



Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software Systems

Federal Aviation Administration
Software Engineering Resource Center

October 2, 2000

TABLE OF CONTENTS

Preface..... 1

Acknowledgements 1

Introduction 2

Application Domains..... 3

Lessons Learned..... 4

 1 Problems with Vendors..... 4

 2 Leverage in Gaining Vendor Cooperation 4

 3 Need for Flexibility in Defining Requirements 4

 4 Importance of Operational Demonstrations 5

 5 Assessment of Specific Attributes 5

 6 Life-Cycle Issues 6

 7 COTS Integrator Experience 7

 8 Need for Technology Watch to Keep up With Vendors 8

 9 Interface to Legacy Systems 8

 10 Impacts of Volatility During Development 8

 11 Vendor Management 8

Summary..... 9

Preface

The Federal Aviation Administration (FAA), most of the Federal Government, and industry are realizing an ever-increasing dependency on commercial off-the-shelf (COTS) products. Office automation products within the FAA have been heavily or exclusively COTS-based for well over a decade. This paper addresses the recent trend of increased COTS use in the National Airspace System (NAS). While the term COTS refers to both hardware and software, the lessons learned reported in this paper are primarily focused on software. It should be noted that this paper is intended to be the first of several papers on the subject and that future publications will include the entire genre of COTS usage. As we grow in the use of COTS products so too will our knowledge and experience of using COTS products.

It is inevitable that systems development will become more dependent upon COTS in the future. Ten years ago software systems with one million lines of code were considered huge. Now ten million lines of code are not unusual. It is expected that the size and complexity of systems will continue to grow. It is not feasible to develop these systems fully from scratch. As software engineering becomes more disciplined and available COTS expands to provide robust and stable functionality, it will become mandatory to include COTS in all major systems. This document addresses experiences the FAA has had in using COTS during this interim period while COTS products are not all robust, do not necessarily work well together, and require significant risk mitigation activities to be successfully used. However, there are significant and growing examples of successful use of COTS.

The Software Engineering Resource Center (SERC) at the William J. Hughes Technical Center is supporting the efforts of Dr. Barry Boehm of the University of Southern California (USC) in developing COCOTS, a life-cycle costing model that specifically will address the use of COTS. This work is predicated upon the earlier success of both the COCOMO and COCOMO II models. COCOTS is an outgrowth of, and works in conjunction with, COCOMO II.

The lessons contained in this report were captured as part of interviews carried out by Christopher Abts of USC and Dr. Betsy Clark of Software Metrics, Inc. to obtain calibration data for COCOTS, and is by no means a total list of COTS risk areas. The SERC appreciates the generosity of the National Airspace System (NAS) project personnel who provided data and who shared both their triumphs and frustrations in developing COTS intensive systems. *The quotes shown in italics* throughout this document are extracted from the interviews with these personnel. Of the twenty projects interviewed, sixteen shared one or more lessons. The projects varied greatly in their success; some proceeded with relatively few problems while several experienced true COTS horror stories. The majority of the problems encountered could have been prevented by either modifying the development approach or by taking appropriate steps before signing contracts with the COTS vendors. The purpose of this document is to help current and future projects avoid the mistakes and implement practices that proved effective. As the data collection activity continues additional lessons will unfold.

Acknowledgements

Contributing editors to this document:

Patrick Lewis, SERC Program Director
Patrick Hyle, SERC COCOTS Project Manager
Marian Parrington, SERC
Dr. Elizabeth Clark, Software Metrics, Inc.
Dr. Barry Boehm, University of Southern California
Christopher Abts, University of Southern California
Robert Manners, Informatica of America, Inc.

Introduction

“COTS is high risk because we are dependent on someone else. Otherwise, we would have to write all the software ourselves. There needs to be a process to help people evaluate [and mitigate] their risks.”

As an increasing number and variety of commercial-off-the-shelf (COTS) and commercially available software (CAS) packages become available, it is important to understand the costs, benefits, and risks entailed in using these components. For purposes of clarity and readability, “COTS” will be used to refer to commercially available software products that are sold or licensed at advertised prices. Furthermore, “COTS”, as used in this paper, will comply with the definition of CAS contained in the FAA’s Acquisition Management System.

In discussing COTS, we need to consider four distinct groups of stakeholders: (1) the acquirer (e.g., the FAA); (2) the application developer or system integrator who is developing a system incorporating COTS components; (3) the COTS vendor who markets and licenses the COTS product; and (4) the end user. The application developer may be a contractor, may be internal to the acquisition organization, or may be a hybrid of the two.

There can be important benefits from the use of COTS, including faster system development time, lower development costs, and continual product improvement, – the cost of which can be shared by many users. The user base may be wide and diverse, increasing the opportunities to surface problems and ultimately leading to a more stable and mature product. However, there are risks as well, most stemming from the fact that there is much that is out of the control of the COTS acquirer, integrator, and user. It is the vendor and not the stakeholders who control the component’s functionality, its performance, and its evolution. As mentioned previously, a COTS product’s life cycle may be shortened by market pressures and economically driven decision-factors beyond the control of the FAA. This is a salient issue, and we must remain mindful of its breadth of impact. The import of these issues is heightened when we are dealing with a mega-corporation as the provider of the COTS product(s). They not only control the previously mentioned functionality, performance and evolution of the COTS product(s), but also the inter-component operability. This is critical when newer versions, updates, and releases of COTS software are involved as components of a fielded FAA system.

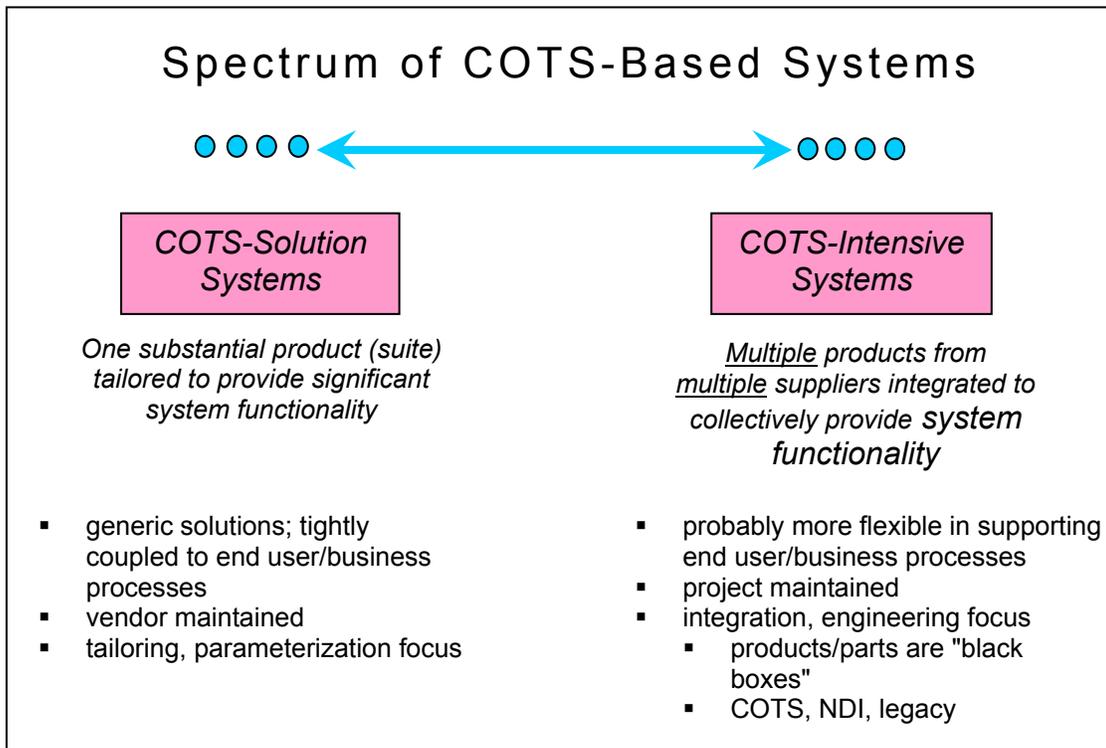
One of the important characteristics of COTS is that the source code is typically unavailable to the COTS integrator. Only the executable code is provided by the vendor. Ownership of the product is not sold per se, rather a license to use the product is granted within established agreement parameters (delimiting product functionality, vendor liability and usage of the product via seat management, etc.). This does not include access to the source code. Without that access, the activities involved in developing and maintaining a COTS-intensive system differ from traditional, custom development. In traditional development, requirements definition is followed by design, code, integration and test. For these systems, effort and cost are driven to a large extent by the number of lines of code to be written or modified. An estimate of system size, usually expressed in terms of the number of lines of code, is a major input into most software cost models. With a system built from COTS components, the effort is not driven by the size of the components but by other factors such as the complexity of the tailoring that must be done in order for the COTS component to work in the local environment. The number of function points drives the complexity of the involved tailoring efforts. These function points can have multiple inputs, interfaces, and interdependencies within the COTS-intensive software system. The function points are typically critical path elements and in traditional systems development comprise a large part of “break point” testing measures. The level of effort (LOE) required to ensure that the COTS products work as advertised can entail significant amounts of tailoring code. This LOE includes the “glue code” needed to ensure a compatible fit and inter–component operation(s). The need for glue code is especially evident when introducing newer COTS components into legacy system environments.

APPLICATION DOMAINS

The sixteen projects reporting lessons learned covered a range of application domains. Twelve were from the FAA. Of these twelve projects, six were air traffic management systems, two handled air-to-ground communications, two were non-operational support systems, and two were administrative systems. Of the remaining four projects, two were from the Air Force and handled missile launches, one was an administrative system from the Army, and one was a mission planning system from the Navy.

For simplicity, the term “COTS-intensive” is used here to describe the class of systems being addressed. In fact, the system may be 100% COTS – this is common for administrative systems – or the system may consist of a mix of COTS and custom code – this is common for operational, safety-critical systems.

There was one notable difference between the administrative and the safety-critical systems in the application level addressed by the COTS components. The COTS components reported for the administrative systems were typically database management systems (DBMSs) or third party components residing on a DBMS. These components embodied business processes in their operation so that choosing a given component had broad implications for ways of doing business. When asked to delineate their COTS components, the administrative projects did not mention operating systems and other infrastructure software. Clearly, these COTS components were part of the overall system but they were not the major focus of the COTS selection or tailoring activity.



In contrast, for the safety-critical systems, much of the application software is still custom developed. For these systems, the focus of the COTS activity is at the infrastructure level. Typical classes of COTS products delineated for these systems included operating systems, device drivers, and network management software. The number of COTS components integrated by a project ranged from one to 150. The lessons reported by the projects are discussed in this document, beginning with those lessons most frequently mentioned.

LESSONS LEARNED

1 PROBLEMS WITH VENDORS

“Our biggest problem was with the vendors, especially with believing their claims. Very few components worked as advertised.”

Six of the sixteen projects (38%) described vendor problems involving misleading claims or promises. The most frequent problem was components that did not work as advertised. One project complained about missed delivery dates. A common frustration was the lack of any real leverage to require vendors to live up to their claims. The consequences ranged from annoying to serious. One of the projects was in litigation with the vendor over an operating system purchased for security features promised by the vendor, which never materialized. After a great deal of effort spent trying to resolve this problem, the only solution was to implement security measures through manual procedures.

One of the more surprising problems occurred for a project that used a COTS product at one site on a pilot basis. The pilot implementation entailed a few hundred copies of the product; the full implementation was worldwide and would entail tens of thousands of copies. Once the decision was made to expand worldwide, the vendor raised the cost per copy. In the words of the project manager, “We assumed that we would get a quantity discount. The price per copy is actually going up. The increase in price is so great that we are seriously considering starting over, this time writing the system ourselves from scratch.” A quantity discount had seemed so obvious that it was not included in the original negotiation.

2 LEVERAGE IN GAINING VENDOR COOPERATION

Vendors are driven by profits, whether current or potential. They can be cooperative and responsive when it is in their perceived interest to be so. One of the projects interviewed was the first government buyer of a product that was previously used only in the commercial sector. The vendor was extremely cooperative in modifying their product for government use because they anticipated a much larger government market contingent upon the successful implementation of their product and were willing to be a partner in making it work.

One person mentioned that any leverage with the vendor occurs before the contract is signed. “That’s when vendors are willing to negotiate. Once you buy a component from a vendor, they are a lot less willing to help out.”

3 NEED FOR FLEXIBILITY IN DEFINING REQUIREMENTS

“Don’t go COTS if you can’t bend your requirements. If you can be flexible, COTS is cheaper. If not, it’s more expensive.”

Five of the sixteen projects (31%) pointed to the need for flexibility in defining requirements, and in particular, the necessity of distinguishing between essential requirements and those that are negotiable. Selection criteria can then be based on essential requirements. Several of the projects described a process that iterated between defining requirements and looking at the capabilities provided by the marketplace. For two of the projects, these iterations included business processes as well.

This can be a lengthy process but one well worth investing in. One of the projects interviewed spent over half of their development time (14 out of 22 months total) iterating between

requirements, business processes, and a market search and pointed to this as one of things they felt they had done right. When end users are included in the selection and iteration, this fosters as understanding of the COTS product as well as user buy-in, both of which are critical for a successful system.

A mistake made by one project was to pay the vendor to make FAA-specific changes to their product. The person interviewed from that project had the following to say: "We made the vendor change his product when we should have changed our process. Since we were unwilling to change our process, new development would have been a better choice."

4 IMPORTANCE OF OPERATIONAL DEMONSTRATIONS

"Operational demos are important. At that point, vendors are bending over backwards to sell their components so they'll participate."

Considering that the number one complaint was that the products did not live up to the vendors claims, the only way to verify what one is really buying is to evaluate the product in the context of an operational demonstration. If multiple components will be used together, they need to be included together.

Once a contract is signed, it can be extremely difficult to force a vendor to make good on their claims. As noted above, one of the projects is in litigation with the vendor but that can hardly be viewed as a desired outcome. Time spent observing and assessing the product in the operational environment in which it will be used was consistently seen as well worth the effort, both by those projects that did this, as well as, by those that did not and later regretted that decision.

Several projects brought users into the operational demonstrations and felt that this was an extremely beneficial activity to ensure user buy-in.

5 ASSESSMENT OF SPECIFIC ATTRIBUTES

"If you have a safety-critical system, you don't want state of the art COTS; you want mature components."

Product maturity was an important attribute in making selections for safety-critical systems. Several people, in particular, pointed to operating system maturity as paramount.

Several of the projects did not assess specific attributes that later proved to be important. Portability was mentioned by two of the projects. Both ended up changing hardware platforms, something that was not anticipated during the original COTS selection.

One project mentioned installation ease as an attribute that they did not assess but later wished they had because the installation turned out to be much more difficult and time consuming than they had anticipated.

Inter-component compatibility and component flexibility were other attributes mentioned. For the latter, the flexibility was with the user interface that did not allow for simpler navigation through the product functions.

6 LIFE-CYCLE ISSUES

Refresh Strategies

“How do we upgrade an operational system without a great deal of disruption?”

One of the major differences in COTS versus traditional systems in the FAA centers upon the frequency and scheduling of system upgrades or refreshes. Heretofore, the strategy has been to upgrade at set, pre-determined intervals. These intervals are included in the initial system project plan. With COTS based or COTS intensive systems, the strategies change due to span of control issue(s). Several of the projects are currently struggling with issues related to refresh strategies. Among the people we interviewed, there do not appear to be lessons learned so much as questions being asked. People do not yet have sufficient experience to know the advantages and drawbacks of different refresh strategies. Among these questions are:

- Do we stay continually abreast of all changes/developments as they appear?
- If possible, do we wait until a "major" refresh is needed (i.e., system operation would be compromised without implementing this change)?
- Do we simply avoid all changes, except for critical Program Trouble Report (PTR) fixes for a specific period of time?

When there are a number of COTS components (we included projects with 120 and 150 different COTS components), the issue of incorporating new versions becomes a major concern. The greater the number of components, the greater the number of version releases, each potentially coming out at different times. The problem of keeping up with these releases is greatly compounded for safety-critical systems that must remain in continual operation. One of our projects faced exactly this situation and the following quote illustrates their dilemma: “Do we freeze the configuration for some period of time and then replace the entire system? Do we incorporate all new versions from all vendors? Do we refresh all components every few years? Do we refresh a selected set of components every few years?” The importance of Operational Demonstration Testing becomes heightened in the deployed COTS world. The need to maintain test bed facilities (or contract to have the vendor do so) to seamlessly incorporate these refreshes is paramount to system success. While one COTS component may remain relatively stable, it is not known if even relatively innocuous changes can be accommodated within the NAS. Risk assessment, risk mitigation, and migration plans all need to be incorporated up front – or at least allowance made for these issues in the initial design and project schedule.

One of the projects – a safety-critical, continuously operating system, made the decision not to upgrade but to freeze the configuration for ten years, after which the entire system will be replaced. With this strategy, the concern shifts to supportability because vendors will stop maintaining a product version after a period of time. This strategy necessitates upgrading to new component version(s), continuing with an unsupported component, or paying the vendor a premium to continue support for that instance of the product. In this case, the project negotiated a one-year extension for vendor support and also purchased the source code, which they will maintain throughout the remainder of the ten years.

One recommendation was to create an operational test environment to use for loading new component versions in order to understand any and all impacts.

Software packages typically have about a two-year life cycle before the vendor no longer supports them. One of the projects pointed to the need to incorporate a refresh cycle during development so that the system delivered to the user is not already at end of life.

Maintenance Costs

“People have to look at the entire life cycle realistically – not just the development cost but consider what it’s going to cost to maintain over a number of years.”

Several people expressed the opinion that COTS saves dollars during system development but may be more expensive than custom-built software over the full life cycle. This is because, as noted above, components must be upgraded every few years or they will no longer be supported by the vendor. In addition to continuing licensing fees, new versions may require additional tailoring, glue code and testing as well as additional product training.

7 COTS INTEGRATOR EXPERIENCE

“Look carefully at the credentials of the COTS integrators. There is a tendency to oversell.”

Several of the people interviewed expressed disappointment with the performance of the COTS integrators stemming from a lack of experience in at least one of the following three areas:

- COTS integration in general
- The specific component(s) being integrated
- Knowledge of government business processes and terminology

Each of these is described in the following sections.

Lack of Experience with COTS Integration

“Our integrator didn’t know what they were doing and used this contract to get smart. They were very unproductive.”

We found several cases in which the integrators were experienced in the application domain but not in integrating COTS components. Some had relatively mature development processes overall but were operating in an immature level for COTS integration. For example, one integrator had been formally assessed as a Software CMM Level 3 but described their COTS integration processes as ad hoc. In evaluating potential COTS integrators, it would be wise to examine their experience in COTS integration and whether they have a defined and repeatable process.

Lack of Experience with Specific COTS Components

The integrators for three of the projects were surprised by the amount of time required to become sufficiently familiar with specific products in order to carry out their integration activities. For example, in one of these cases, the tailoring activity required much more effort than originally estimated due to the complexity of learning a vendor-specific scripting language. In the words of one project manager, “If I were doing this again, I would add 6 months to the project for training the people who had to do the product tailoring.”

Lack of Experience in the Government Domain

The final complaint related to a COTS product that was being tailored for the first time for a government user. A consulting company was brought in that was very experienced in tailoring this product in the commercial domain but this was their first experience tailoring it for government use. A lot of time was needed for them to become familiar with government business processes.

8 NEED FOR TECHNOLOGY WATCH TO KEEP UP WITH VENDORS

"You have to constantly monitor the state of the COTS components. We had a company fold and we were taken by surprise. A technology watch would have prevented us from getting stuck."

The need for a technology watch to track vendors and products was mentioned by two projects, each of which integrated over a hundred software COTS components. As the above quote makes clear, one of these projects was caught by surprise when the vendor of a key component went out of business and left them scrambling to find a substitute.

9 INTERFACE TO LEGACY SYSTEMS

"The Achilles' heel of all COTS projects is the interface to legacy systems. They fail here over and over again. This is the part that is not working well for us."

One of the projects, that in many ways was successful, reported difficulties in interfacing to legacy systems. These difficulties encompassed incompatibilities with data as well as with business processes. The data incompatibilities resulted from different formats and relationships. The project decided not to convert the legacy data, in part, because the users had minimal confidence in the integrity of the data. Their solution was to store the legacy data and make it available for reporting purposes only.

A second type of interface problem occurred with incompatibilities in business rules and timing cycles between the COTS product and the legacy systems. The COTS product was able to make database updates immediately but the legacy system was not which resulted in data inconsistency across systems. Differences in business processes were embodied across systems as well with the result that additional software had to be written to recreate the old business rules in the new COTS-based system just to be able to keep transactions moving.

10 IMPACTS OF VOLATILITY DURING DEVELOPMENT

"You may have to re-tailor COTS components with new releases. In our case, we had to write new scripts to accommodate new features."

In general, we were surprised by the fact that only one project mentioned COTS volatility as an issue at all. That project referred to it in the context of having to write additional tailoring scripts. The lack of COTS volatility among these projects may have been at least in part the result of a deliberate freezing of the product baseline. For example, as noted earlier, one project simply froze their configuration of COTS components so that what was delivered to the field was already nearing its end of life. The recommendation from that project was to build in a refresh cycle prior to delivery.

11 VENDOR MANAGEMENT

"We spent 65 staff months with the vendor. We had never planned on this."

One project mentioned vendor management as an issue. That project was having major problems with a component not performing as advertised. The vendor tried to be accommodating but never could get it working correctly.

Summary

"If you're going to do a COTS effort, upper-level management needs to understand the advantages and disadvantages of COTS and they have to support the effort with resources. They have to understand that they are buying into a different process."

There was a sentiment, expressed by several of the people interviewed, that upper-level management does not understand the risks in moving to COTS solutions. The perception of these people was that upper-management views COTS as low risk. What we have seen in interviewing the projects is that COTS-based solutions do entail risk simply because so much is out of the control of the system integrator.

However, we have also seen actions that projects can take to minimize their risk exposure. These can be summarized as follows:

- Do not rely on vendor claims; verify with operational demonstrations. Time spent on detailed operational demonstrations was consistently viewed as time well spent.
- Bring the users into the operational demonstrations, not just the vendors. The more diverse the user community and the greater the impact on their business processes, the more important this is.
- Establish a technology watch to track vendors and products.
- Be forward looking in assessing attributes. Unanticipated changes in hardware platforms may occur.
- Understand that your leverage occurs before the contract with the vendor is signed.
- Negotiate all prices up front.
- Understand that profits are what motivate vendors. Whether they are cooperative or not depends to a large degree on anticipated profits.
- Distinguish between essential requirements and those that can be negotiated. Successful use of COTS solutions requires the capability to modify requirements.
- Use mature products for safety-critical applications.
- Skill level and experience are important. This includes people on the acquirer's side who are determining essential requirements as well as the COTS integrators.
- Expect to spend time in training. In choosing a system integrator, look not only at experience in the application domain but also with COTS integration in general and with the specific products to be integrated. Do they have a mature COTS integration process?

Consider including a refresh cycle before fielding the system so that the products are not nearing end-of-life. Even with a system development of 24 months, COTS components are likely to be obsolete by one or more versions.

Consider the impact of discrepancies between your COTS-based system and any legacy systems with which your system interfaces. These discrepancies may take the form of inconsistencies in data, in timing of transactions, and in business processes.