A Requirements Bigot looks at Configuration Management

Contributed by Carl Singer, PhD
Monday, 04 May 2009
Last Updated Wednesday, 06 May 2009

It's all about the requirements. Say otherwise and I'll turn my back on you seeking someone who isn't either insane or intoxicated. --- Perhaps a bit overstated, but I am a requirements bigot. I have focused on requirements as the source of all things good and the root of all evil since the mid 1960's when I wrote the first PSL / PSA.

So why have I crossed the road to write a brief note in a journal dedicated to Configuration Management? In two words: "Scope Creep." To me, these are two of the dirtiest words in the English language and, by successfully marrying requirements management with configuration management, one can eradicate S---e C----p and its evil sibling, Requirements Creep.

Let's begin with a fairly abstract look at the systems development life cycle. We begin with a business case - a high level (aka, "vague") overview of what the proposed new system should do (Scope) and a ballpark (aka, "vague") estimate of what is should cost to do it (to build and operate this system.) Perhaps we might add in a solution approach consistent with our current architecture.

With appropriate levels of management approval of the proposed project we are underway. A project plan is brewed, more detailed estimates are developed and we head towards the three R's: Risk, Resources and Requirements. Let's focus our discussion only on requirements.

We begin with a scope statement. This statement has a dual purpose - first it's like sourdough starter, it gets the discussion off in the right direction, and second it is meant to filter the requirements as "in scope" or "out of scope." In the traditional waterfall model requirements are gathered with a big bang approach. "Tell me everything the system is supposed to do." "Tell me now or forever hold your peace." Perhaps a bit of sophistication is employed to distinguish between "must haves" and "like to haves" - cost benefit analysis may be employed. The "ilities" - usability, transportability, recoverability .... are also documented. Requirements may be classified as functional or non-functional. In some approaches the user's manual may be developed as an adjunct to the requirements and it's off to design and code. With an iterative lifecycle such as Agile, requirements may be successively elaborated. In all cases requirements are the link between the user and the system.

During the system lifecycle requirements may change due to several factors, including: Environmental and /or business changes, a more sophisticated or tangible understanding of the system, shortcomings in earlier requirements gathering and analysis, feedback from the design and development activities. Regardless of the reason, there may be reasons to consider changing or augmenting the requirements.

OK, enough about requirements, let's talk about configuration management!

Not yet - first consider traceability. Oversimplifying, we must have a link from requirements --> Code (the implemented system) --> Test --> Delivery. Now we can talk about configuration management.

So here's the simple fundamental. Put requirements under the same level of configuration management that benefits code and other systems artifacts. Whether you manage by phases, releases, "effectivities" - however, you parse and manage the delivered and operational system, provide the same level of management control for requirements (and design and code ....) The first thing this implies is that requirements must be documented in a manner that is traceable,

with unique identifiers as well as supporting information (why, when, who changed this requirement, requirements change board approvals, cost estimates, etc.) Configuration management requires that all of these accompany the requirement.

What else? Trace. Assure that every line of code traces back to a requirement. Without this trace - don't build! Similarly, assure that every code change traces back to either a requirement or a test result. Rocket science - not exactly - just a bit of awareness and perhaps some attitude adjustment.

Hopefully, I'm preaching to the choir - telling you things that you already know and believe in. If not, then this message is deceptively important for your project success.

Carl A. Singer

The story is told of a congregant approaching a clergyman after services and commenting on the sermon asks, "Could you please sum up your sermon in a minute or two?" The clergyman replies, "Well, I guess I could." The congregant then rudely replies, "Then why didn't you!" Same with this article: Treat requirements with the same care that you treat code.

Carl Singer has spent most of his professional career dealing with requirements and software development processes. He has a doctorate in Management Information Systems from Purdue University's Krannert School of Management. He has worked for General Electric, Bell Communications Research and IBM. He was part of the team that developed "The Method" at IBM, a means for sharing project-related intellectual capital. Carl has several other credentials including PMP - and has built PMOs and trained and mentored Project Managers. Additionally, Carl served at the highest echelons of the U.S. Army and retired as a Colonel. You can learn more about him and view some of his publications at his website: www.ProcessMakesPerfect.net.