CMCROSSROADS™
A TECHWELL COMMUNITY
Published on *CMCrossroads* (http://www.cmcrossroads.com)

Summary: The "look and feel" of good design is critical for many of today's interactive and web-based systems. Carl Singer writes on configuration management's role in producing a quality product with good design in mind. CM practitioners—including build, release, and deployment engineers—are in a unique position to understand how the entire system is constructed and implemented.

Why Good Design Is Important to Configuration Management Practitioners

By Carl Singer - July 18, 2013

Software development teams generally consist of technology professionals who often specialize in one area or expertise or another. I have worked with many expert database administrators (DBAs), business analysts, software architects, and programmers. Configuration management (CM) specialists often focus on automating an application's build and release, including continuous integration and continuous delivery.

There are times when I wonder how an application was delivered to production (or even test) with so many obvious flaws that any technology professional would question whether or not the application could meet the most basic requirements of *fitness for use*. Obviously, some CM practioners would respond "It's not my job to worry about design—that's design!" I'll accept that answer if it's appropriate to how your development team produces and delivers quality software. Or perhaps you'll want to say more formally "Design review is not included in the CM process." If so, no need to read any further.

On the other hand, you may remember that some time ago you learned about a functional configuration audit. Yes, Virginia, the *functional configuration audit* includes "look and feel," a major point of good design.

Here is how the IEEE Sevocab online dictionary defines the functional configuration audit (FCA):

> (1) audit conducted to verify that the development of a configuration item has been completed satisfactorily, that the item has achieved the performance and functional characteristics specified in the functional or allocated configuration identification, and that its operational and support documents are complete and satisfactory (*IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering*, 2.1) *See Also:* configuration management, physical configuration audit.

Now, what if we think of configuration management as a late in the lifecycle stabilization and quality force in systems design and development? Consider the following requirement (yes, I'm the guy who wrote about requirements way back when):

***R1000.3.5**: The user shall be able to log off of his session from any screen at any time.*

R1000.3.5 is one of those easy to say, hard-to-do functional requirements. It is not uncommon for a design team to come back with a wishy-washy caveat like "*Whenever possible* the user shall be able to logoff his session from any screen at any time." Regardless, let's stick with this requirement as written above.

Next, add in the following global functional requirement.

***R1000**: All screens shall have a consistent look and feel.*

While R10003.5 is clear (and testable.), R1000 is admittedly somewhat subjective. Let's fix this by mixing in a bit of good design.

***R1000.3.5.1**: Every screen shall have in the upper right-hand corner a red octagonal logoff button that securely ends the current user session.*

In an ideal world R1000.3.5.1 would be part of a corporate standard—perhaps a global standard (today my company, tomorrow the world!) But how does configuration management get involved—or should it?

Here's one more twist: not all components of your system are designed and built in-house specifically for this system. Sometimes packages, or what we used to call sub-routines, are imported from vendors, internal libraries, or other in-house systems. Woe unto the designers.

Now consider this question: Should CM consider itself the boy with his finger in the dyke holding back the flood or should CM be proactive?

No doubt you've raised your hand and gone to the head of the class proclaiming that CM should *always* be proactive. And, being pragmatic you've added the auditor's viewpoint and replied "both" to the above question.

Next, I will describe how to go about conducting a functional configuration audit. The functional configuration audit is conducted at three levels. The first is at the requirements level. As functional requirements are added and changed configuration managers must perform a functional audit before building these requirements into the requirements database. This configuration management functional audit ensures that requirements that have been added and changed were approved by the appropriate authorities as not being in conflict with *any* other existing requirements. In our previous example, we narrowed our focus; do any added or changed requirements conflict with R1000, R1000.3.5, and R1000.3.5.1?

The second level focuses on the design, and once again we need to consider whether or not all of the requirements' adds, deletes, and changes are consistent with existing design. Third, and last, we return to the familiar domain of implementation—design, test, and build.

So how does this work in the real world? Configuration management practitioners—including build, release, and deployment engineers—are in a unique position to understand how the entire system is constructed and implemented. We know where all of the configuration items (CIs) are stored in both source and binary form. We understand how to build, package, and deploy the software in what we call CIs. In my opinion, CM experts should be proactive in collaborating with all of the stakeholders on the team, including developers, operations, testing, and information security professionals. We should be willing to point out issues and anomalies including questions about *fitness for use.* Sometimes, this takes the form of collaborating with testers to help them understand the code that we have built and deployed so that they test the application more effectively. Remember *smoketesting* is the last step in any deploy, so testing is in our job description too.

The next time that you build and release an application you should, as always, verify that the correct binaries were built and deployed. This is known as a physical configuration audit. You should also conduct a functional configuration audit, which focuses on verifying that the CIs are performing as they should. So it turns out you are not straying at all when you consider the "look and feel." CM has always had this focus and it is worth it for you to show the true value of configuration management process and procedures! As a CM professional, silence is *not* golden. When you see something, you need to do more than just say something—you need to do something.

About the author

Carl Singer

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. At IBM he was part of the team that developed "The Method", 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. Carl currently volunteers as the Chairman of the Management Board of the IEEE Software and Systems Engineering Standards Committee. You can learn more about him and view some of his publications at his website: www.ProcessMakesPerfect.net.

View Profile