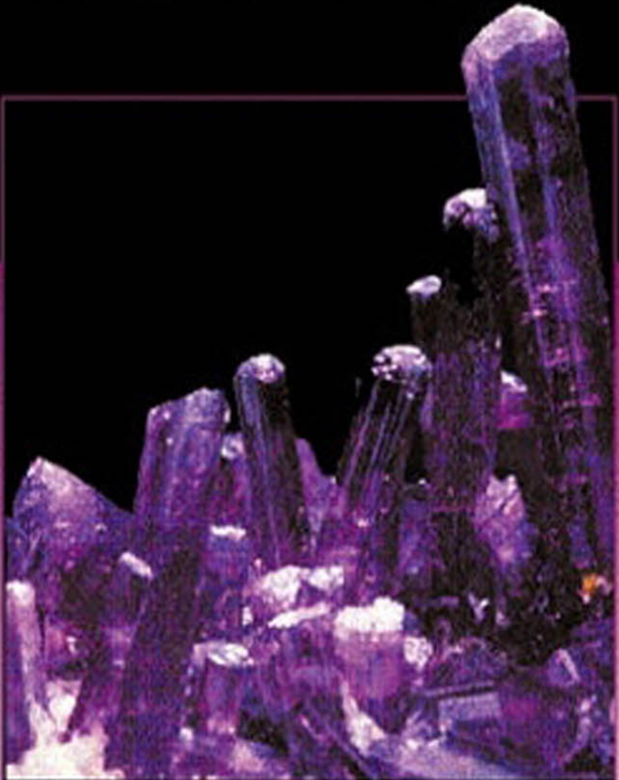
The Crystal Series for Software Developers
Alistair Cockburn, Series Editor

Writing Effective Use Cases



Alistair Cockburn

The Writing Process

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.

- The customer, to the company
- The marketing department, to the software systems combined
- The security department, to the software system itself

These outermost use cases are very useful in holding the work together, and I highly recommend writing them for the reasons given earlier. They will not, however, provide your team with the functional requirements for the system to be built—those reside in the user-goal (blue) use cases.

5.3 SUBFUNCTIONS (INDIGO/BLACK, UNDERWATER /CLAM)

Subfunction-level goals are those required to carry out user goals. Include them only as you have to—they are needed on occasion for readability or because many other goals use them. Examples of subfunction use cases are *Find a Product, Find a Customer,* and *Save as a File*. See, in particular, the unusual indigo use case, Use Case 23, *Find a Whatever (Problem Statement),* on page 78.

Subfunction use cases are underwater, indigo. Some are so far underwater that they sit on the bottom. Those we color black to mean “This is so low level, please don't even expand it into a use case” (“It doesn’t even swim . . . it’s a clam!”). It is handy to have a special name for these ultra-low-level use cases, so that when someone writes one you can indicate that it shouldn’t be written, that its contents ought to be rolled into another use case.

Blue use cases have indigo steps, and indigo use cases have deeper indigo steps, as shown later in Figure 5.2. The figure also shows that to find a higher goal level for your goal phrase, you answer the question, “Why is the actor doing this?” This “how/why” technique is discussed more in Section 5.5.

Note that even the farthest underwater, lowest subfunction use case has a primary actor that is outside the system. I wouldn’t bother to mention this, except that people occasionally talk about subfunctions as though they were somehow internal design discussions or lacking a primary actor. A subfunction use case follows all the rules for use cases. It is probable that a subfunction has the same primary actor as that of the higher-level use case that refers to it.

Summarizing Goal Levels

For now, three points about goal levels are important:

- Put a lot of energy into detecting the sea-level use cases. These are the important ones.
- Write a few outermost use cases to provide context for the others.

- Don't make a big fuss over whether your favorite phrase among the system requirements sentences “makes it” as a use case title.

Making it as a use case title doesn’t mean “most important requirement,” and not making it doesn’t mean “unimportant.” I see people upset because their favorite requirement was *merely* a step in a use case that did not get promoted to a use case that is tracked on its own.

Don’t worry about this. One of the points of the goal model is that it is a relatively small change to move a complex chunk of text into its own use case or to fold a trivial use case back into a higher-level one. Every sentence is written as a goal, and every goal can be unfolded into its own use case. We cannot tell by looking at the writing which sentences have been unfolded and which have not (except by following the links). This is good, since it preserves the integrity of the writing across minor changes. The only goals that are guaranteed to have their own use cases are the blue ones.

5.4 USING GRAPHICAL ICONS TO HIGHLIGHT GOAL LEVELS

In the Using Graphic Icons to Highlight the Design Scope subsection on page 40, I showed some icons that are usefully put to the left of the use case title. Because goal levels are at least as confusing as titles, I put a goal-level icon at the top right of the title. This is in addition to filling the fields in the template. My experience is that readers (and writers) can use all the help they can get in knowing the level.

In keeping with the altitude nomenclature, I separate five altitudes. You will use only the middle three in most situations.

- Very summary (very white) use cases get a cloud, . Use this on that rarest of occasions when you see that the steps in the use case are themselves white goals. If you cannot add icons, add a plus sign (+) to the end of the use case name, as in Use Case 18.
- Summary (white) use cases get a kite, . This is for most summary use cases, whose steps are blue goals. Again, add a “+” to the use case name if you cannot use the icon.
- User-goal (blue, sea-level) use cases get waves, . Add no suffix or an exclamation mark (!) to the name if you cannot use icons.
- Subfunction (indigo) use cases get a fish, . Use this for most indigo use cases. Add a minus sign (–) if you cannot use icons.
- Some subfunctions (black) should never be written. Use a clam, , to mark a use case that needs to be merged with its calling use case.

With these icons, you can mark the design scope and goal level even on UML use case diagrams as soon as the tool vendors support them. You can use the suffixes right away. If your template already contains Design Scope and Goal Level fields, you may use them as redundant markers. If your template does not contain those fields, add them.

5.5 FINDING THE RIGHT GOAL LEVEL

Finding the right goal level is the single hardest thing about use cases. Focus on these guidelines:

- Find the user's goal.
- Use 3 to 10 steps per use case.

Finding the User's Goal

In all of the goal levels, only one stands out from the others:

> You are describing a system, whether a business or a computer. You care about someone using the system. That person wants something from your system *now*. After getting it, she can go on and do something else. What is it she wants from your system now?

That level has many names. In business process modeling, it is called an *elementary business process*. In French, it is the system's *raison d'être*. In use cases, it is the *user's goal*.

Ask the question "Is this what the primary actor really wants from the system now?" For most first drafts of use cases, the answer is "No." Most beginners draft underwater use cases, thinking they are at sea level. To find the higher-level goal, ask either of these two questions:

- What does the primary actor really want?
- Why is this actor doing this?

The answer might be the actor's real goal, but ask the question again, until you are sure. The interesting thing is that even though the tests for a user goal are subjective, people soon come to consensus on the matter. Experienced people have surprisingly similar answers for user goals. It seems to be a stable concept.

Raising and Lowering Goal Levels

The steps in a use case describe how a process gets done; the name of the use case indicates why the process is of interest. You can draw on this how-why relationship when searching for the appropriate goal level to use in a step (see Figure 5.2).

To raise the goal level of one or several steps, ask "Why is the actor doing this?" The answer will be a goal one level higher.

A way to judge goal levels is to look at the use case length. Most well-written use cases have 3 to 8 steps. I have never seen one longer than 11 steps that didn't get better when shortened. I doubt there is anything magical about those numbers, but if I were to guess, I would say that people do not tolerate or think in terms of processes that take more than 10 intermediate steps. I keep waiting for a legitimate counter-example just to prove that the numbers have no deep significance.

Whatever the reason, use this observation to improve your writing. If you have more than 10 steps, you probably included user interface details or wrote action steps at too low a level.

- Remove the user interface details. Show the actor's intent, not his movement.
- Raise the goal level by asking the "why" question to find the next higher goal level.
- Merge steps.
- Compare your use cases with the writing samples in Section 5.6 and in Chapter 19, Mistakes Fixed.

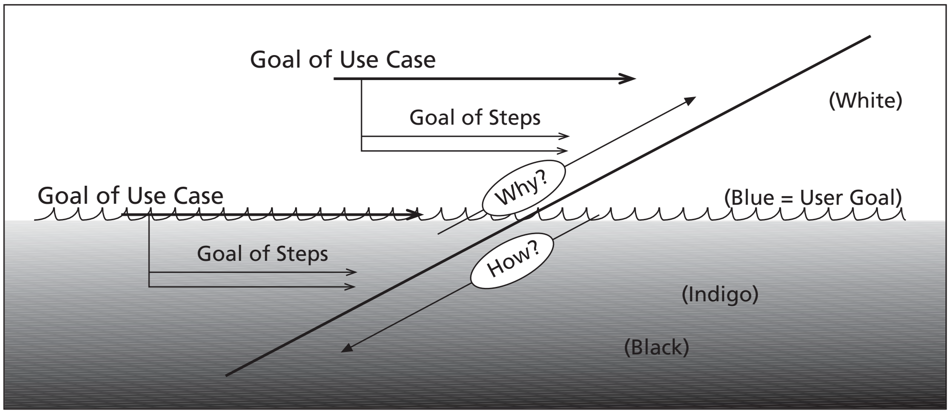


Figure 5.2 *Ask "why" to shift levels*

5.6 A LONGER WRITING SAMPLE: "HANDLE A CLAIM" AT SEVERAL LEVELS

I would like to thank the people at Fireman's Fund Insurance Corporation in Novato, California, for allowing me to include Use Cases 19 through 23 as writing samples.* They were written by claims-handling professionals directly from the field, working with business analysts from the IT department and the technical development staff. The field staff had insights about the use of the system that the IT staff could not have guessed, and the IT staff helped the field staff make the writing precise. Between them, they combined field, corporate, and technical viewpoints.

The writing team included Kerry Bear, Eileen Curran, Brent Hupp, Paula Ivey, Susan Passini, Pamela Pratt, Steve Sampson, Jill Schicktanz, Nancy Jewell, Trisha Magdaleno, Marc Greenberg, Nicole Lazar, Dawn Coppolo, and Eric Evans. They demonstrate that usage experts with no software background can work with IT staff in writing requirements.

I include five use cases to illustrate the things we have discussed so far, particularly design scopes and goal levels. These use cases also illustrate good writing style for steps and extensions. I provide a commentary before each use case, indicating some points of interest or contention.

The set starts with a cloud-level, white-box business use case that shows the business processes involved in handling a claim. Watch how the goals go into lower levels and how the system scope shrinks from "company operations" to "all computer systems" to just "the system under design." The underlined phrases are references to other use cases. The template was modified a little so that the main success scenario is closer to the top and faster to read.

COMMENTARY ON USE CASE 19. The SuD is the operations of the company. Note that the computer system is not even mentioned. The use case will be used by the business to anchor its business procedures and to search for a way to use the computer to facilitate its operations. At the moment, this use case is only in its first stage of sketching. As usual, the main success scenario looks trivial, as it should. It shows how things work in the best success situation! The interesting bits will show up in the failure conditions and in how the company uses this information to make improvements to its IT support of operations. Note the stakeholders.

* Copyright © 1999 by the Fireman's Fund, Novato, CA. Used with permission.

Use Case 19 Handle a Claim (Business)

Scope: Insurance company operations
Level: Business summary
Release: Future
Status: Draft
Revision: Current
Context of Use: Claims Adjuster handles claim.
Preconditions: A loss has occurred.
Trigger: A claim is reported to insurance company.
Main Success Scenario:

1. A reporting party who is aware of the event registers a loss to insurance company.
2. Clerk receives and assigns the claim to a claims adjuster.
3. The assigned Claims Adjuster
   conducts an investigation
   evaluates damages
   sets reserves
   negotiates the claim
   resolves the claim and closes it

Extensions:
To be written.
Success Guarantee: Claim is resolved and closed.
Minimal Guarantee: None.
Stakeholders and Interests:
Insurance company Divisions who sell insurance company policies
Insurance company Customers who have purchased policies
Department of Insurance, which sets market conduct
Claimants who have a loss as a result of act of an insured
Insurance company Claims Division
Future Customers

COMMENTARY ON USE CASE 20. Here is another business use case, in which the SUD is still the operations of the company. However, the goal is at a lower level than in Use Case 19. It shows an adjuster's work that may take days, weeks, or months. It is a kite-level summary use case because it contains many single-sitting activities.

The writer does not mention the computer directly, but only names the goals of the adjuster. The team must take a leap of imagination to invent what in this process the computer can help with. This use case is the raw material for their act of invention.

Step 7 was added because of the interests of the Department of Insurance.