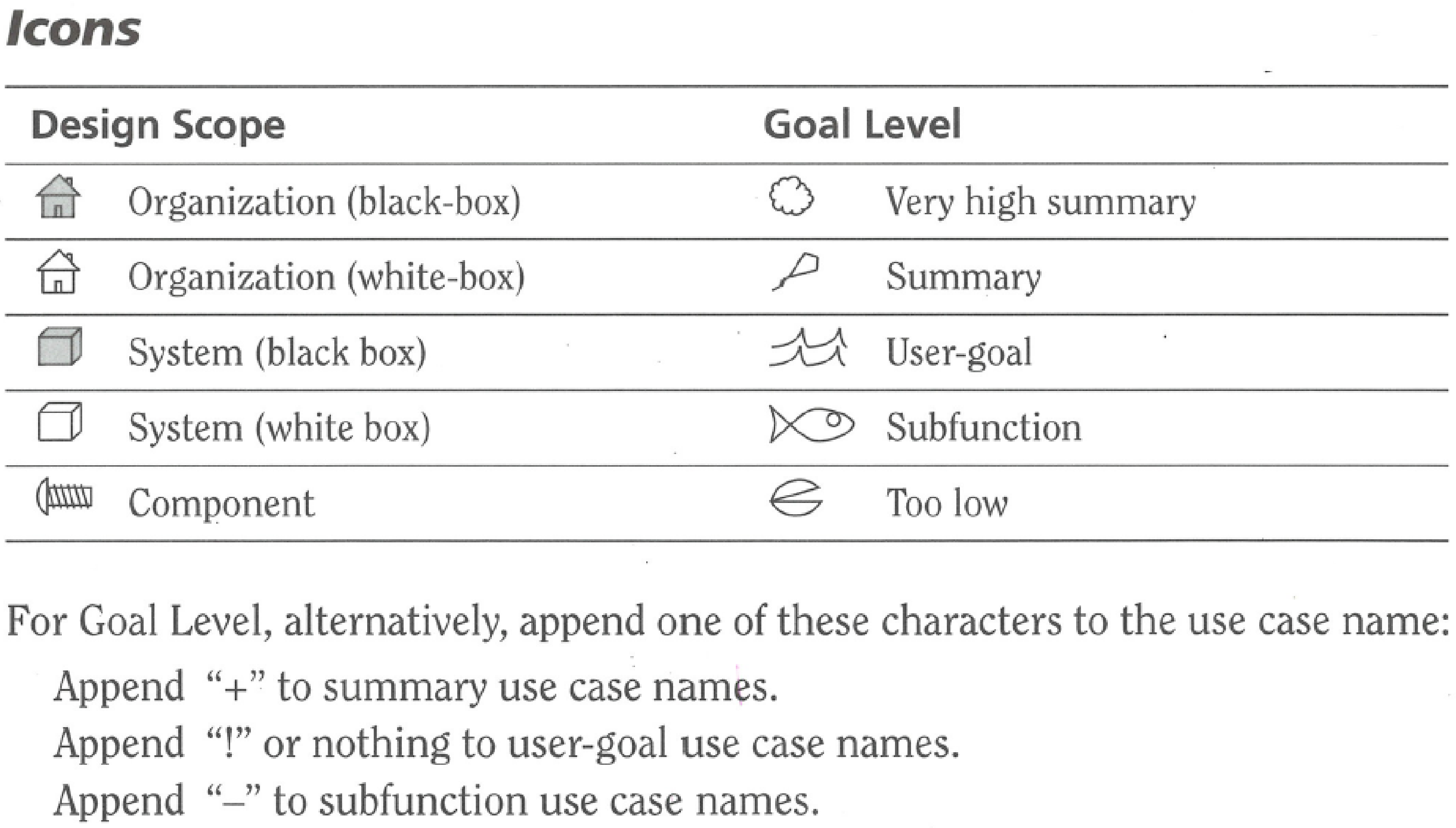
1. Resource Lock verifies that request is for exclusive access.
2. Resource Lock verifies that Service Client already has shared access.
3. Resource Lock verifies that there is no Service Client waiting to upgrade access.
4 Resource Lock verifies that there are no other Service Clients sharing the resource.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



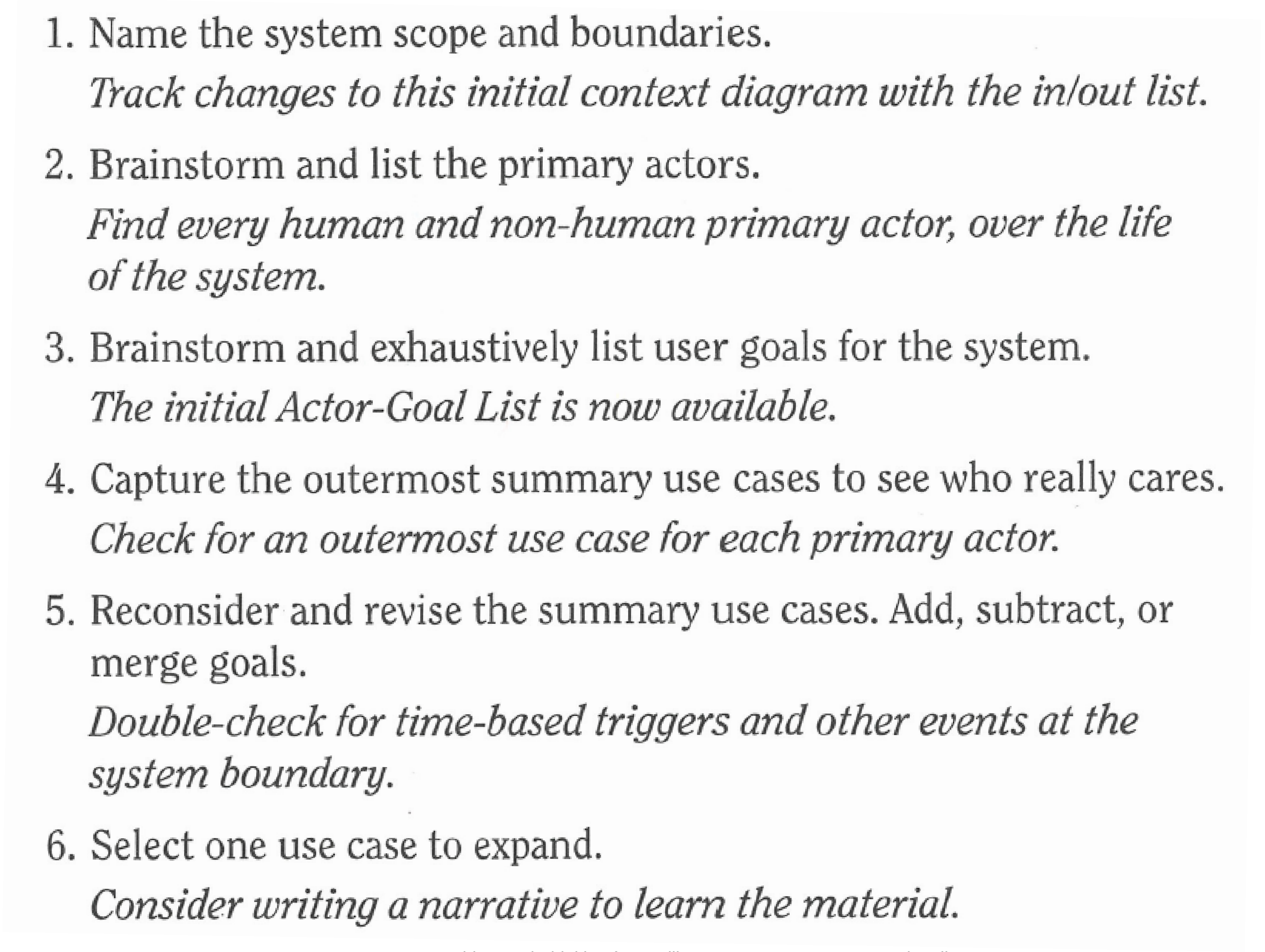
Icons

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.

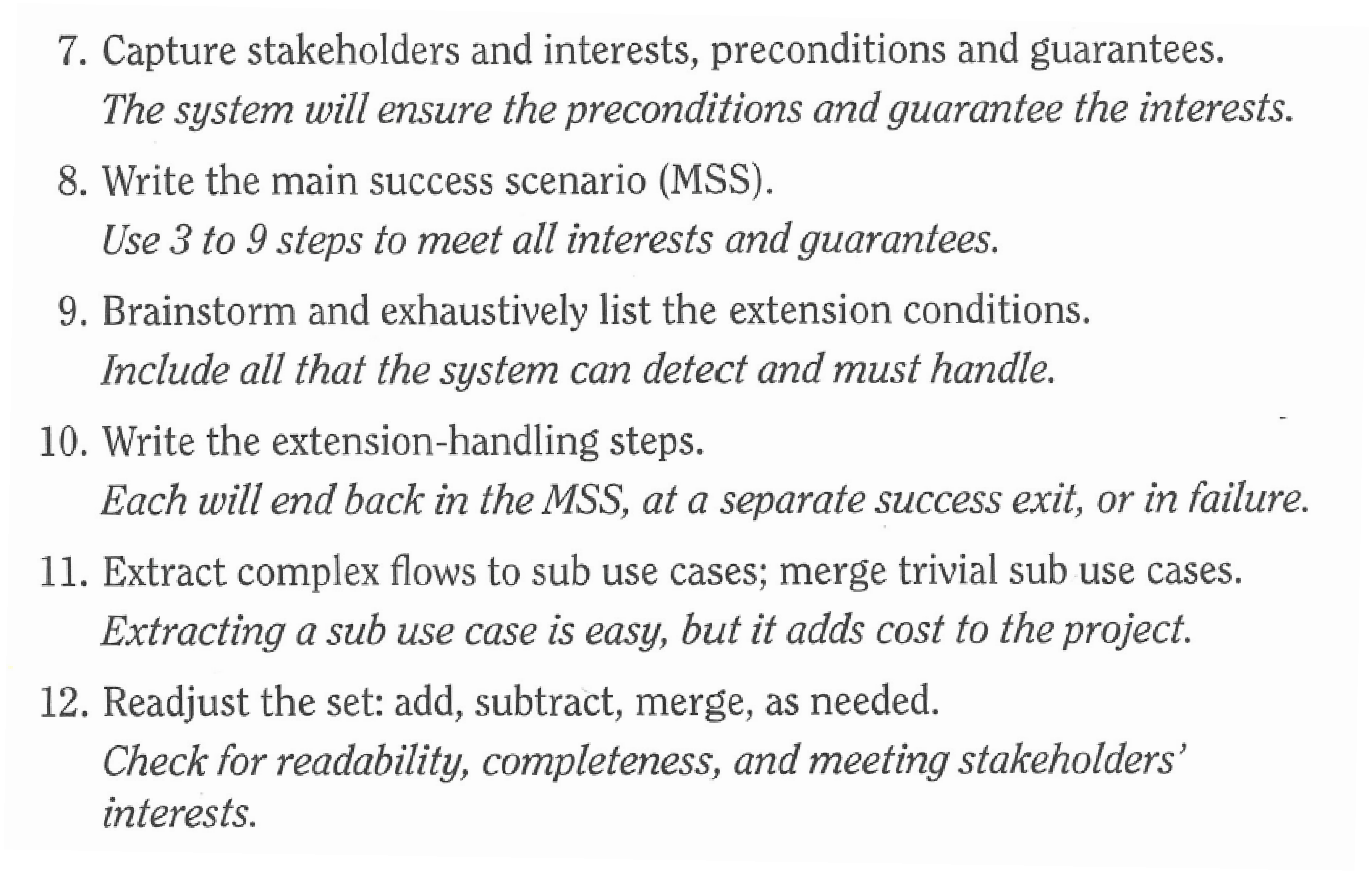
   Check for an outermost use case for each primary actor.

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.

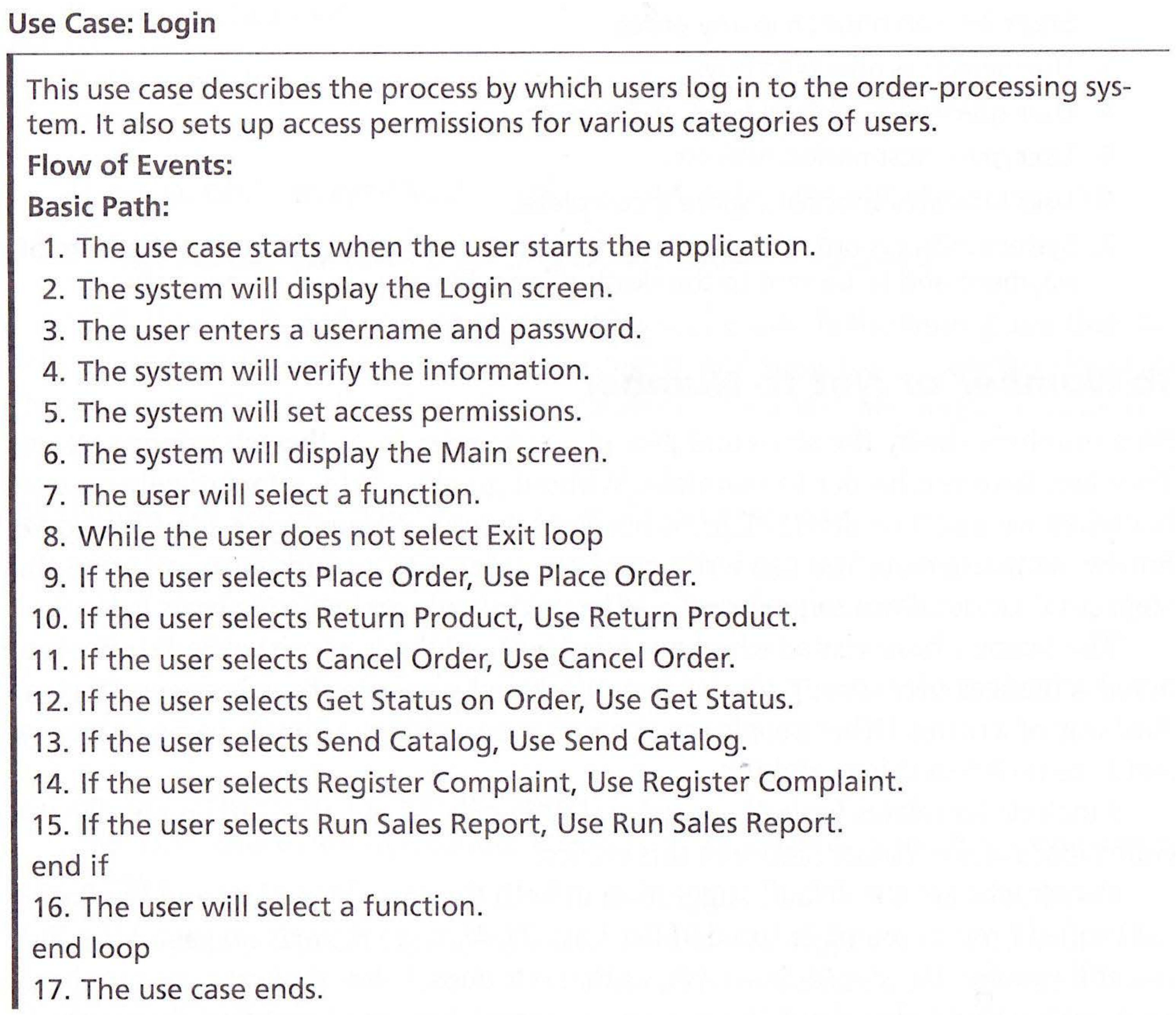


7. Capture stakeholders and interests, preconditions and guarantees.
   The system will ensure the preconditions and guarantee the interests.
8. Write the main success scenario (MSS).
   Use 3 to 9 steps to meet all interests and guarantees.
9. Brainstorm and exhaustively list the extension conditions.
   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.

Exercise: Login Use Case

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

Give him feedback and corrections.



Use Case: Login

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.

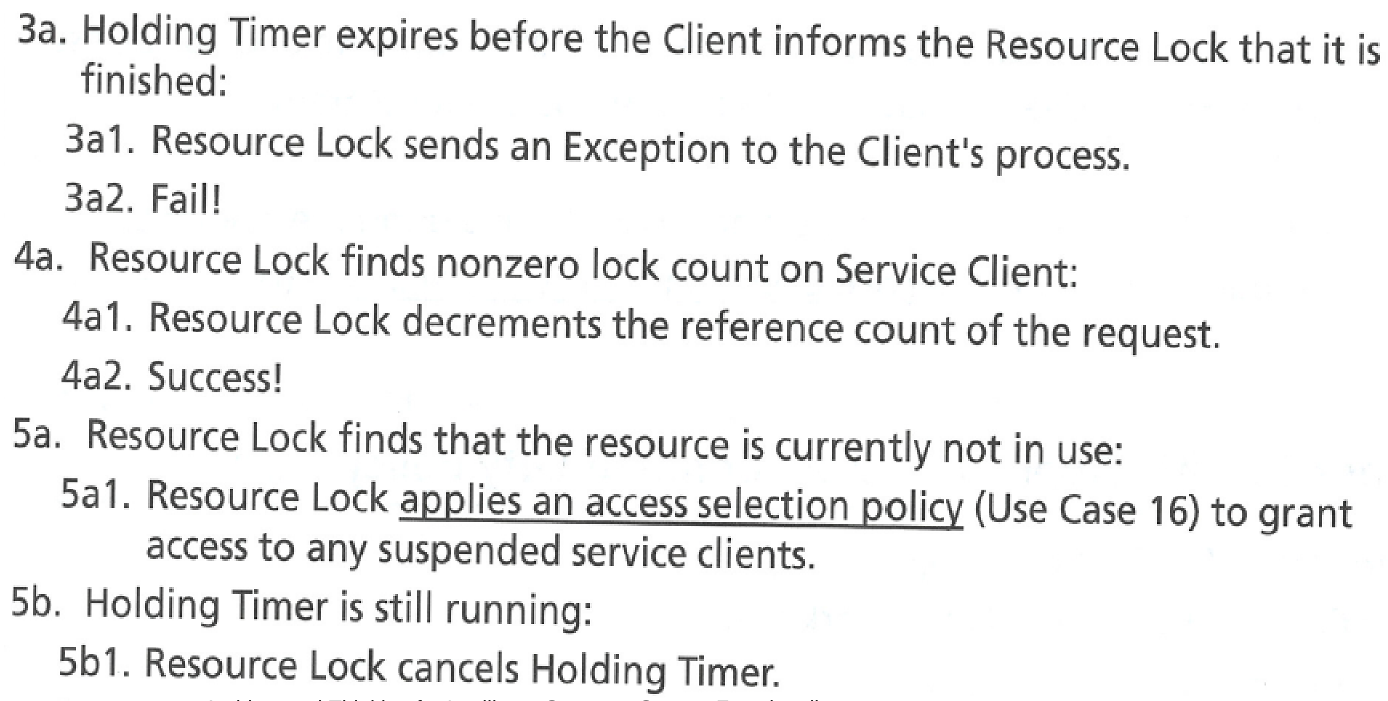
How to Proceed (Cockburn, p. 90 ff.)

- 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!

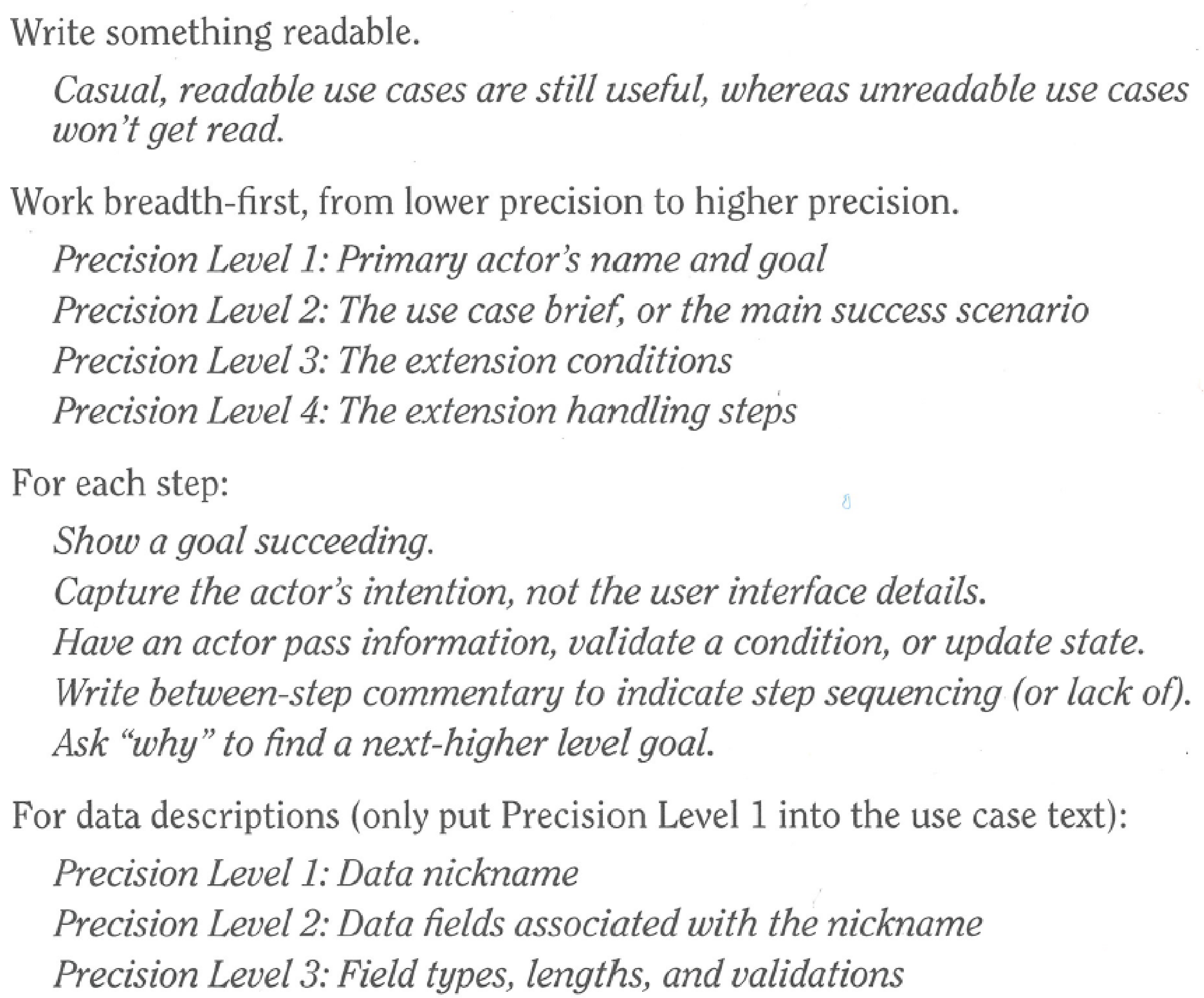
5a. Resource Lock finds that the resource is currently not in use:

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.

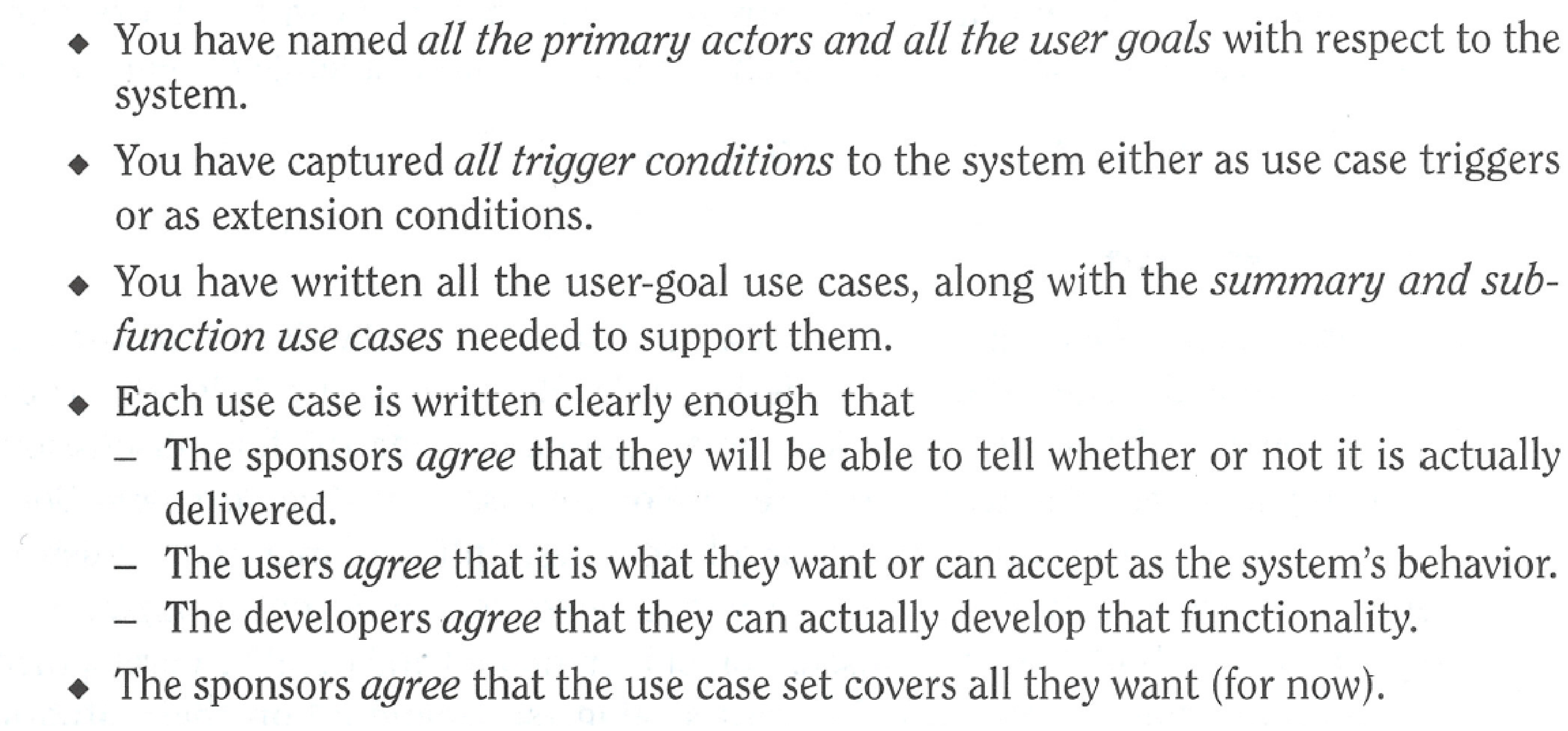
For data descriptions (only put Precision Level 1 into the use case text):

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

UNIVERSITÄT DES SAARLANDES

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?