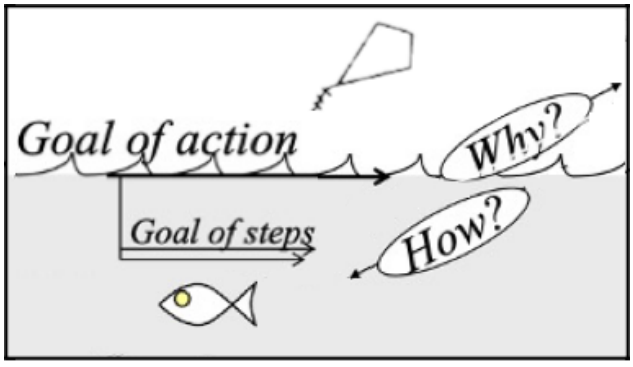
Unifying User Stories, Use Cases and Story Maps

The Power of Verbs: *2nd edition*



Alistair Cockburn

The Simplifying Series

ISBN xxxx -> zzzREVISE needs new isbn
Humans and Technology Press
32 W 200 S #504
Salt Lake City, UT 84101
v0.96c 251024-1200

Changes from the Preview Edition:

- Larger page size, different base font
- High-res images
- Include use case white paper written jointly w Ivar Jacobson
- New use case chapters, section 1 overview, Sample answers to exercises
- Replaced the user story mind-map with a simple list
- Moved summary tables all to the front
- Sync with the *Mini-Book on Use Cases*
- New appendix B with sample use cases

.

Table of Contents

Organization of the book

This is designed to be an instruction book and also a practice book. With that in mind, here are the major parts in the book:

Part 0: Overview of the whole story
Part 1: Key concepts for all of them
Part 2: About user stories
Part 3: About use cases
Part 4: Relating user stories and use cases
Part 5: About story maps
Part 6: Playing them together

You can't make sense of the book if you don't master the key concepts for all of them. After that, you can pretty much pick up any chapter you like, dipping into the other chapters as you need.

In the individual chapters, I provide the key idea for each, a core discussion, additional "fine print" for the interested reader, and some drills.

Part 0: Overview of the whole story

This Part 0 is for people wanting a quick sketch of what's coming up, and for those who already know a lot and just want the punchline.

0.1. "What the heck is a user story, use case, etc., anyway?" is a one pager with just that in it.

0.2. "Just tell me how they fit together, already" walks you through an example.

0.3. "What you're probably doing wrong and how to fix it" is for the more advanced reader, the person who wants some meat right away, in a condensed format. These are all the key sentences from the book with a small table on the key idea, what mistake you are probably making and how to fix it.

0.1. What the heck is a user story, use case, etc., anyway?

*A **user story** is a short phrase or sentence that captures a user's wants.*

Anything they might notice counts. It is not intended to be a specification, it is intended to live in a conversation between a user and a developer. They discuss what the short phrase should intend. The developer goes and programs it up, shows it to the user, revises the work, repeats, etc., until the user says, "Yes, that's good." A user story should be small enough to fit into an iteration.

*An **epic** is a fat user story.*

A user story is supposed to be small enough that the development team can finish it in a single iteration. Clearly, not all user requests fit into just one iteration. The term **epic** indicates that this item is big and will need to be broken down.

*A **use case** is a special writing format to describe the interactions needed for a user to achieve a goal they might have.*

Unlike a user story, it is written with full sentences, failure conditions, how those failures are patched up (or not), and what happens at the end. It is a full spec of the behavior of the system with respect to that user goal.

*A **story map** is a 2-dimensional grid of epics and user stories.*

Every user role gets its own column. The top row(s) of the grid show the user tasks needed to complete a business process. Each column, below that, contains all the user stories involved in delivering the higher-up cards.

0.2. Just tell me how they fit together, already.

Here's the punchline of the entire book:

> *#1: Put the business needs into use cases to align everyone on what will be built and to look for interesting corner cases.*
>
> *#2: Move elements of the use cases to a story map to sequence and track the system's evolution.*
>
> *#3: Cut user stories out of the story map or the use cases to track the individual pieces of work being done.*

And here is the justification:

User stories are tiny things that tag bits of work being done. They are very useful for knowing where each item is during development. However, they don't show the big picture, and it is nearly impossible to find the corner cases early.

Use cases allow you to communicate easily to all parts of your organization what you intend to build, and lets you find obscure corner cases early. They don't easily let you see how each iteration or release differs from its predecessor, nor do they show where all the pieces are in development.

Story maps sketch the big picture at the top level and break down how the iterations evolve from each other in fine detail. They help find some of the corner cases. However, the two-dimensional wall of little rectangles does not communicate easily across an organization. Use cases do that better.

All three require collaboration between the business experts and the developers, user stories and story maps the most. Without that collaboration, they all fail, so don't try.

It is possible to blend them to get the advantages of each in varying circumstances.

A simple example. Buy a Mars bar from a candy machine.

A use case is just how you might instruct a child:

> "Go to the machine and push the button under the Mars bar. It will light up the price on the side. Put in the money. It will drop the candy and the change. Bring those back to me.
>
> If it is out of Mars bars, it will say "Empty". In this case, choose a Milky Way. If that is also empty, then never mind, don't buy anything and give me back the money.
>
> If it is out of change, it will say, "Exact change only." That's okay, I want the candy anyway, so put in the money. It will drop the candy and not give any change. Just bring me the candy."

That is already a (casual style) use case. The more formal, "fully-dressed" style has the steps numbered, like this:

> **Use Case 1**: Buy a candy
> *System under Discussion:* The candy machine
> *Primary Actor:* The customer
> *Goal Level:* Sea-level
> **Main success scenario**:
> 1. Customer selects candy, machine shows the price, and allows money to be put in.
> 2. Customer puts in the money.
> 3. Machine drops the candy and the change.
>
> **Extensions**:
> 1a. No more of that candy:
> Machine shows "Empty" and doesn't accept money.
> 1b. No change available in the machine:
> Machine puts up "Exact change required".
> 3a. No change available in the machine:
> Machine drops the candy but no change.

We will want to design, develop, test and deploy that system in stages. So let's break that use case into user stories to be implemented in a fine-grained incremental manner:

User Stories
1: Show price for a selected item.
2: Open money entry slot when an item is selected.
3: Accept and register amount of money entered.
4: Calculate change.
5: Drop a selected amount of money.
(…and many more)

For any real system, you will have several thousand of these. It is, quite frankly, difficult to make sense of them all.

To make sense of them, let's make a story map, which will also show the evolution of the system.

User role: Customer

Figure 1, Chapter 0.2. Story map for "Buy a candy."

In most systems there will be several user roles, each gets their own sections of the wall. In this system, we have only one, the *Customer*. We lable the top row with the activities to support. This is the "backbone." Under each activity we put the user stories that make up that activity. In this example, the first three rows show the first usable release, the final two of rows the second.

I like this combination:

- Keep the "business" in the use cases, as they show clearly what will get built, with the connections between the various scenarios. It is easy to share around the company.
- Create work assignments and show what will get delivered in each release with a story map. Use it to balance the stories in each release. It works best on a large wall. The online version is not as compelling, but can be shared with others using online picture tools.
- Track the tiny development steps that need to be taken with the individual user stories.

In Part 6 of the book, I show three different combinations:

- Use cases to story maps to user stories (as I showed here).
- Story map first, then find and fill holes in it by writing some use cases and putting those stories back on the story map.
- Use use cases the entire time. Break the steps into user stories directly (not using story maps).