

## Who's in Charge Here?

David Carney



In my last column, I discussed the issue of requirements in the context of COTS-based systems, and described some of the changes that a commercial bias imposes on the way that systems are built (see “Requirements in COTS-Based Systems: A Thorny Question Indeed”). One of the major points of the article was that extensive use of COTS products brings with it an unavoidable loss of control over a system’s requirements. In this column I wish to explore that thesis a little more fully.

### The Basic Idea

My essential premise is that we who are involved with government systems must realize, as we shift our posture in software acquisition toward a preference for commercial products, that we necessarily lose a significant amount of control over our software systems.

First off, this sounds dangerous, since a phrase like “losing control” has a certain ominous semantic ring. But before we hit the panic button, it’s important to recall that the government has willingly chosen to become a consumer; it should be no surprise that consumers generally have only partial control over the shape of the things they consume. To be sure, a consumer has some manner of control, in a very coarse-grained way, through his willingness to buy or to reject a given product. For instance, when Detroit puts out a car that few people like, marketplace rejection is certainly a form of control. But beyond that, it’s very doubtful that any given consumer can be said to *control* the type of motor that the automobile is designed to use, or to *control* the automobile’s shape, silhouette, or profile.

This is not altogether bad, since there are things that we really don’t want or need control over. Stretching the automobile analogy a little further, there are certainly some things that we do explicitly care about. We probably care what color the car is, and we often worry (some of us, anyway) about fuel efficiency. In addition, a buyer typically has some laundry list of other desirable features as well. But few of us car buyers really care too much about the torque (presuming we even know what torque is), and probably no one, not even the die-hard specialist, cares about the firing order of the pistons. For these things, we are just as happy that Ford or GM make all of the decisions, and we are perfectly happy to purchase products whose specifications are at least partially out of our control. (Even now I can hear the wails from the auto enthusiasts about torque...)

## **It's All a Matter of Balance**

Aside from whether we wish or don't wish to make every decision about a system, it is nonetheless true that a loss of control is a necessary corollary to the benefits that come from a COTS bias. Switching to a somewhat different metaphor, the loss of control stemming from the use of COTS products is proof that some of the basic laws of physics hold equally for system acquisition. Let's recall two essential laws of physics. For instance, Newton's Third Law:

*For every action there is an equal and opposite reaction.*

And the Law of Conservation of Matter and Energy:

*The total quantity of matter and energy available in the universe is a fixed amount and never any more or less (i.e., adding energy means reducing matter).*

What is common to both of these physical laws is the notion of balance: if something happens on one side of the ledger then something also happens on the other side; nothing in physics is absolutely independent.

I claim that this balancing act shows up just as naturally in system acquisition as well. For instance, there are many significant benefits and savings that come from shifting to a COTS-based acquisition strategy, at least for certain kinds of software. It is immediately obvious that we gain in speed of deployment. Forgetting for the moment the unfortunate delay of bureaucratic overhead, then buying a commercial product means we get it immediately, not after several years' development time. And it is no less beneficial that by using best-of-breed commercial software, we in the government have a fighting chance to avoid the kind of archaic and unmaintainable systems that still surround us on every side. By positioning our information systems to ride industry trends, we maintain the greatest potential for keeping our systems on an ongoing evolutionary path, and for keeping technological currency in this rapidly changing software world.

But these benefits, like those more tangible events that are subject to physical laws, do not occur in a vacuum. Like atomic fission, where the release of energy means a parallel reduction of matter, our gain (the newfound benefits mentioned above) is balanced by the cost; not just in terms of the purchase price, but also manifest in that we relinquish control over both our systems and the commercial components that comprise them. (Note: there's no claim here that using COTS is some sort of "zero-sum game." But there *is* a claim that when using COTS, you have to pay the piper.)

## **A Brief Tangent: Outsourcing**

We need to be more precise in what this "loss of control" really implies. Since I like tangents, let me try wandering away for just a bit, to examine a term that is very often heard in the business world these days: outsourcing.

It is becoming common for an organization to take steps to avoid duplication of effort, particularly with those business functions that are not part of its core competency. Thus, an organization perceives that some ancillary product or service currently produced internally mirrors a product or service also available externally. Through outsourcing, the organization removes the unnecessary and costly duplication. For instance, imagine a large company whose employees travel frequently. That company might at one time have had an internal travel office to perform all travel-related services. Through outsourcing, the company makes use of an external travel agency to perform the same services. The cost of paying the external agency is more than offset by the savings in internal resources.

Note the key phrase here: *perform the same services*. The things done by the company's internal travel bureau and the external travel bureau are essentially the same service. The internal service might have been a little quicker perhaps, or it may have done a little better in finding optimal flight times. But the basic services provided are the same, which is why the company is willing to outsource the service.

Now, it is sometimes heard (by this listener, at least) that the widespread move toward using COTS products is just a form of outsourcing. From my point of view, this is a misunderstanding of what it really means to use COTS ("to the maximum extent practicable," as the mandate says). To be sure, there are certain parallels: both are partially motivated by a desire to cut cost, and both involve going outside an organizational boundary to procure something previously produced internally.

Probably the thing that most people mean when they claim that "using COTS is a form of outsourcing" is that they now purchase software rather than build it themselves in-house. They have "outsourced" the creation of their software infrastructure. The basis of this view is the realization that many—perhaps very many—government systems have few genuine differences from those used in the industrial world. For systems in the domains of payroll, accounting, inventory, and so forth, the reality is that most of the perceived government-specific requirements are illusory, and use of commercial products to support these services is both feasible and warranted.

But there are vital distinctions as well. With few exceptions, the products available in the COTS software marketplace only partially match with government business processes. By making the conscious choice to use commercial software technology, the government commits itself to whatever changed business processes are needed. The changes might be relatively painless (as in the earlier example of a business outsourcing a travel service), but it may well be far more substantial, requiring a governmental agency to make serious revision of its business processes. This becomes rather distant from the notion of "outsourcing." And, whatever the match between the COTS product and the government's business needs, using commercial products means that someone else controls the government's software infrastructure, which returns us to the main point.

## The Effect of Losing Control

Assuming that we're willing to accept all this, what does it really mean in practice? In a nutshell, it would seem that the loss of control is especially obvious in requirements, in functionality, and in schedule.

We discussed the loss of control over requirements last month at length, so I will merely summarize here. My assertion was that requirements at one time sat alone in the driver's seat, and were foundational in determining how the system was built and how it worked. We all used, and rightly believed in, phrases like "get the requirements right." But in a COTS-based paradigm, this is less true: our requirements now have to share the driver's seat, and we may never get them quite right. To the extent that a given system incorporates commercial components, it incorporates pieces whose requirements were established independently of that system. Whether willingly or no, the builders of the system must permit its commercial components (and their implicit requirements) to have a say in the system's overall shape and functioning.

To be sure, "requirements" in the very widest sense still drive the creation of COTS software, since there is some set of technical objectives that govern how software—any software—is created. But those "requirements" are now the collection of features that a vendor believes will appeal to the widest set of potential customers; those requirements are the aggregate requirements of the whole marketplace, not of any one individual. This results in a kind of democracy of requirements: regardless of the feature you might care about, you're just one consumer among many, and quality is secondary to market share. One need go no further than everyday office software to see how profoundly this is true.

So the requirements that drive a commercial product, and hence, the functional workings of the product, are determined by its vendor. But that's not all; it doesn't stop there. The vendor is also the one who makes the ongoing decisions about which features stay and which are removed from future releases. The vendor makes the choices about long-term sustainment, and also makes the critical decision about whether the product continues to exist at all. The vendor has equal control over the product's release schedule, when upgrades are reissued, how often they occur, and what additional long-term license costs will appear. (This should not be a surprise: the vendor's goal is to make money. That's why he's in business, and *any* strategy we adopt that involves commercial software components should always assume and be based on this fundamental truth.

## But What if We *Can't* Afford to Lose Control?

Supposing that we'll accept that the use of COTS leads to loss of control, whether of the things mentioned in the previous paragraph or any others. Isn't it also the case that there are some occasions when this condition is *not* acceptable? Aren't there circumstances when we really can't afford to lose control? And even more difficult: how can we recognize such circumstances?

Well, let's also go back to the Detroit analogy. Suppose we're not talking about automobiles, but about robotic vehicles that do something nifty with nuclear explosives. Do we seriously expect that the DoD should be willing to leave any decisions about how such a machine works (and especially its deep-down internals!) to a designer of commercial automobiles? Even more to the point: should we applaud if the DoD were to buy such vehicles as cheaply as possible?

It's fairly obvious that I've asked these questions rhetorically, and that I'm arguing against a COTS approach for such machines; they would probably not be good candidates for a COTS-based acquisition strategy. Conceivably, perhaps, some sub-components of such a vehicle might use commercial products, but even that kind of decision would be based on a firm sense that the system requirements are still dominant and not negotiable, that the engineering decisions are made by the robotic vehicles' designers, not by anyone else, and that the system is not being built to leverage market trends but to fulfill a very specific and life-critical mission.

As we said at the top, normal consumers don't control Detroit's plans, nor should they expect to. Most consumers tend to accept what the marketplace offers, and derive the benefits that come from marketplace competition. But some consumers aren't normal, either in schedule or in requirements or in functionality. Some people really do need to worry about torque.

## **Last Thoughts**

We sometimes forget that the current drive toward using COTS is a means to an end, not the end itself. The real end is a complex mix of factors. Savings, sure; there's no room in the budget for the unnecessary duplication by the government of comparable capabilities in the industrial world. But there are lots of other factors as well. One is a realization by the government that it already has too many tasks, and overseeing large-scale production of business software is an unwanted burden. Even more, guidelines for the use of COTS should be based on an awareness that the lightning-fast technological revolution we're caught up in is independent of any single force, even a force as big as Uncle Sam. He (and we) are far better off if we watch the trends as closely as possible, admit the reality of the marketplace, and hang on for dear life.

And hang on we must, given the frenzy of the commercial software marketplace. As Bette Davis, in the film *All About Eve*, predicted long ago: "Fasten your seat belts, everybody. It's going to be a bumpy ride!"

Next time, some thoughts about COTS and risk management. Stay tuned.

---

Copyright © 1999 by Carnegie Mellon University

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.

<sup>SM</sup> Architecture Tradeoff Analysis Method, ATAM, CMM Integration, CMMI, IDEAL, Interim Profile, Personal Software Process, PSP, SCE, Simplex, Team Software Process, and TSP are service marks of 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.