

Ten things to do with your use cases when you're bored

Hüseyin Angay

Karabash Ltd.

Member of the Appropriate Process Movement

<http://www.aptprocess.com>

Abstract

There are times when your use cases look anaemically short. This may cause all sorts of problems to the project and to your reputation. This paper explains some techniques to bulk up your use cases without appearing to do so in order to keep your reputation as a thorough use case writer intact.

Copyright Note

This document resides online at www.aptprocess.com and has been authored by Hüseyin Angay of Karabash Ltd. and of the Appropriate Process Movement. It may be copied freely in part or in whole, with the restriction that anywhere using a copy of more than three paragraphs must include as reference the web address of its origin, as given above.

Sensible Note

In case you've had a long day and all this appears to be genuine advice from a professional engineer, please get some sleep and read this piece in the morning, when you will feel better and less gullible.

This is meant to be an ironic piece. If you still insist that it can be mistaken for genuine advice and decide to go to court over it, I have my defence: "Dean Swift, your honour." (You know, that infamous piece about eating babies, which apparently set a legal precedent.)

We would love to hear from you if you have more tips or if you would like to correct any point...

... unless you have some eloquent argument about why we really should leave stakeholders out of use cases or something similar, in which case, you should get in touch with the team working on the Emperor XVII process framework. They really could do with your help.

10 things to do when you're bored with your use case model

Use cases are generally fairly straightforward to model and to write as long as you follow a number of simple principles. I won't even bother repeating references here, as everyone has a favourite, although the successful results tend to be similar enough. The simplicity of both the principles and of the resulting use cases has a number of drawbacks, of course. Here are some typical reactions to a successful use case model:

"Surely our business cannot be that straightforward. It took me thirty years to get where I am, you know." *I hope you won't be telling me that I wasted two thirds of my life!*

"And you did all this in just a month? But we paid you the equivalent of a year's salary for this piece of work." *We thought you'd take a couple of years over it. We wanted to can the project, anyway, but the MD's got his heart set on it.*

"And it took you a whole month to do this? But there are only two pages!" *We were expecting a doorstop to drop on the client's desk. How am I supposed to intimidate them with this stuff when it won't even raise one spec of dust when it lands on the desk? They'll never pay a million quid for us to build this!*

"The last BA we had took a year for a similar system and had fifty use cases of twenty pages each." *And he was a good laugh in the pub, too. Don't mess with my friends, ok?*

"But we apply the Emperor XVI process here, so you are missing quite a few chapters..." *It's more than my job's worth to question the process. I've got only a couple of years before I retire early on full-pension, you know. How likely do you think I'm to risk messing that up?*

There is also the issue of boredom. You could deflect much of the above criticism by spending a lot longer on the production of your use case model. But, you've been given two months to write your complex use case and you finished it in two weeks. What do you do then?

There is plenty to amuse yourself with at work. You could sit around and bang the pipes loudly with your spanner every now and then – sorry; forget this one; it was meant to be the advice to would-be

cowboy plumbers. You could sit around, listen to office gossip and occasionally pipe up with some incomprehensible comment about more esoteric uses of UML, which might go a long way towards establishing you as the office modelling guru (but don't overdo it, or you'll be ostracised as a sad geek). But what about your use cases? Do you publish them as they are and face the wrath of someone or other or do you *add value*? Here are some tried and tested techniques to add value to your use case models. These techniques have the advantage of not requiring a lot of thinking, so that you can still sit around, listen to the office gossip and pontificate on esoterica, but manage to look extremely busy at the same time. Let's face it: Nobody likes a colleague who appears to be having an easy day at the office.

1. Bulk it up

There is nothing more annoying than being called to a meeting to review a two page use case. There is probably hardly anything to criticise in the use case and it's highly unlikely that any ambiguity or mistake that may be there will provide a long enough discussion – you correct them, and then what? Also, it is surely insulting to tell people that their business is really this simple: Suddenly, you are questioning the value of their work.

You must bulk your use cases up as a matter of courtesy to colleagues and clients. Show them that you respect their business by giving it the space that it deserves. You don't think all those acres of rain forest have been cut down so that they could go to waste because there is no demand for paper, do you? Do your bit for the economy.

Many of the techniques here are extensions of this principle, so remember that point about respect. You are not a social misfit, now, are you?

2. Use lots of alternative flows

A straightforward main flow with a handful of alternatives is a bane of the discerning use case writer. Where is the fun in that? And even if you don't care about fun, how about your professional pride?

Go through your use case and identify all the points where the use case can branch off, however trivial. Create an alternative flow at that point. Do the same with your alternative flows, until you are several layers deep.

Now, chances are that you weren't aware of the handy techniques in this guide. So you already have a model where you chose the most comprehensive path through the use cases for your main flow, so you

will not have so many alternative flows to start with. Here are some mini-techniques to get round this shortcoming:

- Choose as your main flow the path that will produce the largest number of alternative flows. Even if this turns out to be a barren exercise (i.e. your existing main flow already yields the maximum number of possible alternatives), you will have killed many an hour with the search and looked very busy indeed. But don't forget to state the criterion for your choice of main flow, or someone will try to reinstate a simpler path. Some tried and tested criteria are:

"This is statistically the most likely scenario." (How many people do you know who are good at statistics? And are they likely to admit it in front of the boss? Those working in industries whose business *is* statistics, say investment banking, follow this advice at their peril.)

"This is the sunny day scenario." (The boss will love that one. Finally somebody who talks like a human being, instead of R2D2.)

"This one will dictate our architectural spike." (This will shut up that techie who sits in the corner and pipes up about the wonders of XP every now and then.)

"This is the path that adds most value to the business." (You can never say "value add" too often.)

"This is the highest risk path, so we should address it as a matter of urgency." (You can never say "risk" too often.)

Try a combination such as, "This is the sunny day scenario that will dictate our architectural spike. And we cannot overlook the added value, of course." (Now, this guy knows his technology *and* he is a great communicator, too. Most important of all, he never forgets the business angle. A sure winner.)

If you can combine all five of the above in a genuine meeting and manage not to laugh *and* to keep your job, you scare us. Remember what the man said in some black and white film long ago: "One day, my son, all this will be yours." When it is, just leave us a little corner of it, will you?

- An alternative flow that left as a single line embedded in your main flow is a wasted opportunity. Break it out as an alternative and watch it bloom into many lines. Why should you say, "The operator may amend the route," when you could have had one alternative where the operator leaves the route alone and another where he changes it, resulting in the system asking whether he wants to change it or not, to which he could reply

with a 'yes', 'no' or 'I don't know', and... You get the picture; now, paint one yourself.

3. Create a host of tiny use case fragments

But don't call these *fragments*. People may get the wrong impression and think they are worthless. They are not. They are an essential part of the day to day business of the client and they add significant value by highlighting all the tiny little things your clients may be doing in their daily activities. The more bubbles labelled with recognisable business activities you have on the diagram, the more you will engage the attention of the business. You may lose them through the sheer bulk of the diagrams, but don't you keep walking past that A1 plotter sitting constantly idle in the corner? Maybe it's time to justify that capital investment of yesteryear. Diagrams you can hang on the wall are a team-unifying influence. If the plotter can handle colour, too, you're on to a winner.

4. Reuse as much as you can

Use cases and object technology go hand in hand. Object technology means reuse. So, if your use cases do not make use of reuse, we will not be able to exploit (no, leverage) object technology to its full extent.

To increase the reuse in your model and to pave the way for future reuse, break out as much as you can into use case fragments, business rules and screen descriptions.

Also see Technique 3.

5. Introduce ambiguity

Who can resist a good argument over an ambiguous statement? Place enough of these tidbits in your use case and it will become the sweetheart of the review session. And don't worry but rejoice if the ambiguity is missed during the session because it will come back during development or, manna from heaven, during system testing with amplified repercussions. Neither can you be blamed for something that was missed by all those crowds in the review – it would be an admission of corporate weakness.

The best thing about ambiguous statements is that, by the time anyone decides that it is ambiguous, the reasons for the statement will have likely been lost (see Technique 7 for more help in this area), so just the task of clearing up the ambiguity should be worth a good few hours on its own.

A good ambiguity does not jump out of the page at the reader to make them wonder, "Now what does that mean?" It makes them glow with the satisfaction that they know exactly what it means whilst implying something entirely different to other people. For practical guidance on this subject, try BBC's parliamentary coverage or any company financial statement.

6. Talk about the GUI

It's trivial, it doesn't take too much effort to write down, but it expands every step of a use case into three. It also produces the most wonderful explosion of alternative flows and bulks up those suspiciously short ones. Besides, the user interface will change so often that just the task of keeping the use cases and the GUI in synch should keep you busy for months to come.

7. Repeat the bulk of the "shall" requirements in your use case

Use cases typically refer to some "shall"-style requirements, because these are often the starting point for requirements analysis. The lazy amongst us will mention a set of requirement numbers within the use case and leave it at that. Do not fall into this trap. You really don't want to task your readers by making them jump between documents. Save them the trouble by repeating the bulk of the requirements text within your use case. The lazy do this by copying and pasting whole paragraphs from the requirements documents. This does not add value and it makes you look lazy. It might even raise the ugly question of why you have gone to such ridiculous lengths just to repeat some text already available elsewhere. Instead, add value by shaping the text in subtle ways that are obviously relevant to the use case. After you've done this onerous but obviously essential contextualisation, nobody'll dare question what happens when the source requirement changes – anyway, you've already established unquestionably your reuse credentials by using Technique 4, haven't you?

8. Autonumber everything, then use the numbers as labels

This is probably the most important technique here. It will definitely stop your hands from going idle for a long time. And you can blame it on Microsoft – remember, nobody's ever been sacked for buying and then criticising Microsoft. Besides, everybody does it this way, so who can blame you for following the established practices?

Here is the technique: Use the autonumbering feature of your word processor for the use case steps. "But doesn't that actually save time?" I hear you say. Fear not – that's the beauty of the technique:

Everybody knows that it *obviously* saves time. Yet, as soon as you start referring to these step numbers from elsewhere (say, alternative flows or other use cases), your working day is taken care of. When you add or remove a step from your use case, the word processor will obligingly change the numbers of all the steps in the main flow after that point; it will also obligingly leave all the references intact. You now have the delightful task of going round your use case (or round your model, if you played your cards right and referred to the step numbers in including and extending use cases) and updating all the references to those steps. If you become a dab hand in this maintenance task and start to get bored, up the ante by making several changes before correcting the out-of-date references. This provides a challenge no skilled use case writer should miss.

This technique has the added advantage of providing the most delightful distractions in review meetings. Make sure that you leave one or two out-of-date step numbers in your references, which will also cast doubt on the integrity of all the other references. The ensuing verification session may tie up a dozen colleagues for a whole afternoon.

If anyone questions the wisdom of referring to step numbers that change automatically, ask them how to turn autonumbering off in the word processor. Chances are that they won't know. Discredit the rare individual who jumps this hurdle: Ask them how else they propose to number the use case steps. Nobody who spent all their time learning obscure little options in a word processor (no doubt several versions old, thanks to corporate IT policies) will have had the time to work out an answer to that.

9. Make optimal use of available development tools

There are many CASE tools and associated development environments out there, all competing to improve your productivity. But remember: There is a right way and a wrong way of using productivity tools. If you use them purely as drawing tools, you are missing the point and, more to the point, you are not leveraging the full potential of your company's no doubt significant investment in development tools.

Make the CASE tools your friend by using all their features and filling in every single nook and cranny of your model with information. Don't question why, for instance, you need to have a document attached to every diagram; you will obviously need it some day, so create it in anticipation.

Most tools have auto-population facilities for most of the items you can put into your model, so use this facility – it takes no time at all to create model elements that can easily run into pages.

Find the weaknesses of the tools that you are using, and complain often. This will establish you as a guru and will prepare the ground work for any extra work you may need to do in order to keep your model up-to-date.

Make full use of the document generation facilities offered by your CASE tools. It is a trivial task to start off the report generation because most of them come with pre-populated templates. These templates have the added advantage of covering just about everything someone may choose to document. If you've populated your model fully as we mentioned earlier, you will be able to produce reports running to thousands of pages and tie your machine for several hours since most companies provide their workers with state of the art computers that choke on the first word of Computer Aided Stuff, Etc.

Most CASE tools offer facilities for maintaining references between model elements. This may well pose a risk to the use case writer who relies too much on cross-references. Many CASE tools will allow you to attach requirements to your use cases, for instance. The bad news is that use case support in CASE tools is so appalling that they still cannot help use case writers maintain the text or the cross-references in text efficiently. We will be stuck with inefficient support for a long time to come, so we will need to add value by manual maintenance of most of our models. The good news is that we may prolong this state of affairs by ensuring that use case styles differ significantly between enterprises and project teams. Yes, inconsistency in the adopted use case styles makes it difficult for CASE tool vendors to build support for them, but surely a best-of-breed approach works best with use cases, and it is just a shame if this means that we all have to do them differently.

10. Use longer sentences and use more of them

Nothing infuriates the reader more than short, clipped sentences in use cases. They give the impression that the writer, feeling that they had better things to do with their lives, hurried through the use case. Only one thing should concern any writer: The satisfaction of the reader. The readers will be satisfied only when they know that the writer is convinced of the importance of their business. Nothing conveys importance better than elaborate sentences. And if there are many of them, the level of importance is copiously established.

For guidance on this technique, read any good book on technical writing, then do the opposite. Alternatively, buy one of the doorstop paperback novels available in train stations and airports and learn from the masters of the technique.

11. Leave stakeholders out

Resist the temptation to list stakeholders in your use cases. They take too much room and lead to too many awkward questions like, "Are we satisfying the needs of these people/entities?" The answer always tends to be a short yes or no, which really is not desirable if you want to add value by exploring the grey areas in depth. In short, stakeholders are nothing more than a distraction from our main aim of documenting the business in detail.

If you take this stance early enough, it will also help establish your place in the team as the *pragmatic minimalist*, so nobody can accuse you of bloating your use cases afterwards.

12. Never miss a chance for repetition

If you work with use case templates (and who doesn't?), you have the perfect opportunity to drive home some important points about things like preconditions. Most systems have very fundamental axioms that no developer should ignore. These facts should be repeated as often as possible lest they are forgotten. Imagine the embarrassment of delivering a system where users can do things even though they have not logged in, for instance. The solution is simple: Every use case should state in its preconditions that the user should be logged in. This may cause some people to complain that it's mere repetition. Foil their attempt at sabotage: Don't just say, "User must be logged in," the way an amateur would do; use the name of the actor and say, "The Document Enterer must be logged in." This helps you establish the context of the precondition and to drive home the point even more forcefully. This aggressive value add will please everyone except the permanent malcontents who really should know better not to disturb team dynamics with their constant whinging.

Finally

Yes, this turned out to be twelve things to do, instead of ten. So, let's establish Technique 13: All projects overrun. If you're prepared for overruns, you can make a lifelong career by just identifying the projects that will and jumping on them. If you are not prepared for this fact, be prepared to become a failure.