Quality and the Project Management Triangle

Introduction

The basic idea of the project management triangle is illustrating the constraints between scope (features), time, costs (resources), and quality. There are different representations of this model. A prominent one shows quality as the area surrounded by a triangle defined by scope, time, and costs:

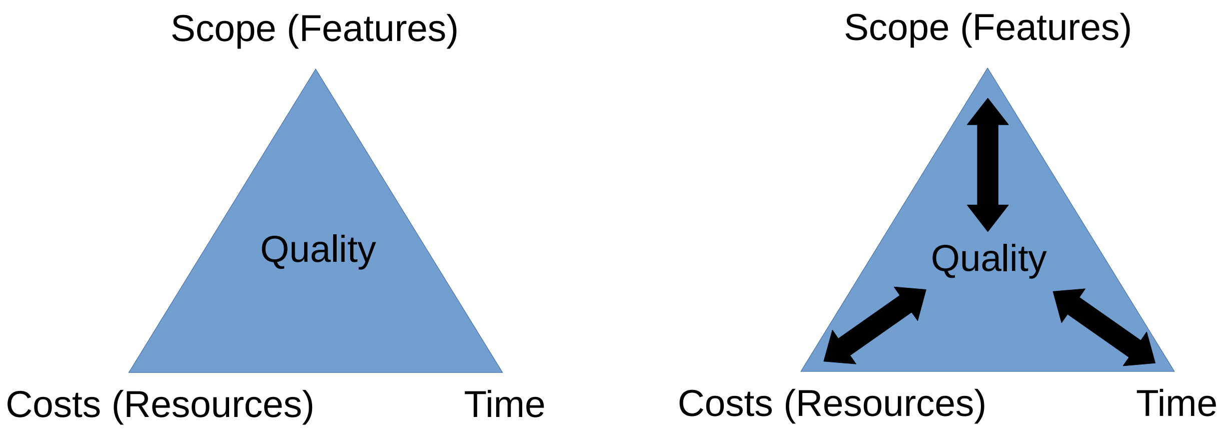

Fig. 1: The project management triangle (left-hand side) and with arrows highlighting the constraints between the elements (right-hand side).

The diagram shows that each change of scope (feature set), time (schedule), and costs (resources) has an impact on quality. However, it suggests quality is a pure implicit parameter. I don't think that's the right way to look at quality. Though, most developers usually have an intrinsic demand for quality, some may even try to create some piece of art and continue their work when the product is already almost perfect. Also strategic leaders will reassure their developers to do so. In contrast, project managers have to deal with the costs. And that's the whole point, because it's their job or let me say their profession to focus on costs, quality could be considered less. Of course, we all know that features might not get accepted, bad quality leads to problems and higher costs in the long run, and adding resources at the end of milestones is not effective and so on and so forth. However, when it comes to money, it's unfortunately most often about immediate costs and not long-term savings. That's why quality does not just happen, you have to care for it.

What about agile?

So better make quality an explicit parameter within the *triangle* and put some spotlight on it.

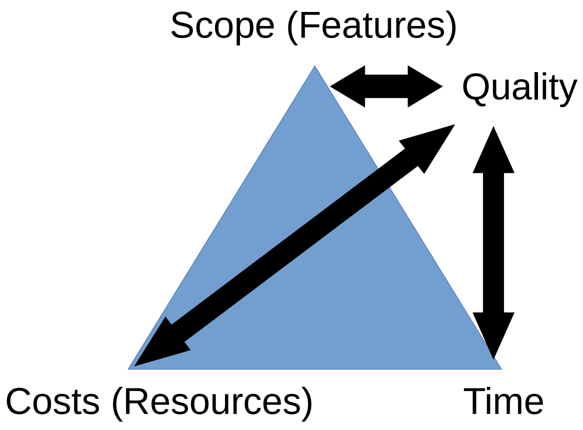

Fig. 2: Pushing quality out of the triangle and making it an explicit parameter.

Agile concepts like XP's test first and test driven development can surely help. The (unit) test code becomes part of the user story and a feature can only be closed if all tests are successfully performed. Timeboxing and splitting efforts and costs into sprints/iterations and increments also seem to be reasonable methods to manage the triangle constraints in a good manner. But is this really the whole truth? Is this really how the daily work looks like? I guess I'm somehow automotive availability biased but from my experience the unit tests are not requested, specified, and delivered in a similar way like functional features are defined and groomed. Maybe there is some general statement about code coverage goals and tests are mentioned in the DoD but even if so, the question remains whether the impact on time and costs of such a demand is also reflected appropriately within the calculation of the commissioned work (despite of all the cost pressure when competing for an award). Same with the request for agile processes. A usual (automotive) milestone plan is quite rigid and strict and in the end you have to meet a fixed delivery date before SOP. Consequently, the next question is, if scope and time can really be considered as variable parameters or rather have to be treated as fixed constants. If scope and time are more or less fixed, the rubber bands (constraints) between the *triangle* elements will strongly pull at costs and quality. What about the chances to find somebody who is willing to increase the costs in order to absorb the exterior forces? Usually not really good, I guess, most likely it's more the other way around and you are asked to go with less budget.

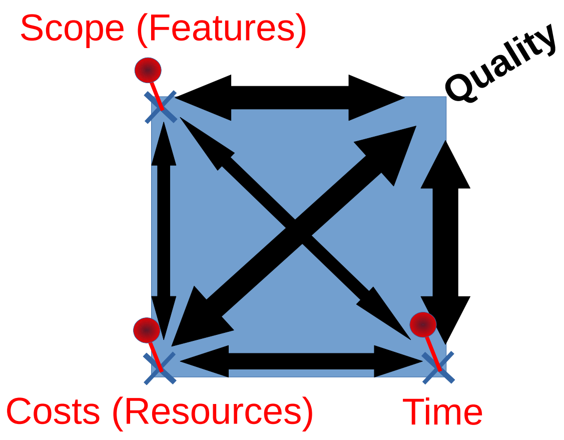

Fig. 3: Another representation of the model, the triangle becomes a rectangle. Unfortunately, with three (mostly) fixed corners.

Grading inflation

So finally there is only one *variable* left. Unfortunately, it's exactly that element which shouldn't be up for discussion at all. That element at which the first diagram looks only implicitly. Of course this effect isn't something that is deliberately or officially accepted. This leads to a dilemma, trilemma or better quadlemma which cannot be easily resolved as long as you aren't able to unfix and tweak the other parameters of the *triangle* which is actually a *rectangle*. However, what also happens instead of untightening scope, time, and costs – at least this is my impression – is the attempt to compensate decreasing quality by grading inflation. Terms like *best* or *excellent* quality are introduced, very good seems not enough anymore. Unfortunately, it's often only an illusion. I don't believe that such relative attributes are suited to reflect the quality of a product. It's like the best pizza in town, where you never really know what you can expect. Quality should be evaluated by absolute grades without dependency too space or time (holds true as long there is no new industrial revolution happening that comes up with some disruptive new production technology). That is to say, if a product is deemed good in Asia, it should also be deemed good in Europe or anywhere else; and if a product reaches today a high grade of quality, a review of this grade shouldn't be necessary just because some new and maybe better products appear on market (a relative rating like *best* or *excellent* would need a reevaluation in that case).

Don't get me wrong, relative grading is not bad per se, e. g. it could be used during the development phase in order to create some internal competition between (sub)components but in the end there has to be an absolute grading target and achievement for the overall product. It's like in school, you are not automatically a good student just because you are the best guy in a class with bad students. However, if you are a very good student, you are still a very good student even if all your class mates are very good. That is to say, very good is good enough and reaching it is not easy – *absolutely* not.

Personally, I have a list in mind with some items I consider to be of very good quality. This list is not increasing much for years now and therefore still easy countable. Anyway, hopefully none of the items will get broken. Of course I'm willing to buy each of these items again, but, unfortunately, chances are high that newer versions of the same product are of less quality if quality is not explicitly taken into account during the value engineering process and cost down project.