

Texto\_\_\_\_\_

<http://www.acm.org/crossroads/xrds6-4/software.html> (adaptado)

## The Many Dimensions of the Software Process

By *Sebastián Tyrrell*

### The importance of process

In the past, such processes, no matter how professionally executed, have been highly dependent on the individual developer. This can lead to three key problems.

First, such software is very difficult to maintain. Imagine our software developer has fallen under a bus, and somebody else must take over the partially completed work. Quite possibly there is extensive documentation explaining the state of the work in progress. Maybe there is even a plan, with individual tasks mapped out and those that have been completed neatly marked - or maybe the plan only exists in the developer's head. In any case, a replacement employee will probably end up starting from scratch, because however good the previous work, the replacement has no clue of where to start. The process may be superb, but it is an *ad-hoc* process, not a *defined* process.

Second, it is very difficult to accurately gauge the quality of the finished product according to any independent assessment. If we have two developers each working according to their own processes, defining their own tests along the way, we have no objective method of comparing their work either with each other, or, more important, with a customers' quality criteria.

Third, there is a huge overhead involved as each individual works out their own way of doing things in isolation. To avoid this we must find some way of learning from the experiences of others who have already trodden the same road.

So it is important for each organization to define the process for a project. At its most basic, this means simply to write it down. Writing it down specifies the various items that must be produced and the order in which they should be produced: from plans to requirements to documentation to the finished source code. It says where they should be kept, and how they should be checked, and what to do with them when the project is over. It may not be much of a process. One that says that all products are to be kept in a shoe box under the stairs, be checked by weighing the shoe box to see if it is over a certain minimum weight and to burn the shoe box the minute the check is received is unlikely to win any awards. However, once you have written it down, it is a defined process.

### The purpose of process

What do we want our process to achieve? We can identify certain key goals in this respect.

**Effectiveness.** Not to be confused with efficiency. An effective process must help us produce the right product. It doesn't matter how elegant and well-written the software, nor how quickly we have produced it. If it isn't what the customer wanted, or required, it's no good. The process should therefore help us determine what the customer needs,

produce what the customer needs, and, crucially, verify that what we have produced *is* what the customer needs.

**Maintainability.** However good the programmer, things will still go wrong with the software. Requirements often change between versions. In any case, we may want to reuse elements of the software in other products. None of this is made any easier if, when a problem is discovered, everybody stands around scratching their heads saying "Oh dear, the person that wrote this left the company last week" or worse, "Does anybody know who wrote this code?" One of the goals of a good process is to expose the designers' and programmers' thought processes in such a way that their intention is clear. Then we can quickly and easily find and remedy faults or work out where to make changes.

**Predictability.** Any new product development needs to be planned, and those plans are used as the basis for allocating resources: both time and people. It is important to predict accurately how long it will take to develop the product. That means estimating accurately how long it will take to produce each part of it - including the software. A good process will help us do this. The process helps lay out the steps of development. Furthermore, consistency of process allows us to learn from the designs of other projects.

**Repeatability.** If a process is discovered to work, it should be replicated in future projects. Ad-hoc processes are rarely replicable unless the same team is working on the new project. Even with the same team, it is difficult to keep things exactly the same. A closely related issue, is that of **process re-use**. It is a huge waste and overhead for each project to produce a process from scratch. It is much faster and easier to adapt an existing process.

**Quality.** Quality in this case may be defined as the product's fitness for its purpose. One goal of a defined process is to enable software engineers to ensure a high quality product. The process should provide a clear link between a customer's desires and a developer's product.

**Improvement.** No one would expect their process to reach perfection and need no further improvement itself. Even if we were as good as we could be now, both development environments and requested products are changing so quickly that our processes will always be running to catch up. A goal of our defined process must then be to identify and prototype possibilities for improvement in the process itself.

**Tracking.** A defined process should allow the management, developers, and customer to follow the status of a project. Tracking is the flip side of predictability. It keeps track of how good our predictions are, and hence how to improve them.

These seven process goals are very close relatives of the McCall quality factors ([8]) which categorize and describe the attributes that determine how the quality of the software produced. Does it make sense that the goals we set for our process are similar to the goals we have for our software? Of course! A process is software too, albeit software that is intended to be 'run' on human beings rather than machines!