

# Value-Based Processes for COTS-Based Applications

**Ye Yang, Jesal Bhuta, and Barry Boehm,** *University of Southern California*

**Daniel N. Port,** *University of Hawaii*

COTS-based applications pose unique development challenges that traditional process models can't anticipate. A value-based set of processes can help steer these projects toward successful, cost-effective integration.

**E**conomic imperatives are changing the nature of software development processes to reflect both the opportunities and challenges of using COTS products. Processes are increasingly moving away from the time-consuming composition of custom software from lines of code (although these processes still apply for developing the COTS products themselves) toward assessment, tailoring, and integration of COTS or other reusable components. Two factors are driving this change:

COTS or other reusable components can provide significant user capabilities within limited costs and development time, and more COTS products are becoming available to provide needed user functions.

We see how these economic drivers are reshaping software development in the six-year trend for COTS-based applications (CBAs) in USC's annual series of rapidly developed campus electronic services applications. Figure 1 shows the percentage of CBAs has risen from 28 percent in 1997 to 70 percent in 2002. In 1997, for example, most development teams programmed their own search engines and Web crawlers. By 2002, these functions were readily available as COTS or open source packages, enabling the teams to provide much more user capability within a short (24-week) definition, development, and deployment schedule. The Standish Group's 2000 survey

found similar results (54 percent) in industry.<sup>1</sup>

Traditional software process models fail to accommodate many challenges of using COTS, however, because their process guidance is overly sequential (as with waterfall-based models<sup>2</sup>) or underdetermined (such as EPIC<sup>3</sup>). This often leads to the selection of best-of-breed but incompatible COTS products, without considering the increased costs and reduced benefits incurred by trying to glue these together. CBA project analyses conducted while developing the Constructive COTS-based development (COCOTS) cost estimation model<sup>4</sup> led us to evolve a value-based set of processes that helps CBA project teams avoid or minimize such value losses. We use a supply chain example to illustrate how the CBA process-decision framework enables CBA projects to generate process element combinations that best fit their project situation and dynamics.

## Five principles for CBA development

We define a COTS-based application as a system for which COTS products provide at least 30 percent of the end-user functionality (in terms of functional elements: inputs, outputs, queries, external interfaces, and internal files) and COTS considerations consume at least 10 percent of development effort. We observed similar behavioral CBA boundaries in the e-services projects.<sup>5</sup> Projects with low COTS-related effort often had considerable COTS infrastructure but relatively little COTS end-user functionality.

Our value-based set of processes for CBAs includes five primary principles, an associated process framework, and a set of composable process elements for developing CBAs. These are the principles for CBA development:

1. Process happens where the effort happens.
2. Don't start with "requirements."
3. Avoid premature commitments, but have and use a plan.
4. Buy information early to reduce risk and rework.
5. Prepare for COTS change.

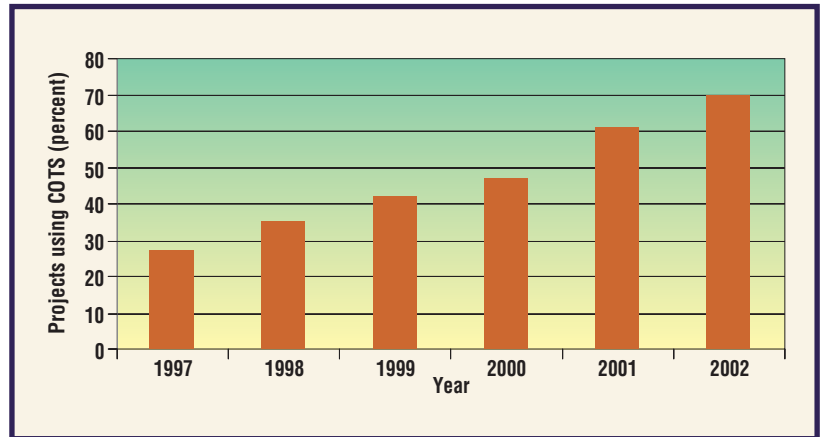
### Process happens where the effort happens

For CBAs, we define the three primary sources of effort:

- **Assessment:** Evaluating and selecting COTS products as viable components for a user application
- **Tailoring:** Configuring COTS software products for use in a specific context
- **Glue-code development:** Designing, developing, and using code to ensure that COTS products interoperate satisfactorily to support the user application

Figure 2 shows the CBA effort distributions for USC campus e-services applications and large industry-government CBAs. (The larger CBAs also averaged 13 percent additional effort for adapting to new releases; we'll discuss this in a moment.) We also found that similar effort distributions tended to produce similar process sequences.

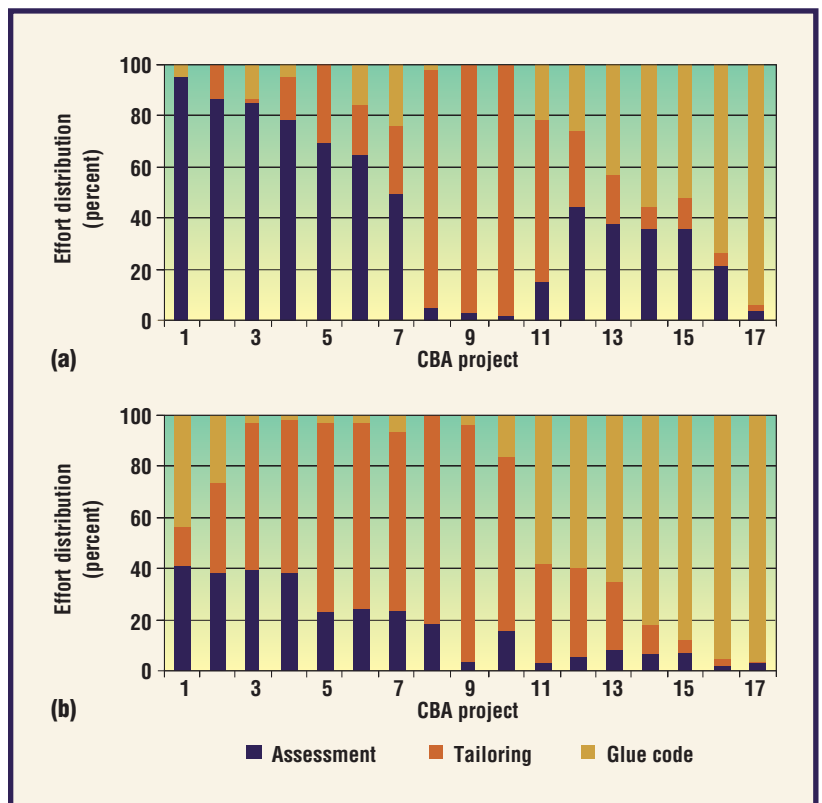
Figure 2 shows that no one-size-fits-all effort distribution or development process exists for CBAs. Some projects, particularly in the small e-services area, focused almost exclu-



**Figure 1. COTS-based applications' growth in USC e-service projects. (2000 data courtesy of the Standish Group)**

sively on assessment, with little tailoring or glue code required once the COTS products were selected. Others, generally applications supported by large, single COTS enterprise resource planning or Web portal packages, primarily required tailoring. Some CBAs needed extensive glue code development and integration because the selected best-of-breed COTS packages weren't designed to operate with each other. And some projects demanded a great deal of effort in all three areas.

**Figure 2. CBA effort distributions: (a) small e-service projects; (b) large industry projects.**



**The risk-driven spiral model offers a good process framework for addressing “how much COTS assessment is enough” issues.**

This variation implies the need for an integrated product and process development approach, as in the Capability Maturity Model Integrated guidelines.<sup>6</sup>

**Don’t start with “requirements”**

*The Merriam-Webster Dictionary* defines a *requirement* as “something claimed or asked for by right and authority.” Once the developers, customers, and users agree to call something a “requirement,” many customers and users will see the requirement as non-negotiable. This puts the developer on a fixed-price project or the customer on a cost-plus project in the untenable position of managing unmanageable expectations or overconstrained COTS solution options. Fundamentally, something isn’t a requirement if you can’t afford it.

This presents serious problems for requirements-first CBA processes such as waterfall-variant processes.<sup>2</sup> These are better than a pure waterfall for concurrent COTS package evaluation, selection, and requirements analysis. But committing to requirements before performing design and glueware integration analysis will likely create architectural mismatch problems, often causing factor-of-four schedule overruns and factor-of-five budget overruns.<sup>7</sup>

This kind of problem will confront organizations whose official acquisition processes are requirements-first waterfall models. It can even be a problem in agile methods such as Extreme Programming that espouse simple design and refactoring as the silver bullet for accommodating requirements change. If your early agile iterations for initial users lock you into a COTS product with limited growth potential, no amount of refactoring will get the COTS product to, for example, double its throughput when this is needed for later iterations.

In such cases, the best you can do is reinterpret the process, invest in enough concurrent engineering of overall requirements and COTS-based design to ensure a feasible combination before setting user expectations, and have a much later system requirements review that includes a viable system design and a COTS integration feasibility demonstration. Risk analysis (Principle 4) is key here.

**Avoid premature commitments, but have and use a plan**

So you follow Principle 2, but you still need a process model with enough planning content

to let you monitor the project’s progress toward completion. EPIC offers an example of an underdetermined CBA process model.<sup>3</sup> It can identify and provide insights on important COTS considerations, but its lack of intermediate milestones leaves it open to at least three major problems:

- It lacks guidance for what steps to perform next or how long to perform them.
- It generates very little status information for communicating and controlling progress toward completion.
- It increases the likelihood of nonconvergence, as in the “study-wait-study” cycle of performing a COTS evaluation study, noting that better new COTS capabilities were on the horizon, and waiting to perform a new study that concludes the same thing and repeats the loop.

**Buy information early to reduce risk and rework**

Project teams often scope the COTS assessment and selection process on the basis of COTS products’ likely cost. A better scope criterion is the amount of risk exposure (probability of loss × value of loss) involved in choosing the wrong combination. For example, a supply chain project team scoped its assessment based on the COTS products’ cost of US\$200,000, but the loss from picking an incompatible combination of best-of-breed COTS products ran about \$3 million plus eight months of delay. Had they invested more than \$200,000 in COTS assessment to include interoperability prototyping, they would have been able to switch to a more compatible COTS combination with only minor losses in effectiveness and without expensive, late rework.

The risk-driven spiral model offers a good process framework for addressing “how much COTS assessment is enough” issues. One good example<sup>8</sup> balances the risk of erroneous choices from insufficient COTS assessment with the risk of delay from excessive COTS assessment, to find a risk-driven “sweet spot” for the amount of COTS assessment effort. However, the general spiral model resembles the EPIC model in not providing COTS-specific process milestones and decision points. We provide a COTS-specialized spiral process framework and process elements here.

## Prepare for COTS change

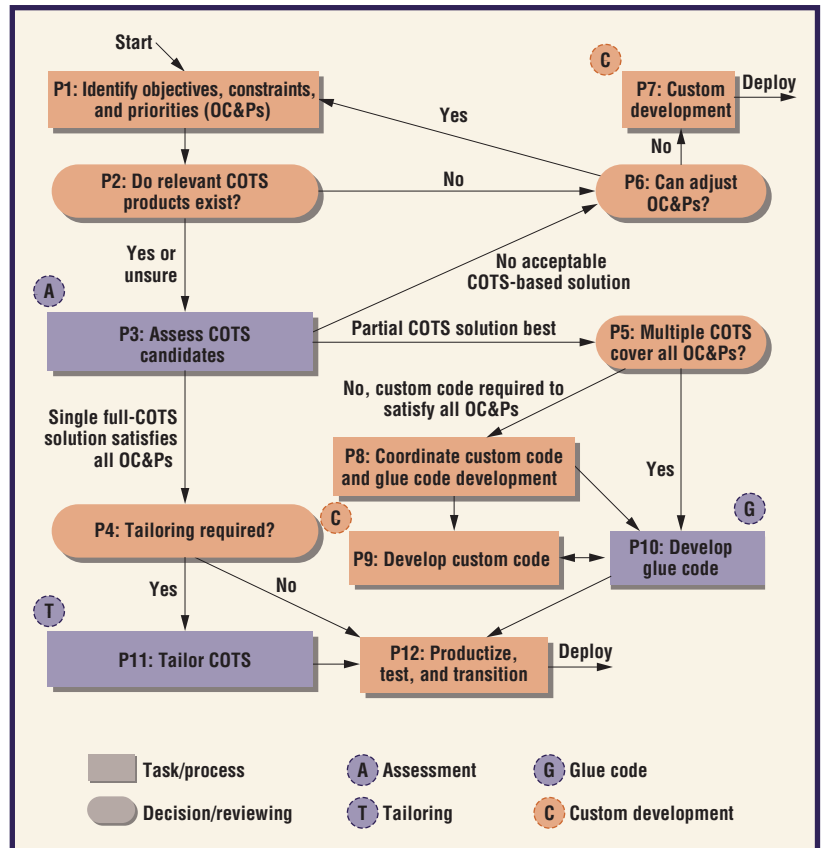
An annual survey of COTS change characteristics conducted at the aerospace-related Ground System Architectures Workshops reports that vendors upgrade their COTS products every 10 months, on average. The surveys also show that releases go unsupported after an average of three releases. Thus, if you attempt to stabilize on COTS release 3.0, you will typically find it going unsupported with release 6.0 about 30 months later. Or, if you outsource the development of a large application that takes more than 30 months, you may find that it's delivered with unsupported COTS releases. One large CocOTS calibration project had 120 COTS products delivered, 55 of which (46 percent) were no longer vendor-supported.

The challenge of vendor-controlled COTS change becomes even greater during software maintenance. The more COTS products an application has, the greater the challenge will be; an application with 120 COTS products would likely have an average of 12 new releases per month. If the COTS products are tightly coupled, the maintenance effort can scale as high as the square of the number of products.<sup>9,10</sup> Furthermore, as COTS vendors try for competitive differentiation, they are likely to introduce new features that are incompatible with those of other COTS products.

These challenges don't mean application developers have no way to cope with COTS change. Useful strategies include

- investing effort in a proactive COTS market-watch activity;
- developing win-win rather than adversarial relations with COTS vendors;
- reducing the number of COTS products and vendors;
- reducing inter-COTS coupling via wrappers, mediators, or interoperability standards;
- contracting for delivery of latest-release COTS products; and
- developing and evolving a lifecycle COTS refresh strategy that's synchronized with business cycles (such as annual product models or 18-month operator retraining cycles), using tailored versions of the process framework discussed next.

Further discussion of CBA maintenance lessons appears elsewhere.<sup>11</sup>



**Figure 3. A value-based CBA process decision framework. The process starts by “walking” a path from start to deploy that connects activities (boxes) and decisions (ovals). The circles indicate the assessment, tailoring, glue code, and custom code development process elements. Each area may enter and exit in numerous ways.**

## Value-based CBA process decision framework

In analyzing both USC e-services and industry CBA projects, we found that the better projects handled their individual assessment, tailoring, and glue code activities in similar ways at the process element level. We also found that these process elements fit into a recursive and reentrant decision framework that accommodates concurrent CBA activities and frequent go-backs based on new and evolving client needs and risk considerations. We therefore present a value-based CBA process decision framework and three composable process elements as a COTS-specialized risk-driven spiral framework.

### Process elements

Figure 3 illustrates the overall CBA decision framework comprising the assessment,

**Market trend analysis often proves useful for ensuring relevant, up-to-date assessment information.**

tailoring, glue code, and custom code development process elements within an overall development life cycle. This scheme was developed from and is consistent with the CBA activity distributions in Figure 2. We summarize the less obvious aspects of each process area next, focusing on three of the process elements (P3, P10, and P11) in more detail later.

**P1: Identify stakeholders' desired objectives, constraints, and priorities (OC&Ps).** First, stakeholders must identify and prioritize their value propositions (about features, platforms, performance, budgets, schedules, and so on) and negotiate a mutually satisfactory or win-win set of objectives, constraints, and priorities.<sup>12</sup> As the project progresses, risk considerations, stakeholders' priority changes, new COTS releases, and other dynamic considerations can alter the OC&Ps. In particular, if the team identifies no suitable COTS packages (P6), the stakeholders can change the OC&Ps, and the process begins anew with these revised considerations.

**P2: Do relevant COTS products exist?** In most cases, stakeholders know of some available COTS packages that can provide part or all needed functionalities. COTS assessment then helps them determine the best option. However, the project will follow a custom-development approach if the stakeholders realize that no relevant COTS products are available.

**P4: Is tailoring required?** When a certain COTS product can satisfy all the OC&Ps, application code or glue code isn't necessary. The team might still need to tailor the selected COTS to work for the specific system context.

**P5: Will multiple COTS cover all OC&Ps?** If a combination of COTS products can satisfy all OC&Ps, integrate them with glue code. Otherwise, combine COTS packages to cover as many OC&Ps as feasible and develop custom code to cover what remains. The supply chain example mentioned previously shows us the assessment should address multi-COTS integration and glue-code risks.

**P6: Can we adjust OC&Ps?** When you can't identify any acceptable COTS products, reexamine the OC&Ps for areas that may allow

more options. Can any constraints or priorities be relaxed? Can the objectives (easier to modify than "requirements") be modified to enable consideration of more products? Are there analogous areas in which to look for more products and alternatives?

**P8: Coordinate application code development and glue-code effort.** Custom components must eventually be integrated with the chosen COTS products. To ensure the custom interfaces' compatibility with the COTS products and the particular glue-code connectors used, coordinate the glue-code effort with the custom development. This coordination should continue through the concurrent development of custom code and glue code (P9, P10).

The framework in Figure 3 looks sequential, but its elements support recursive and reentrant use to support the frequent go-backs involved in CBA processes. These can include the emergence of new COTS candidates or releases, changes in enterprise-wide COTS preferences, or the emergence of important new users with additional value propositions or OC&Ps. We've studied these and found frequently occurring ATGC patterns or "genetic codes" that characterize CBA processes.<sup>13</sup>

### **P3: Assessment element**

Current COTS assessment process approaches identify key tasks and artifacts<sup>14-16</sup> but fail to address the concurrency and high coupling among COTS assessment, tailoring, and glue code development. Figure 4 shows the CBA assessment steps that provide these links.

*Entry conditions* include the stakeholders' negotiated system OC&Ps and the availability of relevant COTS products.

**A1: Establish evaluation requirements.** This step includes establishing COTS evaluation criteria and their corresponding weights, business scenarios, and COTS candidates. These derive from the OC&Ps. In the supply chain application, separate OC&Ps were initially defined for the application's enterprise resource planning (purchasing, receiving, sales, accounting, and shipping), advanced planning and scheduling (demand, production, distribution planning and scheduling), transportation management, and customer relationship management parts.

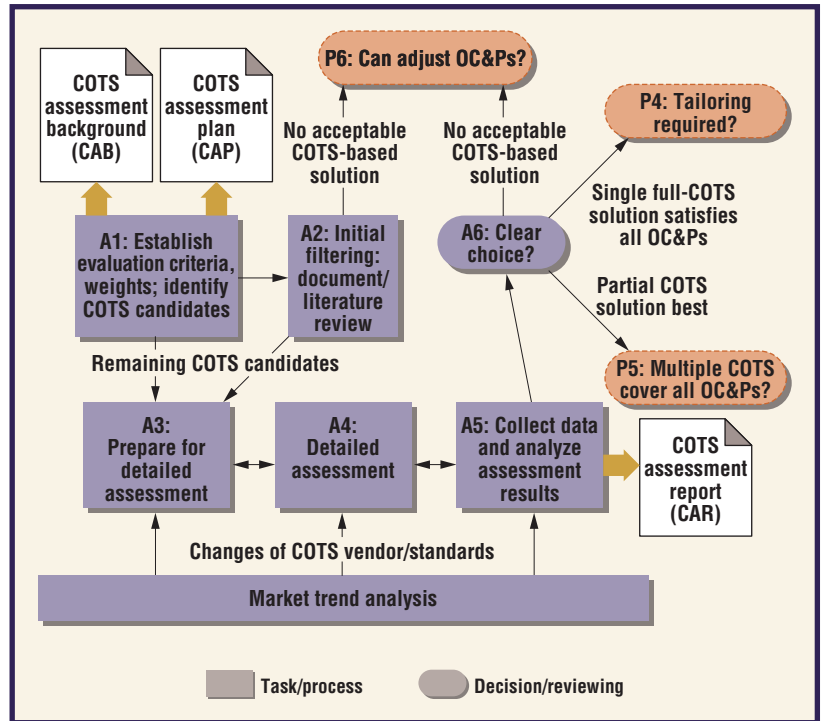
**A2: Initial filtering.** Initial assessment tries to quickly filter out the COTS packages that don't meet the evaluation criteria. This reduces the number of COTS candidates needing detailed evaluation. If no available COTS products pass this filtering, this assessment element ends up at the "none acceptable" exit.

**A3: Prepare for detailed assessment.** Remaining COTS candidates undergo more detailed assessment, and some cases require multiple rounds of detailed assessment to incorporate iteratively refined evaluation criteria. Example preparation activities include vendor contact, obtaining the COTS product (or a test version), installation, and refining evaluation criteria and weights.

The supply chain application started with an \$80,000 effort to use separate weighted-sum evaluations of the best-of-breed enterprise resource planning, advanced planning and scheduling, transportation management, and customer relationship management COTS package candidates based on documentation reviews, demos, and reference checking. The team belatedly found the resulting best-of-breed COTS choices to have many technical and business-model-assumption incompatibilities, leading to the \$3 million, eight-month overrun and associated system effectiveness shortfalls. A better approach would have been to use the separate analyses to further filter the leading choices and identify the major interoperability risk exposures, then use the size of the overall risk exposure to scope a more detailed COTS interoperability assessment, as in the next step.

**A4: Conduct a detailed assessment.** This step applies such techniques as prototyping, black-box testing, interoperability testing, and business-case analysis on COTS candidates to evaluate their relative fitness. Often, to obtain satisfactory evaluation data the team must tailor some COTS products (to assess usability, for example) and develop glue code to integrate some of the COTS with each other and with the overall system (to assess interoperability). Therefore, the tailoring and glue-code process elements (discussed in the P11 section) might be initiated separately or concurrently to support detailed assessment.

**A5: Collect evaluation data and analyze evaluation results.** The team collects data and in-



**Figure 4. The composable CBA assessment process element.**

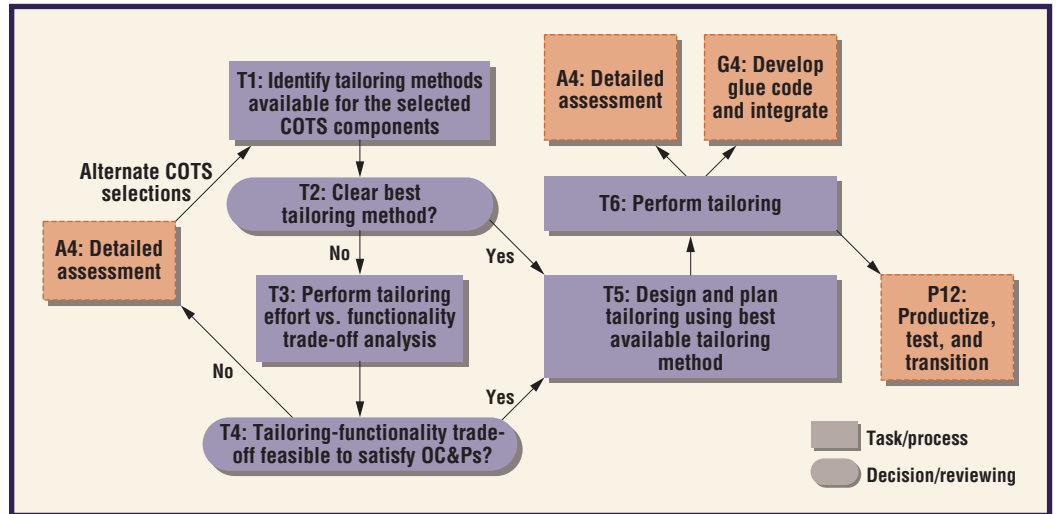
formation about each COTS candidate and analyzes it against evaluation criteria to facilitate trade-offs and decision making. In this step, a screening matrix or analytic hierarchy process is a useful and common approach to analyze collected evaluation data.

**A6: Is there a clear COTS choice?** This step establishes the exit direction from the COTS assessment process:

- *Single full COTS solution best:* A single COTS product covers all desired OC&Ps.
- *Partial COTS solution best:* A single COTS product or a combination of COTS products will work but covers only some OC&Ps, requiring custom development or glue code development to meet all desired OC&Ps.
- *No acceptable COTS:* Pure custom development is the optimal solution unless stakeholders are willing to adjust unsatisfied OC&Ps.

Market trend analysis often proves useful for ensuring relevant, up-to-date assessment information. For example, use a market watch activity to get the latest information regarding COTS products or standards, and collect COTS information from current users.

**Figure 5. The composable CBA tailoring process element.**



We also developed a set of three lightweight assessment process guidelines:

- The *COTS assessment background* document provides the essential set of OC&Ps and situation background needed to perform the COTS assessment.
- The *COTS assessment plan* document covers the “why/whereas, what/when, who/where, how, and how much” aspects of the activity being planned.
- The *COTS assessment report* document presents the major results, conclusions, and recommendations.<sup>17</sup>

The ISO/IEC 14598-4 standard<sup>16</sup> defines a set of artifacts with similar goals, but we found it important to include some specific guidance not found in the standard. This guidance includes results-chain analysis, value-based OC&P identification, and process planning to support concurrent COTS assessment, tailoring, and glue-code activities.

### P11: Tailoring element

In most cases, you must tailor COTS packages to work in a specific system context. Figure 5 illustrates the tailoring process element. It can be used for a single COTS tailoring activity or to tailor several COTS products simultaneously.

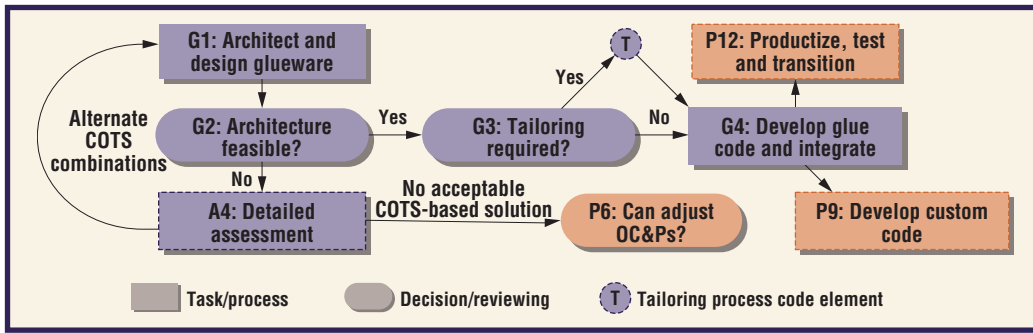
*Entry conditions* include the COTS package to be tailored and the activity that initiates the tailoring. For example, the COTS component can be under consideration by the assessment element, adapted and ready for integration with other components, or fully assessed

but needing some specialization for the particular application. This helps determine the exit direction from tailoring activity.

**T1: Identify tailoring options.** Tailoring options include GUI operations, parameter setting, or programming specialized scripts. If a COTS product has multiple tailoring options, the team must decide which options to use for implementing which capabilities.

**T2: Is there a clear best tailoring choice?** The best choice can be determined by either the COTS component, such as products supporting a single tailoring method, or by the developers’ tailoring knowledge and skills. If a dominant COTS tailoring option emerges, developers can proceed to design and plan the tailoring following that option (T5). If multiple tailoring options remain, developers must evaluate for the best one (T4).

**T3: Perform tailoring-effort versus functionality trade-off analyses.** When no single best tailoring method choice exists, the team must perform trade-off analyses between the effort that available tailoring methods require and the functionality to be achieved via tailoring. It’s common, even in the same product domain, for the tailoring effort to significantly vary depending on the COTS package selected. Automated tools such as Microsoft Excel’s macro recorder can significantly reduce the tailoring effort. Another factor to consider is the amount of retailoring that might be required during a COTS refresh cycle. Addi-



**Figure 6. The composable CBA glue-code process element.**

tional considerations include the need to implement a particular design, the complexity of tailoring needed, the need for adaptability and compatibility with other COTS tailoring choices, and available developer resources.

**T4: Is a tailoring-functionality trade-off feasible?** Review the trade-off analyses' results to obtain stakeholder commitment before preparing to do the actual tailoring.

**T5: Design and plan tailoring.** While tailoring might be straightforward for nondistributed, single-solution systems, it can be extremely complex for large distributed applications such as enterprise resource planning packages. In such cases, teams should plan the tailoring. Planning formats can vary from a simple checklist of steps to a complete set of UML diagrams (programmable tailoring). For the supply chain interoperability assessment, the team would choose a few higher-risk interoperability scenarios and selectively tailor the COTS candidates to support them.

#### **P10: Glue-code element**

In some fortunate cases, the combination of COTS and application components being integrated or assessed will easily plug and play together. If not, developers must define and develop glue code to integrate the components and perform an evaluation to find the best combination of COTS products, glue code, and application code to reduce the risk of the integrated components failing to perform as expected. Figure 6 illustrates the activities and decisions made when working with glue code.

*Entry conditions* include the combination of COTS products requiring glueware or custom code for successful operation and the activity that initiates the glue-code element. The latter condition helps determine the exit direction.


**G1: Architect and design glueware.** The first step in the glue-code process element involves designing the glueware required to integrate the applications. Major architectural considerations include determining interconnection topology options to minimize interaction complexity, selecting connectors<sup>10</sup> (such as events, procedure calls, pipes, shared memory, and database), ensuring support for connectors and architectural styles provided by COTS package interfaces, and identifying architectural mismatches between COTS packages.<sup>7</sup>

**G2: Is the architecture feasible?** This step focuses on acquiring stakeholder commitment to a given architecture. If the team determines no architecture can integrate the selected COTS packages within project constraints, they might need to revisit the assessment process element to identify an alternate combination of packages.

**G3: Is tailoring required?** The team must now determine if tailoring will be necessary to satisfy system OC&Ps. If so, the project will carry out the tailoring process element to meet system objectives. These tailoring activities can be situation-selective (as for the supply chain assessment prototype) or incremental in an incrementally fielded application.

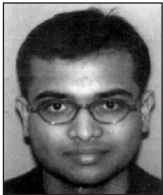
**G4: Develop glue code and integrate.** Finally, the team implements the connector infrastructure, develops appropriate interfaces (simultaneously with application code interfaces if necessary), and integrates components to build the application. For the supply chain assessment prototype, the assessment costs would have exceeded the \$200,000 COTS acquisition costs but would have been nowhere near the \$3 million, eight-month risk exposure.

**A**pplying the process framework and guidelines along with the risk-driven spiral model in the past two years has led to considerably more effective, efficient, and predictable execution of over 20 CBA

projects. Its client satisfaction track record of 93.7 percent on-schedule delivery of highest-value capabilities compares quite favorably with the Standish Report's 28 percent on-time success rates.<sup>1</sup> 

## About the Authors

**Ye Yang** is a PhD student in the Computer Science Department at the University of Southern California. Her research interests include empirically based software engineering on COTS-based development (software process, cost estimation modeling, and risk assessment modeling) and product line investment models based on Cocomo II. She received her master's degree in software engineering from Chinese Academy of Sciences. Contact her at the Center for Software Eng., Univ. of Southern California, 941 W. 37th Place, SAL Room 330, Los Angeles, CA 90089-0781; yangy@cse.usc.edu.



**Jesal Bhuta** is a PhD student at the University of Southern California's Center for Software Engineering. His primary research interests are COTS-based systems design and development and software risk management. He received his MS from the University of Southern California. Contact him at the Center for Software Eng., Univ. of Southern California, 941 W. 37th Place, SAL Room 337, Los Angeles, CA 90089-0781; jesal@cse.usc.edu.

**Daniel N. Port** is professor in the Department of Information Technology Management at the University of Hawaii and visiting associate at the Center for Software Engineering, University of Southern California. His research interests are strategic software engineering (economic methods, COTS, risk management, project development volatility metrics, and estimation models), empirically based software engineering, Model-Based (System) Architecting and Software Engineering (MBase), software product adoption, and software engineering education. Daniel received his PhD in applied mathematics and theoretical computer science from the Massachusetts Institute of Technology, where he specialized in theoretical computer science. Contact him at the Dept. of Information Technology Management, Univ. of Hawaii at Manoa, Honolulu, HI 96822; dport@hawaii.edu.



Los Angeles, CA 90089-0781; boehm@cse.usc.edu.

**Barry Boehm** is the TRW Professor of Software Engineering and director of the Center for Software Engineering at the University of Southern California. His research interests include software process modeling, software requirements engineering, software architectures, software metrics and cost models, software engineering environments, and value-based software engineering. His contributions to the field include the Constructive Cost Model (Cocomo), the Spiral Model, and the Theory W (win-win) approach to software management and requirements determination. He is a fellow of the ACM, AIAA, IEEE, and InCose and is a member of the US National Academy of Engineering. He received his PhD from UCLA in mathematics. Contact him at the Center for Software Eng., Univ. of Southern California, 941 W. 37th Place, SAL Room 337,

## References

1. Standish Group, *Extreme CHAOS*, tech. report, 2001, www.standishgroup.com.
2. M. Morisio et al., "Investigating and Improving a COTS-Based Software Development Process," *Proc. 22nd Int'l Conf. Software Eng. (ICSE 00)*, ACM Press, 2000, pp. 32-41.
3. C. Albert and L. Brownsword, *Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview*, tech. report, CMU-SEI-2002-TR-009, Software Eng. Inst., Carnegie Mellon Univ., July 2002.
4. C. Abts, *Extending the COCOMO II Software Cost Model to Estimate Effort and Schedule for Software Systems Using Commercial-Off-the-Shelf (COTS) Software Components: The COCOTS Model*, PhD dissertation, Univ. Southern California, Oct. 2001.
5. B. Boehm et al., "Not all CBS Are Created Equally: COTS Intensive Project Types," *Proc. 2nd Int'l Conf. COTS-Based Software Systems (ICCBSS 03)*, LNCS 2580, Springer-Verlag, 2003, pp. 36-50.
6. M.B. Chrissis, M. Konrad, and S. Shrum, *CMMI: Guidelines for Process Integration and Product Improvement*, Addison-Wesley, 2003.
7. D. Garlan, R. Allen, and J. Ockerbloom, "Architectural Mismatch, or, Why It's Hard to Build Systems Out of Existing Parts," *Proc. 17th Int'l Conf. Software Eng. (ICSE 95)*, ACM Press, 1995, pp. 179-185.
8. D. Port and Z.H. Chen, "Assessing COTS Assessment: How Much Is Enough?" *Proc. 3rd Int'l Conf. COTS-Based Software Systems (ICCBSS 04)*, LNCS 2959, Springer-Verlag, 2004, pp. 183-198.
9. V. Basili and B. Boehm, "COTS Based System Top 10 List," *Computer*, vol. 34, no. 5, 2001, pp. 91-93.
10. N.R. Mehta, N. Medvidovic, and S. Phadke, "Towards a Taxonomy of Software Connectors," *Proc. 22nd Int'l Conf. Software Eng. (ICSE 00)*, ACM Press, 2000, pp. 178-187.
11. D. Reifer et al., "COTS-Based Systems: Twelve Lessons Learned," *Proc. 3rd Int'l Conf. COTS-Based Software Systems (ICCBSS 04)*, LNCS 2959, Springer-Verlag, 2004, pp. 137-145.
12. B. Boehm, "Value-Based Software Engineering," *ACM Software Eng. Notes*, vol. 28, no. 2, 2003, p. 3.
13. D. Port and Y. Yang, "Empirical Analysis of COTS Activity Effort Sequences," *Proc. 3rd Int'l Conf. COTS-Based Software Systems (ICCBSS 04)*, LNCS 2959, Springer-Verlag, 2004, pp. 169-182.
14. P. Lawlis et al., "A Formal Process for Evaluating COTS Software Products," *Computer*, vol. 34, no. 5, 2001, pp. 58-63.
15. S. Comella-Dorda et al., "A Process for COTS Software Product Evaluation," *Proc. 2nd Int'l Conf. COTS-Based Software Systems (ICCBSS 03)*, LNCS 2580, Springer-Verlag, 2003, pp. 86-96.
16. *ISO/IEC 14598-4: 1999, Software Engineering—Product Evaluation—Part 4: Process for Acquirers*, ISO/IEC, 1999.
17. Y. Yang and B. Boehm, *Guidelines for Producing COTS Assessment Background, Process, and Report Documents*, tech. report, USC-CSE-2004-502, Univ. of Southern California, Feb. 2004.

For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).

## Become a IEEE Software reviewer

The key to providing you quality information you can trust is *IEEE Software's* peer review process. Each article we publish must meet the technical and editorial standards of industry professionals like you. Volunteer as a reviewer and become part of the process.

Become an *IEEE Software* reviewer today!  
Find out how at [www.computer.org/software](http://www.computer.org/software).