

COTS and Risk: Some Thoughts on How They Connect

David Carney



The focus of the past few columns has been on the various ways in which commercial off-the-shelf (COTS) products affect the way we develop systems: how we define requirements, evaluate products, and relate the activities of product evaluation and system design. This time I'd like to take a little step back, and focus on a different kind of issue that arises when an organization that is building a new system decides to "go COTS." I'd like to discuss a more programmatic topic, one that we at the SEI have had some interesting experiences with, and also describe some of our work in the area.

Risk: The Word of the Hour

The topic at hand is risk: its identification, evaluation, mitigation—all of those related things that we lump under the heading of *risk management*. The thrust of this column will be first to consider how, for a given project, the familiar notions of risk and risk management are affected by the presence of commercial software. Second, I will describe an approach that we have been developing that specifically addresses the question of risk evaluation for COTS-based programs.

Now, the importance of managing risk needs no advocacy here. Within the software community, it has been high on everyone's list for many years. In Barry Boehm's seminal work on "spiral" development, for instance, he presents a model that is inherently risk-driven. The SEI's Software Risk Evaluation (SRE) has been a useful risk management mechanism for dozens of projects. And mandates to do program management through risk management have now made their way into such high-level policy directives as the DoD 5000 series, Clinger-Cohen, Raines' Rules, and other such binding mandates.

And, as the saying goes, we are not alone. The predilection for seeing the world through risk-colored glasses is not unique to the software community; it is equally ubiquitous from a wider perspective as well. A recent national advertisement for General Motors automobiles stresses how "you will be risk-free" if you buy a GM car. Educational building designers focus on creating "risk-free environments" where students can learn. For many years now, every citizen of this land has become aware of the financial and credit agencies that increasingly invade our privacy to determine "how good a risk" each of us might be. And throughout our society as a whole, there is a growing perception that we are trying to nail down all-encompassing safety standards, tamper-proof bottles, child-proof everything, and so forth, presumably to reduce the daily risk of living our lives. (Other writers may wish to comment on whether such a goal is achievable, much less desirable. I am simply noting the societal trend.) With all of this interest in risk, I'm a

little surprised that the manufacturers of the old board game "Risk" have not seen fit to embark on a new advertising campaign. Given the current interest in the word, they could make quite a bundle.

But all of this furor notwithstanding, it is no less true that the central idea—that early in any enterprise we should identify those things that threaten our success, and either avoid them or somehow mitigate them—makes perfect sense in most things, and especially good sense in software development, a very risky enterprise indeed.

COTS: An Up-to-the-Minute Idea

It should be abundantly clear to readers of this column that the widespread use of commercial products in complex software systems poses many novel challenges to both developer and manager. We have, I hope, gained widespread agreement that "doing COTS" means doing some very different things. And there is ample evidence for the following assertion: the simplistic notion that one can easily "add COTS" to a traditional notion of system construction has brought grief to many programs. So as a community, we are now beginning to gain some clarity about the changes—some subtle, some drastic—that we must make when building COTS-based systems. But compared with the "build-from-scratch" paradigm, we are still in a fairly early stage of maturity, and most of us need a good deal more experience in many areas before we will have a firm understanding and broad expertise about "the COTS-based development paradigm" and what it means for system development.

These cautions are no less true when one deals with risk management. If one concurs that the impact of COTS is pervasive across the life cycle (and I hope that you do), and that good risk management entails analyzing and mitigating risks wherever they occur, then it is logical that risk management in COTS-based programs must accommodate the marketplace, whether subtly or drastically, just as everything else must. The changes may be various, though most often will involve COTS either bringing new sources of risk to a program, or exacerbating old forms of risk.

It is not difficult to conjecture some of the specifics. New sources of risk lurk wherever new entities appear. If we now bring many COTS vendors into the equation, then potential risks include the financial health [or "viability"] of those vendors, both now and at some point in the future. If dealing with vendors requires contractual stipulations, then novel risks include contract stipulations and guarantees about product commercialization, cooperation among vendors, licenses, ownership of throw-away prototypes, and many such unfamiliar issues.

Old and familiar risks tend to mutate, just as old and familiar practices do. Thus, we all think we know something about system design. But from a risk perspective, consider how

we now must deal with COTS-based design risks: must we really throw the design out the window if product *X* doesn't include PKI in its next release? What is the fallback? Do we go back to the old design we rejected last month? How do we plan to mitigate such a risk? And so forth, down the long litany of the way things are now, and the way things should be done differently with COTS.

CURE: A Solution for the Moment

The Software Engineering Institute has, over the past few years, been called in to examine a large number of DoD projects that were in varying stages of trouble. Some were in severe "crash-and-burn" mode, others were just beginning to nose toward the ground. But there were two common denominators in most of these troubled programs. One was the strong presence of COTS as a major factor in the systems being built. A second common factor in these programs was that all parties—on both the government and the contractor sides—were generally ignoring the particular risks that stemmed from COTS. Thus, after a number of these "red team" experiences, it became fairly clear that there was a pattern to the questions we were asking, and some common patterns to the behaviors we were witnessing.

We found ourselves repeating the same list of "didn't anyone think of *X*" questions, where *X* would be something like "asking the vendor about his market share," or "asking why the product needed to be modified." We found some common areas where significant danger was waiting for programs—product modification being a major one—and also found that many program managers were simply unfamiliar with these dangers, and thus simply hadn't thought to ask some of these questions.

The result is that we developed a mechanism that would fill this specific need, at least during the present period when there is so little shared experience with using COTS. We distilled the aggregate experience from those "red teams" into a form of COTS-centric risk evaluation, to be performed early in the acquisition process. Some persons have called it a front-end "red team," which is a convenient, though perhaps simplistic, way to think about it.

This mechanism is called a "COTS Usage Risk Evaluation," with the imposing acronym of "CURESM." (More later on the implications of that acronym.) CURE is quite different from the "risk evaluation" as found in a traditional SRE. For one thing, an SRE has the philosophical goal of motivating the personnel of a program to understand the general nature of risk, and to change their behavior, learning to manage it on an ongoing basis. CURE has a different philosophical goal. It is a diagnostic tool aimed at a particular and bounded target, namely the presence of COTS products and the risks that accompany their use.

CURE is intended to be used on any of the participants in a program, whether on the acquiring side or the contracting side. This is a direct result of our "red team" experiences, where both government and contractor exhibited the behaviors that led to the problems. CURE is also designed to be aimed at key personnel: the actual program manager, the contractor's lead architect (or chief engineer, or whatever the job title actually is), and the contractor's project manager. In brief, CURE seeks to find the key decision makers of a project, and to understand their individual awareness of COTS-related issues.

CURE consists of an initial data-gathering phase, an onsite visit, and preparation of a detailed evaluation report. The central instrument for CURE is an extensive questionnaire, used in both the initial data gathering and the onsite visit. The questionnaire covers the following topic areas:

General Topics:

- system description
- management readiness
- technical readiness

Business Topics

- contractual issues
- vendor and supplier relationships

Infrastructure Topics:

- standards
- process
- environments

Lifecycle Topics:

- COTS product evaluation
- system design
- integration
- testing
- maintenance

First, the questionnaire is sent to the site to be filled out and returned to the evaluation team. This provides the evaluation team with an initial view of the program, and also indicates to the program manager, et al., what kind of data the evaluation team is searching for. During the onsite visit, the topics in the questionnaire are revisited in

greater detail. In addition to the raw data that the questionnaire seeks, each question also has a number of optional discussion topics, which are the basis for longer discussions during the onsite visit.

The following excerpt shows the format and content of one of the CURE questions:

11.3 COTS modification

Describe the degree to which the system will require program-specific modification, tailoring, extensions, or enhancements to COTS products.

Identify aspects of the system design that are (or are expected to be) dependent on modifications to specific COTS products.

Potential discussion topics for onsite interview:

Strategy for product replacement if necessary

Decision factors that indicate modification is necessary

How the level of necessary modification was determined

Estimates for the cost and schedule of making these modifications

Plans for sustainment of the modifications

After the onsite interview, the evaluation team prepares a report detailing the major COTS-related risks to the program. Each statement of risk includes the dangerous condition that exists (or will exist), the factors that led the team to this analysis, the resulting consequence, and the overall criticality of that consequence to the program. Perhaps the most important part of the report is that, for each risk, we also suggest one or more mitigations. This report is sent to the evaluation site within 10 working days after the onsite interview.

Results of CURE

As noted above, the acronym is somewhat misleading because this mechanism is not really a "cure" so much as it is a diagnosis. But that slight cheating aside, we have found CURE to be a valuable and powerful tool. In the programs that have had CUREs so far, we have generally found roughly a dozen critical risks, some of which were surprising to the personnel involved, others of which were known but underestimated. Indeed, one

very useful result of CURE, when done on both the government and contractor side, is to unearth misunderstandings about the very same issues from the two sides, a common problem found when rights to products are at stake. ("But I thought *we* had bought all rights to the source code! Didn't the contract define that?")

At the moment, we are revising the questionnaire based on the experiences gained during early trials of the mechanism. We are also in discussion with one large DoD agency concerning transition of CURE for their use. As we perform more evaluations, we hope to gather some firmer data about its efficacy, both in predicting risks to programs as well as in giving them some management assistance in mitigation of those risks. When that data is available, we will report it, either in this space or in an SEI technical report. Stay tuned.

About the Author

David Carney is a member of the technical staff in the Dynamic Systems Program at the SEI. Before coming to the SEI, he was on the staff of the Institute for Defense Analysis in Alexandria, Va., where he worked with the Software Technology for Adaptable, Reliable Systems program and with the NATO Special Working Group on Ada Programming Support Environment. Before that, he was employed at Intermetrics, Inc., where he worked on the Ada Integrated Environment project.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.

® Capability Maturity Model, Capability Maturity Modeling, CERT, CERT Coordination Center, and CMM are registered in the U.S. Patent and Trademark Office.

SM Architecture Tradeoff Analysis Method; ATAM; CMM Integration; CMMI; CURE; IDEAL; Interim Profile; OCTAVE; Operationally Critical Threat, Asset, and Vulnerability Evaluation; Personal Software Process; PSP; SCE; Simplex; Team Software Process; and TSP are service marks of Carnegie Mellon University.

TM Simplex is a trademark of Carnegie Mellon University.