

Proceedings
Focused Workshop #7
System Integration
with
Commercial Off the Shelf
(COTS) Software
November 6 - 8, 1996

Center for Software Engineering



UNIVERSITY

OF SOUTHERN

CALIFORNIA

Editors:
Christopher Abts
Barry Boehm

Proceedings
USC Center for Software Engineering
Focused Workshop #7:
System Integration with COTS Software

Table of Contents

Workshop Overview.....	p.1
• Workshop Highlights (Dr. Barry Boehm).....	p.3
• USC-CSE Issue Paper: Workshop on COTS Software Integration (Dr. Barry Boehm & Dr. Walt Scacchi).....	p.19
◇ Topic Background.....	p.19
◇ Workshop Objectives.....	p.20
• Workshop Agenda.....	p.25
• Welcoming Briefing (Dr. Barry Boehm).....	p.29
Workshop Issue Papers.....	p.53
• “Supporting Distributed Configuration Management in Virtual Enterprises” (Noll & Scacchi).....	p.55
• “Use of COTS/NDI in Safety-Critical Systems” (Cochrane).....	p.69
• “Using Commercial Off-the-Shelf (COTS) Software in High Consequence Safety Systems” (Scott, Preckshot & Gallagher).....	p.93
• COTS product integration in view of COCOMO (Buchness).....	p.99
• “COTS Integration in Software Solutions - A Cost Model” (Ellis).....	p.105
• “Architectural Mismatch: Why Reuse is So Hard” (Garlan, Allen & Ockerbloom).....	p.117
Technical Presentations.....	p.127
<i>USC Faculty and Graduate Students</i>	
• Dr. Barry Boehm (COTS Integration Cost Modeling).....	p.129
• Mr. Christopher Abts (COTS Software Research Effort Status Briefing).....	p.147
• Dr. Ahmed Abd-Allah (COTS Integration Interoperability Analysis).....	p.163
• Dr. David Wile (Software Architecture Technology for COTS Integration).....	p.183
• Dr. Walt Scacchi (Process Integration of COTS, Legacy and New Tools Over Wide Area Networks).....	p.213

(continued)

Table of Contents

(continued)

Technical Presentations (continued)

Affiliate Members

- Ms. Gail Cochrane, TRW (COTS/NDI Use in Safety-Critical Systems)..... p.231
- Ms. Agi Sardi, Raytheon Rapid Lab (COTS Risk Mitigation: Strategies and Goals)..... p.257
- Ms. Patricia Oberndorf, SEI (Integrated CASE Environments: Lessons and Predictions)..... p.277
- Dr. Rami Razouk, Aerospace, (COTS Use: Theory and Practice)..... p.291
- Dr. Anthony Wasserman, (A Tool Builder’s View of COTS Integration)..... p.295
- Mr. Marvin Carr, SEI (COTS Integration Risks)..... p.315
- Ms. Karen Kelley, Lockheed Martin (COTS Integration Process Model)..... p.327

Breakout Groups..... p.329

- Architecture Technology Solutions..... p.331
 - ◊ Group Discussion Summary p.333
 - ◊ Final Report..... p.339
- Risk Management Approaches..... p.353
 - ◊ Group Discussion Summary p.355
 - ◊ Final Report..... p.363
- Cost Estimation Approaches..... p.371
 - ◊ Group Discussion Summary p.373
 - ◊ Final Report..... p.393

Conclusion..... p.405

- *USC-CSE Response to Workshop Ideas: COTS Integration Process Approaches: Relations to WinWin and Architecture Projects* (Dr. Barry Boehm)..... p.407
- *Wrap-up* (Dr. Barry Boehm)..... p.427

Appendices..... p.431

- **A** - “COTS Integration: Application of Lessons Learned” (Lockheed Martin)..... p.433
- **B** - “Anchoring the Software Process” (Boehm)..... p.479
- **C** - “Best Current Practices: Software Architecture Validation” (AT&T)..... p.505
- **D** - CSE Preliminary COTS Survey..... p.557
- **E** - Conference Attendees..... p.585

Workshop Overview

Workshop Highlights

Barry Boehm

Introduction

Although the workshop concentrated on COTS software integration risks and ways to improve COTS integration, this was done in a context that COTS usage is generally a good thing. For most applications, deciding not to use COTS is simply unrealistic. But the road to successful COTS integration has many risks and pitfalls. In this context, I'd like to summarize the workshop results in terms of:

- I. Four characteristics of COTS integration which I found helpful in explaining the various pitfalls and recommendations highlighted by the workshop participants (several of the points are adapted from a particularly good Lockheed Martin briefing included in appendix A of these proceedings, "COTS Integration: Application Lessons Learned," which is well worth your study).
- II. The primary research priorities identified by the participants, and USC-CSE's plans for addressing them.
- III. A set of corporate-level COTS integration issues emerging from the workshop which we plan to address at the upcoming USC-CSE Executive Workshop for Affiliates on March 12, 1997.

I. Four Key COTS Integration Characteristics and Their Implications

The four key COTS integration characteristics below make COTS integration significantly different from other forms of software development (including maintenance). They require traditional approaches to software development to be significantly revised. They are:

1. You have no control over a COTS product's functionality or performance.
2. Most COTS products are not designed to interoperate with each other.
3. You have no control over a COTS product's evolution.
4. COTS vendor behavior varies widely.

1. You have no control over a COTS product's functionality or performance.

If you can modify the source code, it's not really COTS--and its future becomes your responsibility. Even as black boxes, big COTS products have formidable complexity: Microsoft people have indicated that Windows 95 has 25,000 entry points.

Resulting Pitfalls

- *Using the waterfall model on a COTS integration project.* With the waterfall model, you specify requirements, and these determine the capabilities. With COTS products, it's the other way around: the capabilities determine the "requirements" or the delivered system features.
- *Using evolutionary development with the assumption that every undesired feature can be changed to fit your needs.* COTS vendors do change features, but they respond to the overall marketplace and not to individual users.
- *Believing that advertised COTS capabilities are real.* COTS vendors may have had the best of intentions when they wrote the marketing literature, but that doesn't help you when the advertised feature isn't there.

Resulting Recommendations

- *Use risk management and risk-driven spiral-type process models.* Assess risks via prototyping, benchmarking, reference checking, and related techniques. Focus each spiral cycle on resolving the most critical risks. The Raytheon "Pathfinder" approach summarized in the Sardi presentation included in these proceedings is a particularly effective way to address these and other risks.
- *Perform the equivalent of a "receiving inspection" upon initial COTS receipt,* to ensure that the COTS product really does what it is expected to do.
- *Keep requirements negotiable until the system's architecture and COTS choices stabilize.*
- *Involve all key stakeholders in critical COTS decisions.* These can include users, customers, developers, testers, maintainers, operators, or others as appropriate.

2. Most COTS products are not designed to interoperate with each other.

The [Garlan et al, 1995] article included in these proceedings provides a good case study and explanation for why interoperability problems can cause COTS integration cost and schedule overruns by factors of four to five.

Resulting Pitfalls

Lack of COTS interoperability exacerbates each of the previously cited pitfalls. Some additional direct pitfalls are:

- *Premature commitment to incompatible combinations of COTS products.* This can happen in many ways: haste, desire to show progress, politics, or uncritical enthusiasm with features or performance. Short-term emphasis on rapid application development is another source of this pitfall.

- *Trying to integrate too many incompatible COTS products.* As the Garlan article shows, four can be too many. In general, trying to integrate more than a half-dozen COTS products from different sources should place this item on your high-risk assessment list.
- *Deferring COTS integration till the end of the development cycle.* This puts your most uncontrollable problem on your critical path as you approach delivery.
- *Committing to a tightly-coupled subset of COTS products with closed, proprietary interfaces.* These restrict your downstream options; once you're committed, it's hard to back yourself out.

Resulting Recommendations

The previously cited recommendations on risk-driven processes and co-evolving your requirements and architecture are also appropriate here. In addition:

- *Use the Life Cycle Architecture milestone as an anchor point for your development process.* It is described in the "Anchoring the Software Process" paper included in appendix B of these proceedings. In particular include demonstrations of COTS interoperability and scalability as risks to be resolved and documented in the Architecture Rationale.
- *Use the Architecture Review Board (ARB) best commercial practice at the Life Cycle Architecture milestone.* It is described in the AT&T/Lucent ARB paper included in appendix C; AT&T has documented at least 10% savings in using it over a period of 8 years.
- *Go for open architectures and COTS substitutability.* In the extremely fast-moving software field, the ability to adapt rapidly to new best-of-breed COTS products is competitively critical.

3. You have no control over a COTS product's evolution.

Again, COTS vendors respond to the overall marketplace and not to individual users. Upgrades are frequently not upward compatible. And old releases become obsolete and unsupported by the vendor. If COTS architectural mismatch doesn't get you initially, COTS architectural drift can easily get you later. Figure 1, from the Lockheed Martin briefing, indicates that current COTS-intensive systems often have higher software maintenance costs than traditional systems, but that good practices can make them lower. (Don't over-interpret the chart; it's not meant to imply that it's cheaper for you to maintain your own home-brew relational DBMS than to pay the COTS maintenance fees for a clean SQL-compliant product.)

Resulting Pitfalls

Lack of evolution controllability exacerbates each of the previously cited pitfalls. Some additional direct pitfalls are:

- *"Snapshot" requirements specs and corresponding point-solution architectures.* These are not good practices for traditional systems; with uncontrollable COTS evolution, the maintenance headaches become even worse.
- *Under-staffing for software maintenance,* and lack of COTS adaptation training for maintenance personnel.
- *Tightly coupled, independently evolving COTS products.* Just two of these will make maintenance difficult; more than two is much worse.
- *Assuming that uncontrollable COTS evolution is just a maintenance problem.* It can attack your development schedules and budgets as well.

Resulting Recommendations

The previously-cited risk-driven and architecture-driven recommendations are also appropriate here. In addition:

- *Stick with dominant commercial standards.* These make COTS product evaluation and substitutability more manageable.
- *Use likely future system and product line needs as well as current needs as COTS selection criteria.* These can include portability, scalability, distributed processing, user interface media, and various kinds of functionality growth.
- *Use flexible architectures facilitating adaptation to change.* These can include message/event-based, software bus, encapsulation, and layering.
- *Carefully evaluate COTS vendors' track records with respect to predictability of product evolution.*
- *Establish a pro-active system release strategy, synchronizing COTS upgrades with system releases.*

4. COTS vendor behavior varies widely.

Vendor behavior varies widely with respect to support, cooperation, and predictability. Sometimes a COTS vendor is not even the developer, just a value-added reseller. Given the three major sources of COTS integration difficulty above, an accurate assessment of a COTS vendor's ability and willingness to help out with the difficulties is tremendously important. The workshop identified a few assessment heuristics, such as the experience that the value of a COTS vendor's support follows a convex curve with respect to the vendor's size and maturity (see Figure 2). Small vendors often lack the capability to support you; very large vendors have the capability, but not the motivation (the classic example is Microsoft).

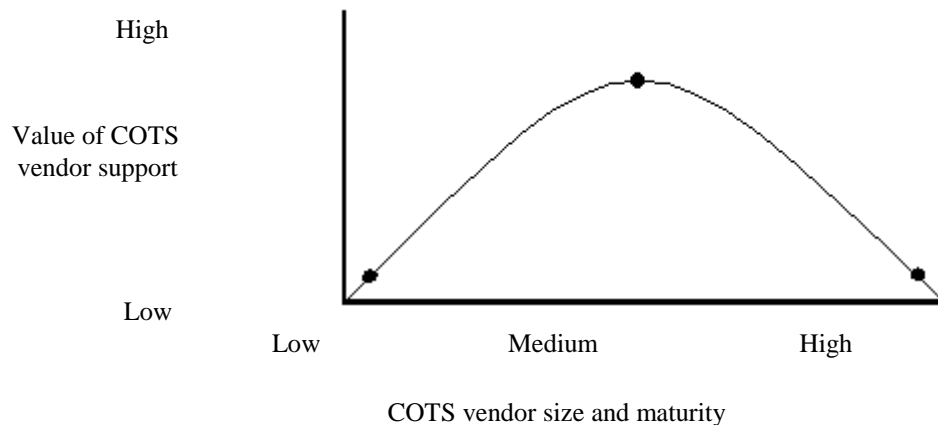


Figure 2 - Variation of COTS Vendor Support with Size

Resulting Pitfalls

Poor COTS vendor support exacerbates each of the previously-cited pitfalls. Some additional direct pitfalls are:

- *Uncritically accepting COTS vendors' statements about product capabilities and support.*
- *Lack of fallbacks or contingency plans, for such contingencies as product substitution or escrow of a failed vendor's product.*
- *Assuming that an initial vendor support honeymoon will last forever.*

Resulting Recommendations

The previously cited recommendations are also appropriate here. In addition:

- *Perform extensive evaluation and reference-checking of a COTS vendor's advertised capabilities and support track record.*
- *Establish strategic partnerships or other incentives for COTS vendors to provide support.* Incentives can include financial incentives, early experimentation with and adoption of new COTS vendor capabilities, sponsored COTS product extensions or technology upgrades.
- *Negotiate and document critical vendor support agreements.* Establish a "no surprises" relationship with vendors.

Summary of Four Key COTS Integration Characteristics and Their Implications

The main results of the general workshop presentations and the Risk Management breakout group can be summarized in terms of the four key COTS integration characteristics and their associated pitfalls to avoid and recommended practices to adopt. For easy reference, these are highlighted in Table 1.

Pitfalls to Avoid	Recommended Practices to Adopt
<i>1. You have no control over a COTS product's functionality or performance.</i>	
<ul style="list-style-type: none"> • Using the waterfall model on a COTS integration project. • Using evolutionary development with the assumption that every undesired feature can be changed to fit your needs. • Believing that advertised COTS capabilities are real. 	<ul style="list-style-type: none"> • Use risk management and risk-driven spiral-type process models. • Perform the equivalent of a “receiving inspection” upon initial COTS receipt. • Keep requirements negotiable until the system’s architecture and COTS choices stabilize. • Involve all key stakeholders in critical COTS decisions.
<i>2. Most COTS products are not assigned to interoperate with each other.</i>	
<ul style="list-style-type: none"> • Premature commitment to incompatible combinations of COTS products. • Trying to integrate too many incompatible COTS products. • Deferring COTS integration till the end of the development cycle. • Committing to a tightly-coupled subset of COTS products with closed, proprietary interfaces. 	<ul style="list-style-type: none"> • Use the Life Cycle Architecture milestone as a process anchor point. • Use the Architecture Review Board (ARB) best commercial practice at the Life Cycle Architecture milestone. • Go for open architectures and COTS substitutability.
<i>3. You have no control over a COTS product's evolution.</i>	
<ul style="list-style-type: none"> • “Snapshot” requirements specs and corresponding point-solution architectures. • Understaffing for software maintenance, • Tightly coupled, independently evolving COTS products. • Assuming that uncontrollable COTS evolution is just a maintenance problem. 	<ul style="list-style-type: none"> • Stick with dominant commercial standards. • Use likely future system and product line needs as well as current needs as COTS selection criteria. • Use flexible architectures facilitating adaptation to change. • Carefully evaluate COTS vendors’ track records with respect to predictability of product evolution. • Establish a pro-active system release strategy, synchronizing COTS upgrades with system releases.
<i>4. COTS vendor behavior varies widely.</i>	
<ul style="list-style-type: none"> • Uncritically accepting COTS vendors’ statements about product capabilities and support. • Lack of fallbacks or contingency plans. • Assuming that an initial vendor support honeymoon will last forever. 	<ul style="list-style-type: none"> • Perform extensive evaluation and reference-checking of a COTS vendor’s advertised capabilities and support track record. • Establish strategic partnerships or other incentives for COTS vendors to provide support. • Negotiate and document critical vendor support agreements.

Table 1 - Key COTS Integration Characteristics and Their Implication

II. Affiliate Research Priorities and USC-CSE Response

Figure 3 summarizes the Affiliate research priorities from the Architecture breakout group. It assesses the candidate research activities considered by the group in terms of Relative Payoff and Relative Difficulty on (High, Medium, Low) scales. The most attractive research areas were those assessed as High in relative payoff and Medium in relative difficulty:

1. Architecture specification and modeling techniques.
6. Interface specification techniques.
11. Extend Ahmed Abd-Allah's work to look at architectures and styles from multiple views.
12. How to map a COTS component into the system architecture.
13. System and software engineering processes to deal with COTS integration.

The group also identified the areas they believed would be most fruitful for the USC-CSE architecture research activity to focus on:

- Identifying criteria for assessing architectures.
- Extend Ahmed-Abdallah's work to look at architectures and styles from multiple architectural views.
- How to map a COTS component into the system architecture.
- System and software engineering processes to deal with COTS integration:
 - ⇒ Assess implications of architecture first vs. COTS first.
 - ⇒ How to do "architecting."
 - ⇒ How to map system requirements to architectural components.
- Ways of characterizing architectural dependencies existent on each of the four views.

USC-CSE will respond to these priorities by reinforcing those of our existing efforts which are aligned with the priorities, and by establishing a new effort to cover the major high-priority topic we are not addressing: integration of multiple architectural views.

In the area of heterogeneous architectural style composition, Cristina Gacek is extending Ahmed Abd-Allah's AAA tool to cover additional architectural styles, such as repository, blackboard, real-time, and closed-loop feedback control. In the area of architecture-based tradeoff analysis, we are pursuing architectural style-sensitive analysis tools.

Ahmed Abd-Allah has identified approaches for some needed extensions to classical reliability models to cover style-sensitive phenomena (e.g., spawning of multiple process threads).

In the view integration area, we plan to build on our recent experiences in specifying multiple-view guidelines for fifteen student teams developing multimedia applications for the USC Library. The results indicated that a tighter integration of the views was both feasible and desirable. Alexander Egyed, who was Teaching Assistant for the fifteen projects, will be investigating this for his Ph.D. research.

The cost modeling breakout group focused on creating a baseline functional form for the USC-CSE COTS Integration Cost Estimation Model. CSE had developed a straw man functional form based on a literature review, analysis of Affiliate questionnaire responses, and analysis of a number of candidate cost driver factors' influence on several sources of COTS integration costs. The baseline functional form uses the External Interface portion of Function Points for sizing, with effort multipliers to be derived from the "Major Significance" and "Intermediate Significance" variables identified in Tables

2 and 3, which show the relative effects of the cost drivers on the five main sources of COTS integration costs.

III. Corporate-Level COTS Integration Issues

The workshop also identified a number of significant COTS integration issues that are important to understand and address at the executive level. We will be addressing these at the USE-CSE Executive Workshop on COTS Integration at USC on March 12, 1997. The major issues are:

- *Integration of COTS considerations and corporate strategies.* Using COTS software is an attractive strategy for "10X" -type initiatives to reduce software cost and cycle time by factors of 10, but they need to deal with the facts that software maintenance costs frequently increase due to COTS volatility, and that COTS-grade quality and volatility may conflict with concurrent "six sigma" -type initiatives to radically reduce software defect rates.
- *Effects of COTS considerations on corporate software processes.* Traditional software processes involve top-down, requirements-to-capabilities, waterfall-type models. COTS -intensive processes tend more to involve bottom-up, capabilities-to-requirements, spiral-type models, with a strong emphasis on prototyping, risk management, and emerging best practices such as architecture review boards.
- *Corporate strategies for dealing with COTS vendors.* How does one best balance long-term vendor strategic alliances with architecting for COTS substitutability? How do strategies vary in dealing with small, medium, and very large-size COTS vendors?

- *Assessing emerging technology and standards.* How effective are wrappers, mediators, instrumented connectors, and glue-code generators in dealing with COTS incompatibilities? What COTS integration problems are and are not solved with such interface standards as DCE, CORBA, and Active-X? How effective are emerging models for analyzing COTS integration cost, risk, and performance?

The preliminary agenda for addressing these issues at the Executive Workshop begins with an overview session, followed by three Affiliate executive panel sessions on COTS considerations in commercial organizations, in government contracting, and in emerging technology. A subsequent proceedings will summarize the results of the Executive Workshop.

Final Agenda

USC Center for Software Engineering
Focused Workshop #7
System Integration with COTS Software
Nov. 6-8, 1996
Leavey Library Auditorium, USC Campus

Wednesday, Nov. 6

07:30 - 08:15 *Continental Breakfast*
08:15 - 09:00 Barry Boehm, USC COTS Integration Cost Modeling
09:00 - 09:25 Chris Abts, USC COTS Integration Survey Status Report
09:25 - 09:50 Ahmed Abd-Allah, USC COTS Integration Interoperability Analysis
09:50 - 10:20 *Break*
10:20 - 11:00 Dave Wile, USC Software Architecture Technology for COTS Integration
11:00 - 11:30 Walt Scacchi, USC Integrating COTS, Legacy and New Software Applications
11:30 - 12:15 *Lunch*
12:15 - 12:45 Gail Cochrane, TRW COTS for Safety Critical Systems
12:45 - 01:15 Agi Sardi, Raytheon COTS Integration Risk Management
01:15 - 01:45 Patricia Oberndorf, SEI SEI Integrated CASE Environments: Lessons and Predictions
01:45 - 02:15 *Break*
02:15 - 02:45 Tony Wasserman A Tool Builder's View of COTS Integration
02:45 - 03:15 Marvin Carr, SEI COTS and SEI Risk Taxonomy
03:15 - 04:00 Other Affiliate Presentations
04:00 - 04:30 Determine Breakout Groups
04:30 - 05:30 Breakout Group Sessions
05:30 - 07:00 *Reception (Faculty Center)*

Thursday, Nov. 7

07:30 - 08:15 *Continental Breakfast*
08:15 - 12:00 Breakout Group Sessions
12:00 - 01:30 *Lunch & Plenary Breakout Group Progress Reports*
01:30 - TBD Breakout Group Sessions

Friday, Nov. 8

07:30 - 08:00 *Continental Breakfast*
08:00 - 10:30 Breakout Group Presentations & USC Responses
10:00 - 11:30 General Discussion & Wrap-up

Breakout Groups

Three breakout group topic areas are discussed in the USC-CSE Issue paper:

1. Architecture Technology Solutions: Judy Kerner, Aerospace Corp., chair
2. Risk Management Approaches: Walt Scacchi, USC, chair
3. Cost Estimation Approaches: Barry Boehm, USC, chair

Workshop Issue Papers

Technical Presentations

COTS Integration cost Modeling
Dr. Barry Boehm
USC
11/06/96

In this talk Dr. Boehm presented an overview of COTS modeling issues as examined by research being done at USC. He identified five sources of COTS integration effort (assessment, tailoring & tuning, glue code, application volatility, and added application validation & verification effort). Then he discussed the pros and cons of four COTS modeling alternatives within the context of COCOMO (COCOMO 2.0 reuse model, COTS-integration effort multiplier or exponent factor, COTS-integration effort added to other development effort, Stand-alone COTS-driven estimation model). This was followed by a review of the Loral CotsCost model, including cost drivers and calibration results. He concluded with a discussion of the proposed USC COTS integration cost model, including a proposed estimation procedure and candidate cost drivers, and areas for further discussion.

**COTS Software Research
Effort Status Briefing
Christopher Abts
USC
11/06/96**

Mr. Abts presented an overview of the COTS integration cost model research being done at USC, followed by a discussion of the results of a first pass COTS data collection survey. The overview described near-term goals (a prototype cost estimation tool, estimation guidelines), funding sponsorship (Air force Electronics System Center and USC-CSE Affiliates), and the data collection approach (two data survey passes) forming the basis for model development. The second part of the discussion focused on preliminary results of the first round survey, including a prioritization of candidate cost drivers.

COTS Integration Interoperability Analysis

Dr. Ahmed Abd-Allah

USC

11/06/96

Dr. Abd-Allah began with an example of the architecture of a COTS based tool (Aesop) taken from Garlan. Then he presented the architectural styles of representative COTS packages (UNAS, Prolog, SoftBench, OBST, BaseWorX, Delphi). This was followed by an approach to identifying architectural mismatches among COTS products (implicating potential COTS integration difficulties) by modeling *architectural styles*. This included discussion of efforts focused at a uniform representation of architectures in terms of base entities and style spaces, using those representations to indicate the architectural composition of COTS products, which in turn forms a basis for identifying possible architectural mismatches between COTS products. He concluded by discussing the prototype Architect's Automated Assistant (AAA) as a tool for analyzing architectural constraint mismatches, and pointing to future research directions.

Software Architecture Technology for COTS Integration
Dr. David Wile
USC ISI
11/06/96

Dr. Wile began by discussing the importance of “domain specific” architectures. He talked about three levels of architectural environments (domain development, specific application development, application execution), and two architectural levels (reference architecture, application architecture). This was followed by a discussion of various forms of architecture “connectors,” which form the basis for all inter-module interactions. Then he discussed architectural integration paradigms (old style: co-resident—new style: distributed & autonomous). He finally concluded with a discussion of example architectural tools.

Process Integration of COTS, Legacy and New Tools Over Wide Area Networks

Dr. Walt Scacchi

USC

11/06/96

Dr. Scacchi began with a discussion of conventional software environment architecture and the difficulties associated with extrapolating this architecture to the modern distributed “virtual” software enterprise. This was followed by a discussion of the goal of process integration (application of tools to data by an explicit process), the potential problems interfering with attainment of that goal (data access, multiple process models and instances, legacy and COTS tools, process guidance), and the inadequacies of existing standards (CORBA, OLE) for mitigating these problems. This was followed by discussion of the DHT model as a way of overcoming these difficulties. The solution to process integration is found in a Process Modeling Language (PML) and a PML “engine,” an environment for enacting processes.

COTS/NDI Use in Safety Critical Systems
Gail Cochrane
TRW
11/06/96

Ms. Cochrane began with the goal of this study done for the FAA, which was to assess the potential need for changes in the FAA safety certification process due to the impact of COTS/NDI usage in safety critical systems. This was followed by an overview of the approach the study followed (evaluate direction of software engineering, identify motives for using COTS/NDI software, determine how COTS/NDI issues are addressed by other organizations, determine recommendations). Next came a discussion of the focus of the study, particularly with respect to software as the provider of functionality. After an in-depth review of the findings under each area of interest (evaluate direction, identify motives, determine other organization practices), she concluded with—among other recommendations—the following process recommendations:

- Perform risk assessment of proposed system.
- Perform extensive prototyping and testing.
- Safety certification process must begin at the start of the life-cycle.

COTS Risk Mitigation: Strategies and Goals

Agi Sardi

Raytheon

11/06/96

Ms. Sardi began with a brief “history” of the recent practice of COTS integration at Raytheon, moving from the ESC PRISM/CARDS program to the RAPID lab, and a focus on COTS integration issues to the RAPID technology deployment to COTS technology performance. This was followed by an in-depth description of the RAPID process, PRISM and RAPID Domain architecture, and change management model. Then an overview of the RAPID assessment process, which is focused on choosing the “right” COTS. In summary, RAPID is

- an *architecture* based approach
- with an investment in COTS *qualifications*
- and needing comparable *performance measures*

Integrated CASE Environments: Lessons and Predictions
Patricia Oberndorf
SEI
11/06/96

Ms. Oberndorf began with a background overview of the history and role of CASE environments, making the point that there isn't much evidence of extensive use of such tools and environments in actual production of code. This was followed by historical examples from three broad categories: language-centered environments, framework-based environments, and tool-kit environments. This led to a discussion of common observations found to hold true across large organizations using CASE environments:

- Naïve expectations about the benefit of the environment.
- Poor understanding of the nature both of an environment and of the CASE tools that might populate it.
- Lack of technical awareness of the difficulty and cost of actually integrating several tools

This was followed by an overview of the current setting for CASE environments (open systems, COTS software, better understanding of software architectures, object oriented technology, and the World Wide Web). She concluded with some predictions regarding new attempts to produce CASE environments:

- focus will be on tractable portions of functionality.
- assumption will be that processes and tools are intimately related.
- dependence will be on economic considerations.
- influence will come from the current trend in COTS-based systems.
- strong influence will come from the current trends in software architectures.

COTS Use: Theory and Practice
Dr. Rami Razouk
The Aerospace Company
11/06/96

This was an informal talk given as background. As such slides are not available for distribution. Salient points of the briefing are presented below:

- ◆ Goals in using COTS products include shared/lowered development costs and higher end-product stability.
- ◆ In the practice of COTS product integration at Aerospace there are many examples of successes *and* failures.
- ◆ To ensure continued successful COTS product integration efforts, open products are not enough; open *architectures* also need to be developed.

A Tool builder's View of COTS Integration
Dr. Anthony Wasserman
11/06/96

Dr. Wasserman discussed his experiences at IDE with the *Software Through Pictures* environment. He began with an overview of environments and tools falling under the following categories: programming environments, documentation tools, requirements tools, configuration management, schema generations and version control. From these examples he derived several observations: all involve point-to-point integrations, no standards used, all need commitment to updating, large software engineering effort required, and customers rarely pay for integrations. Then he discussed why tool integration is difficult (the need for interaction with a vendor; tool rarely designed for integration; incompatibility between tool releases; individual and unique data storage, environment, and resource requirements). He also made the point that tool builders are the users of other tools, which can impose their own constraints on tool development. Then he discussed tool interface decision issues, ranging from the minimum effort (entire tool is invoked from another program) to the maximum effort (everything within the tool can be invoked individually). This led to a discussion of the types of tool integration (data, control, presentation, platform and process integration). He concluded with a discussion of the future of tool integration mechanisms (tempered by the caveat that technical solutions may not always solve business problems):

- cross-platform standards (CORBA, ActiveX, TCP/IP).
- common use of components and libraries.
- helper applications and plug-ins.

COTS Integration Risks

Marvin Carr

SEI

11/06/96

Mr. Carr gave a presentation relating COTS integration risks to the SEI Risk Taxonomy and Risk Repository. The SEI study collected empirical data on COTS integration risks from thirty-six risk assessments conducted by SEI via interviews with COTS integration practitioners. From analysis of the data collected, SEI identified seven areas of concern with respect to COTS integration activities (documentation, performance, maturity, interface, verification, vendor support/upgrades, and data rights). He concluded with the admonition that COTS software is not often integrated truly “right off the shelf.” Considerable modified code may be needed to make it work, and upgrades are frequently significantly more work and error-prone than anticipated.

COTS Integration Process Model
Karen Kelley
Lockheed Martin
11/06/96

This was an informal talk given as background. As such slides are not available for distribution. Salient points of the briefing are presented below:

- ◆ One new process model for COTS software integration includes *implementation* of lessons learned from prior corporate experience by using those experiences in business case studies.
- ◆ Company began improving COTS integration processes by educating *themselves* about COTS integration issues.
- ◆ Company is continuing to improve COTS integration processes by now expanding its education efforts to include educating the *customer* about COTS integration issues.
- ◆ Company has also developed a web browser based *intranet* COTS integration training tool called PRIME (Process for Rapid Integration, Migration and Evolution). It captures corporate COTS integration experience in a single repository available to engineers and managers company wide.

Breakout Groups

Architecture Technology Solutions

Risk Management Approaches

Cost Estimation Approaches

COTS Integration Cost Estimation Approaches Group Discussion Summary

Moderator: Barry Boehm (USC)

Participants: Marvin Carr (SEI)
Sunita Devnani (USC)
Roger Dziegiel (Rome Laboratories)
Gary Thomas (Raytheon E-Systems)
Peggy Wells (USAF/ESC)

Scribe: Christopher Abts (USC)

Initial evening session 11/06/96:

First order of business for the group was to agree upon a discussion agenda (though each item was not necessarily approached in the given order):

Agenda

- Review proposed CSE COTS integration cost model
 - model appropriateness
 - determining initial parameters
 - data collection
- Review CSE preliminary COTS questionnaire and presentation items
- Develop incremental strategy for COTS model development
 - size, breakage factor definitions (lines of glue code and/or Unadjusted Function Points from weighted External Interface Files)
 - data definitions and rating scales
 - milestones
 - scope
 - data collection instrument and Round 2 questionnaire
- Maintenance version of model
- Effort distribution by sources of effort

This was followed by some random exchanges to further stimulate ideas:

Issues to consider for the second round CSE COTS integration data collection questionnaire:

- Need standards for categorizing effort with respect to COTS integration:
 - what effort data would organizations likely have?
 - what effort data might they roll up from existing sources?
 - could effort be defined in something other than raw hours?
 - this is a tougher problem in labor-hour definition than occurred with COCOMO 2.0 due to its narrow target: capturing effort related solely to COTS integration!
 - the proper definition of labor-hour in the next survey could be very activity or labor specific.
- Five recognized sources of COTS integration effort (*COTS assessment, COTS tailoring, Glue code development, Application volatility, and IV&V*) frequently captured in disjoint places in the WBS (Work Breakdown Structure) - if captured at all!
 - Validation & Verification part often hiding in Integration & Test.
 - Ripple effects of COTS integration can be gobbled up in General Integration.
 - And where is COTS integration effort due to *breakage* captured?
- For the above reasons, COTS integration effort data collected for the second round CSE COTS survey may consist of a significant amount of “engineering judgment.”
- Definition of COTS software:
 - what definition is the most beneficial?
 - would a downstream relaxation of the bounded definition currently used in the preliminary CSE COTS survey be useful for further work?
- To be most effective, conducting the second round COTS survey ideally should involve as many in-person interviews/site visits as feasible to facilitate consistent responses to the survey.
- Appropriate individuals to include in the second survey might include the systems engineers as well as the program managers and programmers.

Morning session 11/07/96:

Began by characterizing sources of effort, arriving at the following assessments:

COTS Integration Sources of Effort

- COTS Assessment (pre- and post-commitment)
 - of functionality, performance, interoperability, COTS architectural match to application, etc. (covered in COCOMO 2.0 by driver RESL - Architecture/Risk Resolution)
 - *effort small but influential*
- COTS Tailoring, Tuning and Installation
 - effects of platform, other COTS products
 - *effort small but influential, sometimes compensated for in other areas*
- COTS Glue Code Development
 - similar to other COCOMO 2.0 s/w development estimations
 - *effort ranges small to large depending upon COTS product being integrated, but is usually large*
- Application Volatility due to COTS Products (including Configuration Management)
 - COTS volatility, shortfalls, learning curve
 - *effort generally large*
- Added Application IV&V Effort (including Documentation)
 - COTS option and stress testing (e.g., of extra features)
 - debugging complications, incorrect fixes
 - *effort medium to large (there is some control over this effort; the more added COTS products, the more work; “ultimately depends on how well work was done up front”)*

General comments evolving out of the above:

- Importance of effort vs. duration of effort needed for a task are two different things!
- **Glue code is nearly always the bulk of the effort in a COTS integration task.**
- Question: do COTS-COTS, COTS-legacy, COTS-new application, etc., integration activities have different cost factors?

Answer: cost factors likely the same, but the ratings of those factors for the various activities may be different.

Next followed a closer study of the sources of COTS integration effort delineated previously:

COTS Assessment

Recognizing that this effort is already captured somewhat in the COCOMO 2.0 cost factor RESL (Architecture/Risk Resolution), the group looked at *Table 8: RESL Rating Components* from the *COCOMO 2.0 Model Definition Manual* (reproduced on the following page). Conclusion was that this table will be useful in defining ratings for a similar factor in a COTS integration cost model, but needs to be tailored more specifically towards COTS integration issues. In particular, another row to this table could be added to address *architectural mismatch*.

COTS Tailoring, Tuning and Installation

Referring to tables showing both the COCOMO 2.0 and Loral Model cost driver contributions across the five COTS integration sources of effort (reproduced here as tables A and B), the point was made that good documentation makes tailoring, configuring, etc., much simpler. The rating impact of these five sources of effort really requires a 3-D solution space; e.g., the more information you have about a product and/or vendor, the less important some of these other drivers may become.

A lot of COTS products can't be tailored at all, but problems with tailoring are often compensated by adding more functionality in the glue code.

COTS Glue Code Development

This effort can be small or large, depending upon the cleanliness and openness of the COTS product external interface elements, and how much additionally functionality must be added to the glue code for the reasons noted above under COTS tailoring. As indicated previously, however, this effort is usually large, and in fact is almost invariably the source of greatest required effort during the integration task.

Application Volatility due to COTS Products

This can be a very large effort to manage and/or contain, because the developer rarely (read: almost never!) has much control over when and how often the COTS vendor releases new versions of their product. This can become particularly acute if the overall software development project encompasses a large system whose development is spread out over a significant period of time. During that same period, the COTS products vendors, in order to stay competitive within their markets, have likely released multiple updates of their own products, which by default can lead to significant volatility in the main application software as the developer struggles to keep in step with the COTS vendors.

Table 8: RESL Rating Components

(excerpted from *COCOMO 2.0 Model Definition Manual, USC 1996*)

Added Application IV&V Effort

COTS products usually come with more functionality than is needed, but the developer dare not forego doing IV&V on the *entirety* of functionality offered by the COTS product to avoid unexpected interactions and problems.

More general comments:

- Often it is difficult to avoid “double counting” some sources of effort.
- Assessment of sources of effort almost require a “risk” component.
- Differences in programmers’ experience makes an order of magnitude difference in being able to handle COTS integration; e.g., “people who have spent much time learning traditional batch programming have more trouble with COTS integration vs. those who have spent time pulling applications off the Internet and getting a good intuition as to what’s in those packages.”
- To really *know* what’s in a COTS product, as quickly as possible jump past the procurement people to one-on-one peering with operations/engineering people to get the true information about a product.
- Look at the database required for a COTS product; if proprietary, you’ve got problems; if open or substitutable, you’ll be okay.
- Regarding multiple level COTS products: buying COTS products with other COTS products embedded within, i.e., “tiered” COTS products, adds loads of complexity and you lose track of who is really responsible for fixing problems.
- Question to ask during 2nd round COTS data collection interviews: how much effort percentage-wise is allocated across the *Rationale: Cost Contributors* tables? (Response will likely be based upon “engineering judgment.”)

Next topic approached was a COTS Integration Cost Estimation Procedure, with discussion converging on the following:

COTS Integration Cost Estimation Procedure

- I. Sizing
 - A. determine Unadjusted Function Points (UFP) from External Interface Files (EIFs)
 - B. includes COTS-COTS, COTS-application, COTS-platform (the operating system, not the database, though not always)
- II. Breakage
 - A. estimate percentage of UFP breakage (BRAK) during the development period
 - B. function of: number of COTS packages, average new releases per package, average interface breakage per release

(continued)

COTS Integration Cost Estimation Procedure (continued)

3) Effective Size

- A. $ESIZE = UFP (1.0 + BRAK/100)$
- B. *(an additive term to account for installation, assessment effort independent of size might be warranted)*

4) Scaling

- A. initially assumed linear for simplicity
- B. $B = 1.0$

5) Effort Multipliers

- A. approach same as in COCOMO 2.0
- B. EM_i

6) Estimated Effort

- A. Person Months (PM) = $A * (ESIZE)^B * \Pi (EM_i)$
- B. (A is a constant used to capture the multiplicative effects on effort with projects of increasing size)

7) Effort Distribution by Source-of-Effort

- A. *downstream possibility, not included in current model design*

General comments on this topic:

Sizing

- Estimate Function Points by “shalls” instead of *sloc* and size estimate comes out reasonably well!
- Sometimes the operating system should be considered as a COTS product.

Breakage

- Similar as used in COCOMO 2.0; but not so much a function of requirements volatility as how much breakage occurs due to multiple COTS products and releases!
- Should provide examples of how to estimate breakage by specific case examples.
- Interfaces between multiple COTS products being integrated into one system sometimes change, sometimes don't.

Effective Size

- Should there be an additive term capturing assessment, installation factors, etc., *not* wrapped into size estimation directly?
- Might *aggregated* COCOMO 2.0 factors apply to a COTS model?

Scaling

- Initially assuming a linear model for simplicity, though recognize that as COTS integration tasks get bigger, involving more COTS products, the effect on effort becomes exponential.
- Question: does adding more COTS products to a system development exhibit the same diseconomies of scale behavior as does adding more people?

Effort Multipliers

- Once defined, these will be handled as in COCOMO 2.0; i.e., as multipliers which either increase or decrease the nominal effort according to the assessment of the impact of a given cost factor in the current software development project.

Estimated Effort

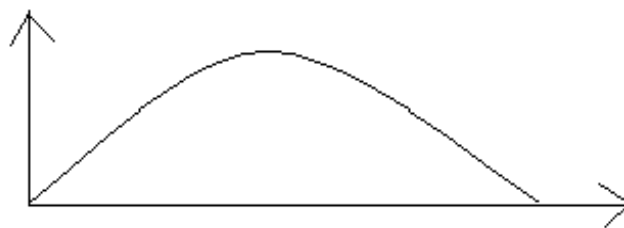
- The final output of the proposed model as determined from the modeling equation shown above.

The next activity pursued by the group was to compare all the cost drivers listed independently in tables A, B and C to try to reduce them into a single set of prioritized candidate cost drivers for the CSE COTS integration cost model. The result can be found in tables D and E.

General comments related to this activity are as follows, focusing on Loral factors:

- Vendor maturity doesn't necessarily affect glue code/application development in a positive way (a mature vendor can have an immature product).
- The level of support you receive from a COTS vendor is a function of how important a customer you are to that vendor.
- Level of support received from a vendor actually follows a convex curve:

Support Offered



Vendor

Behemoth (e.g., Microsoft)

Young

- Product and vendor maturity are somewhat correlated.
- Increased effort due to frequency of COTS product release cycles can be captured in *breakage*.
- Need to rename/clearly define “Vendor Extension Responsiveness” - a mix of *vendor cooperation/COTS product vendor support* as defined in the CSE 1st round COTS survey and the Loral definition of *extensibility*; really the willingness of the vendor to modify/extend the capabilities of their COTS product to meet the developer’s specific needs.
- *Training* and *quality of documentation* can be inversely correlated; e.g., good documentation may obviate the need for extensive formal training.
- The definition of *training* can remain as defined by Loral.
- *Administrative effort* is mostly captured in the *breakage* factor.
- *Portability* needs to be redefined as portability as demanded by requirements; handled similarly to *reliability*.
- *Reliability* is key on safety critical systems, but is not such a big deal on other kinds of systems.

Afternoon session 11/07/96:

Discussion of candidate cost drivers continued, this time focusing on the COCOMO 2.0 drivers:

- Quality of COTS documentation should be defined relative to desired documents.
- Referring to *Table 20: Module Complexity Rating vs. Type of Module Components* for the COCOMO 2.0 post-architecture model in the *COCOMO 2.0 Model Definition Manual* (reproduced on the following page): too much is being captured in *complexity*.
- A preference expressed for separating *complexity* and *reliability*.
- *Reliability* and *documentation* are related.
- *Complexity* and *data* are related.
- *Reliability* in the CSE COTS model will be defined as *COTS reliability relative to required system reliability*.
- Designing “reusable” glue code seems like an unlikely occurrence; *required reuse* can be removed as a cost factor in terms of COTS integration
- People often confuse the difference between the effort to reuse code and the effort to design code for reuse!
- The same is true for the difference between *configuration management* and *change management*.

Table 20: Module Complexity Ratings versus Type of Module

(excerpted from *COCOMO 2.0 Model Definition Manual, USC 1996*)

- *Platform difficulty* ; redefined as “performance.”
- *Platform volatility* is captured in *breakage*.
- Need to benchmark COTS product performance *before* and *after* integration to see if the various COTS products are “knocking into each other.”
- *Personnel capability* as defined in COCOMO 2.0 is okay.
- Multiple development/delivery sites is a configuration management issue.
- *Process (tools, sites, etc.)*; redefined only as *process maturity* as defined by SEI’s 18 KPAs.
- A *schedule* function for COTS integration makes no sense in a model independent of the whole system development process.
- *Documentation* is pretty much correlated with everything, both positively and negatively!
- *Breakage* will need a good list of what it encompasses.
- Effort to size is going to be a *big*, particularly if done in terms of Function Points where it is meant explicitly count only external interfaces.
- DoD will count in Function Points, most everybody else will count in *sloc* (with the exception of MIS arena).
- Question: can we come up with some kinds of “proxies” to solve the sizing in Function Points problem?
- The application domain accounts for the wide variance in converting Function Points to *sloc* per specific language (the inverse of *backfiring*).
- 2nd CSE COTS survey question: ask for an estimate of how many lines of glue code were written, what language was used, and if available, how many Function Points were determined to be needed.
- In designing definitions for cost drivers, be careful to create truly orthogonal definitions to prevent correlation!

Future plans for the CSE COTS integration cost model were then discussed:

- Address a maintenance model incorporating COTS related issues.
- Maybe add a section of maintenance drivers/questions in the 2nd round CSE COTS survey to in effect “kill two birds with one stone.” -- *an appealing idea, but one which could also over complicate a likely already overly complicated survey!*

The final activity broached by the group was to create an incremental strategy for the further development of the CSE COTS integration cost model:

Incremental COTS Model Development Strategy

- Near term
 - size, breakage factor definitions
 - data definitions and rating scales
 - further model scoping
 - new development milestones
 - data collection instrument and 2nd round COTS questionnaire
 - calibration of the model

- Longer term
 - maintenance version of the COTS integration cost model
 - effort distribution by source-of-effort
 - risk assessment based on cost driver ratings
 - low COTS maturity
 - high COTS volatility
 - low personnel capability

-- *finis* --

Conclusion

**USC-CSE Response to Workshop Ideas:
COTS Integration Process Approaches
Dr. Barry Boehm
USC
11/08/96**

Dr. Boehm's USC-CSE response briefing related a number of the COTS integration process issues (early risk identification and resolution; anticipating and architecting for users' and maintainers' downstream evolutionary needs; early reconciliation of stakeholders' COTS-related win conditions) to USC-CSE's research on the WinWin Spiral Model and the WinWin requirements/architecture negotiation tool. The *Workshop Highlights* discussion found at the front of these proceedings elaborates on the process issues. The resulting discussion identified further Affiliate suggestions for USC-CSE research priorities in the architecture and cost modeling areas—again, these are summarized in the *Workshop Highlights*.

Wrap-up
Dr. Barry Boehm
USC
11/08/96

The Wrap-up discussion focused on suggestions for future workshops. Its primary outcome was to establish COTS Integration as the topic for CSE's next Executive Workshop to be held at USC in Los Angeles on March 12, 1997.

Appendices

Appendix A

**"COTS Integration: Application
of Lessons Learned"
Lockheed Martin**

Appendix B

"Anchoring the Software Process"

Barry Boehm

Appendix C

**"Best Current Practices: Software
Architecture Validation"
AT&T/Lucent.**

Appendix D

CSE Preliminary COTS Survey

Appendix E

Conference Attendees