Build In Quality

Lawrence H. Putnam and Ware Myers

Quantitative Software Management, Inc.

"Getting people to do better all the worthwhile things they ought to be doing anyway."

That is the language with which Philip B. Crosby began his now famous book, *Quality is Free: The Art of Making Quality Certain.* [i] "People" includes management, he went on to say, but it is up to the professionals in a field to instruct management about this portion of the management job. What can management do to make software quality more certain?

It can provide an adequate amount of schedule time for software development, sufficient staff hours, and the productivity to enhance that time and effort. It can provide a software development process in which the key actions that assure quality are given early emphasis. You can see that providing "adequate" amounts of these ingredients is just another way of saying that quality rests upon the ability to employ the *metrics* for time, effort, size, productivity, and quality judiciously.

Quality is a positive concept

Quality is the positive side of the quality-defect continuum. Obviously, a product doesn't have quality if defects overwhelm it. Beyond the absence of defects, however, quality is the getting into the product of the attributes, not only that users want, but that they actually need. It also restricts these attributes to those users can afford. The marketplace cannot sustain a high price for a *large* number of little-needed attributes. Just how large is "large" is often a matter of judgment by the stakeholders concerned with the product in question.

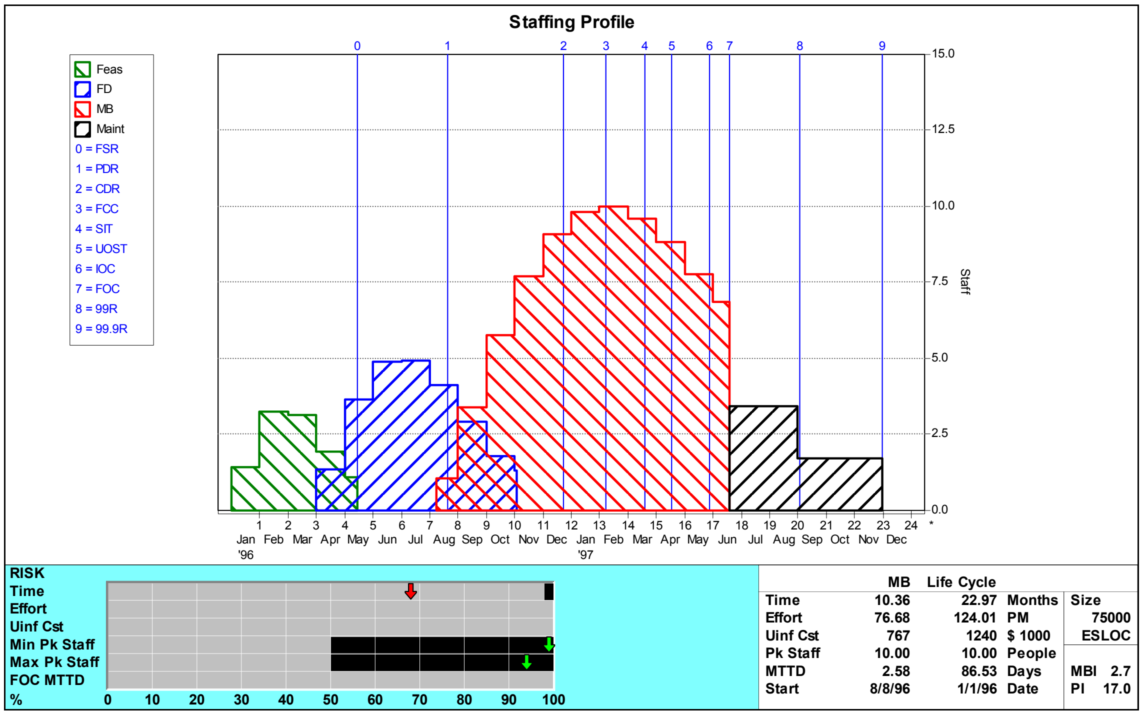
What we do know is that we want to incorporate in our product planning such qualities as expandability, flexibility, integrity, interoperability, maintainability, portability, reusability, resilience, and usability. These qualities cannot, in general, be "inspected in" or "tested in." They must be "designed in," or more specifically, brought in during the early phases of development--requirements capture, analysis, architecting, and design. During this "design in" process they must be weighed against users' needs in an iterative feedback loop.

We know further that it takes time to think through these steps, to implement some of them in iterations, and to get stakeholder concurrence in incorporating the winners in the product. Therefore, it is important to provide this time, and that gets us back to metrics. We need to plan for adequate development time. We need also to plan to employ a development process that allows for these activities.

What we are trying to do is relate the five core metrics to the achievement of positive quality, something beyond the absence of defects.

Identifying quality in the first phase

The achievement of quality begins in the first phase, feasibility study. A division of the life cycle into four phases is shown in Figure 1. The ultimate absence of quality is a project that fails--a system that never gets built. It is in this phase that we establish the boundaries of the system and take the first stab at establishing the requirements within those boundaries. Setting these requirements is analogous to characterizing the quality attributes. So it is important that the main concepts underlying the requirements be established in this first phase. They can be filled out further in the next phase, functional design.



	MB	Life Cycle		
Time	10.36	22.97	Months	Size
Effort	76.68	124.01	PM	75000
Uinf Cst	767	1240	$ 1000	ESLOC
Pk Staff	10.00	10.00	People	
MTTD	2.58	86.53	Days	MBI 2.7
Start	8/8/96	1/1/96	Date	PI 17.0

Figure 1. The software life cycle is typically divided into four phases, shown here as feasibility study, functional design, main-build construction, and maintain and change. Software organizations name these phases differently, but they usually perform comparable functions.

It follows logically that it is important to have a first phase, at least where the project is headed into a new area or new work. Project continuation or project activity in an established field does not require this first phase or, at least, not much of one.

If we need a first phase, it follows further that management allocate enough time to it to establish these main concepts. There are basically two of them:

- Identification of risks of magnitude great enough to bring the successful completion of the system into question. Risks of this magnitude are usually few, if any, but the first phase must explore them to the point at which the stakeholders see that they can be surmounted.
- Categorization of requirements and architecture to the degree necessary to assure the stakeholders that the project falls within time and cost parameters they can afford. These initial parameters may vary by several hundred percent from the eventual "bid."

Metrics plays a role in this first phase in this sense. Some one has to provide some calendar time, some experienced people, some effort in the sense of person-months, and the funding to support this phase. These elements are metrics or depend upon metrics. The funding source has to appreciate the role that metrics plays even in this first phase. Moreover, aside from the funding the first phase takes, it takes time. That time delays getting down to the business of producing Main Build code. The funding source has to be willing to accept this delay. The tradeoff is that the Main Build goes better after a good first phase.

That presents the problem of how much time, effort, and cost to allow the first phase. As soon as we can make a very rough estimate of the size of the proposed system, we can make an equally rough estimate, by employing macro-estimating tools, of the time and effort of the main build. Then the rules of thumb for estimating the feasibility study are:

- Schedule: about one quarter of this main-build schedule.
- Effort: about five to 10 percent of the main-build effort.

These estimates are uncertain to the same extent that the main-build estimates are uncertain at this early point. They are also uncertain because the rules of thumb are rough. In each practical situation they need to be modified by experienced judgment, for example, how much of a feasibility phase does this project need? But they are something to go on! They do undergird the reality that this work--establishing feasibility--needs to be done.

Continuing into the second phase

The identification of quality attributes continues in the second phase, functional design (also called high-level or architectural design). From a quality standpoint, we do three things in this phase.

- We develop the requirements to the extent that it takes to establish the architecture baseline of the system. That means that we further refine the requirements, certainly all the novel ones, and those necessary to establish the main features of the architecture design. You may defer a few detail requirements to the early part of the main-build construction phase.

- We identify the areas in which risks affect estimating and investigate these uncertainties to the extent necessary to prepare the estimate.
- We carry the architecture design to the point of being able to project the project plan and estimate the cost of carrying it out to the precision level of a business bid.

Basically, to get quality, you have to know what you are going to do, you have to know that doing it will not run you into unanticipated risks, you have to have enough information to plan the project, and you have to schedule and staff the plan to the level it takes to get quality.

Again this second phase takes time and effort at a productivity level-- and these are three of the five core metrics. By the beginning of the functional design phase, we have reached a better estimate of the project size--the fourth core metric. (Defects are the fifth.) We estimate the time and effort of this phase with these rules of thumb:

- Schedule: about 30 to 35 percent of the estimated main-build schedule.
- Effort: about 20 percent of the estimated main-build effort.

These estimates are more accurate than the first-phase estimate, because by the end of the first phase we have a better size estimate. Still, the longer you can delay making this estimate, the less the uncertainty of the size estimate and the better the functional design phase estimates will be.

Metrics underlies quality

As a matter of common sense, we observe that it takes time and effort at a productivity level to achieve product functionality at a quality level. Therefore, to achieve quality, we have to provide at a minimum this time and effort. We can do it still better if we can provide this time and effort at a high productivity level. In other words, we have to provide metrics appropriate to achieve the quality goal.

Posters on the wall, saying "Quality First," don't provide this time, effort, and productivity. Managers making occasional gung ho speeches about quality don't provide these metrics. Only metrics can tell you when you are providing the right amount of time, effort, and productivity. It you are not providing them, your posters and speeches will ring hollow, as the daily experience of Dilbert's pointy haired boss has been revealing.

So, spend enough time on the front end

The software process consists of a series of overlapping and iterative phases. Developers can accomplish the tasks of each phase more successfully if management is in a position to provide the time and effort each phase properly takes. In particular, a software organization can estimate the time and effort the main build will take more accurately if the early-phase team can reach a good estimate of the system size. Reaching that good estimate, in turn, takes time and effort, for which management has to make allowance--itself another estimate.

---

[i] Philip B. Crosby, *Quality if Free: The Art of Making Quality Certain,* McGraw-Hill Book Co., New York, 1979, 370 pp.