A Process for COTS Software Product Evaluation

Santiago Comella-Dorda
John Dean
Grace Lewis
Edwin Morris
Patricia Oberndorf
Erin Harper

July 2004

TECHNICAL REPORT
CMU/SEI-2003-TR-017
ESC-TR-2003-017

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

A Process for COTS Software Product Evaluation

CMU/SEI-2003-TR-017
ESC-TR-2003-017

Santiago Comella-Dorda
John Dean
Grace Lewis
Edwin Morris
Patricia Oberndorf
Erin Harper

July 2004

Integration of Software-Intensive Systems Initiative

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Christos Scondras
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

Deleted: 1744444445454646474954252555759616365676973

List of Figures

List of Tables

Abstract

The growing use of commercial software products in large systems makes evaluation and selection of appropriate products an increasingly essential activity. However, many organizations struggle in their attempts to select appropriate software products for use in systems. As part of a cooperative effort, the Software Engineering Institute and National Research Council Canada have defined a tailorable commercial off-the-shelf (COTS) software product evaluation process that can support organizations in making carefully reasoned and sound product decisions. The background fundamentals for that evaluation process, as well as steps and techniques to follow, are described in this report.

1 Fundamentals of COTS Software Product Evaluations

Using commercial off-the-shelf (COTS) software products in large systems provides many benefits, including the potential of rapid delivery to end users, shared development costs with other customers, and the opportunity to expand capacity and performance as improvements are made in the products. For systems that depend on COTS products, the evaluation and selection of appropriate products is essential to the success of the entire system. Yet many organizations struggle during the evaluation and selection process.

1.1 COTS Products and COTS-Based Systems: Definitions

Although the meaning of COTS may seem obvious, in practice determining what is COTS and what is not can be complex. The definitions in this report are by design imprecise, since the COTS market continues to define and redefine itself.

We define a *COTS product* as one that is

- sold, leased, or licensed to the general public
- offered by a vendor trying to profit from it
- supported and evolved by the vendor, who retains the intellectual property rights
- available in multiple, identical copies
- used without modification of the internals

Some products are COTS-like, but, based on our definition, are not pure COTS products. Freeware and open source products, for example, are available to the public, but no vendor is trying to profit directly from selling the product; in the case of open source, the community acts like a vendor in enhancing the product. Products that are available for sale to approved organizations or governments but not the general public are also COTS-like, as are software products developed for internal uses but opportunistically offered for sale to a particular customer. The Defense Commissary Information System (DCIS) program warns against such “opportunities.” The DCIS found that a company that has not created a product with the intention of making money from it is unlikely to support and evolve that product the way normal vendors would.

The point of this definition is to capture the characteristics that most people have in mind when they say “COTS.” Deviations from this definition simply change the corresponding set of expectations and risks. The evaluation principles and process described herein apply equally to COTS products and those that are COTS-like.

Although our definition of a COTS product does not include products that are modified internally, a particularly difficult distinction exists between tailoring and modifying a COTS product. We consider tailoring to mean changes to COTS software product functions along parameters the vendor has predetermined. We consider modification to mean changes to the internal capabilities of a product that are not part of the vendor’s original intent; generally, this means any change to the vendor’s source code. However, many COTS products provide only a core capability, with vendor or third party support for extensive tailoring of product functions. In these cases, often only the core capability is considered COTS. In general, the degree to which the tailored capability has moved away from pure COTS is roughly indicated by the magnitude of the management, handling, and potential retrofitting of changes required when new releases of the core product are installed in the deployed system.

COTS-Based Systems

Our definition of a *COTS-based system* (CBS) is any system partially or completely constructed using COTS software products as integral components. This definition is primarily aimed at software systems but does not specifically exclude hardware. The term “integral” is important because it highlights the tight relationship that the COTS product has with the rest of the system. The system would not function or would be incomplete without the product.

Measures like “percent of system composed of COTS products” or even “percent of functions provided by COTS products” are not part of our definition of a COTS-based system. While these and similar concepts might be heard elsewhere, they are misleading because they fail to capture the extent to which COTS products are central to fulfilling the mission of the system. The percentages are also difficult to measure because there is no clear basis for calculating them, and even when they can be measured, they offer little value.

A broad spectrum of COTS-based systems exists, ranging from COTS-solution systems to COTS-aggregate systems. A *COTS-solution system* is a single product or suite of products, usually from a single vendor, that can be tailored to provide the system’s functionality. Vendors offer such solutions if a consistent and well-bounded range of end-user needs exists throughout a broad community, justifying the vendors’ costs for developing the products or suites of products. Significant tailoring is required to set up and use these products, and the ability and willingness of an organization to understand and adopt the processes supported by the products are often key factors in success or failure. COTS-solution systems are commonly found in such well-established domains as personnel management, financial management, manufacturing, payroll, and human resources. Typical software vendors in this area include PeopleSoft, Oracle, and SAP.

COTS-aggregate systems are systems in which many disparate products (from different and sometimes competing vendors) are integrated to provide a system's functionality. Such systems are created if operational procedures are sufficiently unique to preclude the possibility of a single COTS product solution, if the constituent technologies are immature, if the scale of the system is large enough to encompass several domains, or simply because different products provide distinct pieces of functionality to form the complete system. Systems with these characteristics include software support environments, large information systems, and command-and-control systems. Often, the COTS products and other components are combined in ways or to degrees that are unprecedented. Figure 1 shows this spectrum of COTS-based systems.

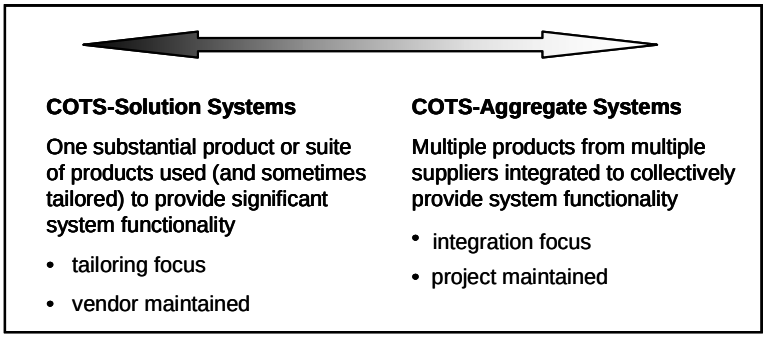


Figure 1: A Spectrum of COTS-Based Systems

1.2 Why a COTS Product Evaluation Process?

All organizations faced with the prospect of constructing major software systems from COTS products must evaluate the available commercial products to determine their suitability for use in a particular system. Yet, as we study the experiences of some of these organizations, we find one of the hard lessons of COTS product evaluation: reasonable people doing reasonable things still have problems that can be traced to the quality of their evaluation process.

One system we studied was initially conceived in the 1980s as a custom computer-based replacement for an old supply system. However, planning for the system was overtaken by rapid changes in computing and communications technology and the system was redirected to accommodate a burgeoning number of interfaces with related systems. The resulting "stovepipe" system proved expensive to sustain, and a few years later the direction changed again. This time the use of a particular COTS enterprise resource planning (ERP) product was mandated without being evaluated. The selected product was incompatible with the system design and the project was ultimately cancelled.

In another system, the program office selected a product proposed by four out of seven integration bidders, assuming that it meant the product represented the best solution.[1] No independent evaluation was performed. Many problems came to light after the system was installed. For example, end users were required to manually enter data that the legacy system handled automatically, and required capabilities had to be added to the product. After a few years of trying to make the product fit, the program abandoned the effort and COTS products in general and started all over again with a largely custom system.

1.2.1 Common Evaluation Mistakes

The mistakes shown in Table 1 were found to occur again and again in many of the projects we studied. It is not easy to prioritize these mistakes, since all are potentially "project killers."

Table 1: Common Evaluation Mistakes

Mistake	Description
Inadequate level of effort	For example, critical products are selected using only an informal Internet search.
"Once and done"	New product releases or changes in the system context are not reevaluated.
Non-contextual	Consumer reports or "best of breed" lists are used as the only source of information.
Limited stakeholder involvement	Requirements are defined by engineers without consulting end users.
No hands-on experimentation	Marketing data are accepted as facts without further checking.

By applying the principles and recommendations identified in this report, organizations can avoid these (and other) pitfalls and be more successful in their COTS product evaluations and their COTS-based system (CBS) development efforts. To be successful, evaluators need to

- understand the impact of COTS software products on the system development process
- determine evaluation requirements for COTS software
- develop COTS software evaluation criteria
- select COTS software evaluation techniques
- employ a COTS software evaluation process that addresses the inherent tradeoffs

[1] Among the reasons that eventually came to light was that the bidders knew how much fourth generation language (4GL) code would need to be written and saw an opportunity for a large amount of paid work.

1.3 What Makes Evaluations Difficult

Evaluation means different things to different people, and organizations use COTS product evaluations in different ways. For example, the type of evaluation carried out by an accountant concerned with cost will be different from one carried out by an engineer concerned with system architecture and design. Some evaluations of COTS software products are carried out by parent organizations to identify products that can be endorsed for use by their subsidiaries, while other evaluations are carried out solely to gain a feel for the marketplace.

This report focuses on COTS product evaluations conducted for the purpose of selecting products to meet a known need in a system (that is, to select products for use). To determine if a product is fit for use in a system, ask such questions as

- Does it provide adequate functionality?
- Is it capable of operating with other components?
- Can it be adapted to satisfy my needs?
- Is it appropriate for my business strategy?

Evaluation is a complex operation in which individual products are not evaluated in isolation, but in the context of intended use. Part of the evaluation must be conducted in concert with other system components, and possibly at the same time as evaluations of other COTS products that are being considered for use in the system. Determining their fitness for use involves more than just meeting technical criteria. Criteria can also include the fitness of the vendor (such as reputation and financial health), the technological direction of the marketplace, the expectations placed on support staff, and a wide range of other concerns.

Evaluation is inevitably difficult. Often, COTS products must be chosen before the system architecture is defined and before requirements are completely understood. Added to the difficulty of understanding the system is the inevitable conflict of interest between stakeholders. For example, while the end users typically want as much capability as they can get, software designers and engineers often prefer a simpler solution. Fortunately, evaluation helps to resolve conflicting interests while simultaneously leading to more complete system understanding.

Another complexity involves identifying and considering the interactions (both favorable and unfavorable) between products. Products often provide functions that overlap with those provided by other products (such as data storage and version management). Overlapping functions can lead to conflicts when a product attempts to perform a task while other products are performing that same task. Normally, you cannot see how COTS products are engineered, which limits your ability to understand the assumptions embedded within the products. If these assumptions are not discovered by evaluators and engineers, they can prove disastrous in systems.

Unfortunately, many of the most important attributes of a product are subject to assumptions that are difficult to assess.

Evaluations are also limited by how quickly COTS products change. Many vendors release new versions of products more than once per year. If you are fortunate, new releases will not invalidate your plans for using the product. However, even with the best vendors and most stable products, unanticipated technology advances can affect product viability (consider the rapid impact of browser technology on the user interface).

1.3.1 COTS Product Evaluation in the COTS-Based System Context

Successful COTS product evaluation is part of a larger process of architecting, designing, building, and sustaining a COTS-based system. This section places COTS product evaluation within the context of a larger process for developing COTS-based systems defined by the Carnegie Mellon® Software Engineering Institute (SEI). The COTS-based system process presented here is only one of many possible. You do not need to follow this particular contextual process to make effective use of the COTS product evaluation process presented in this report. You can find details of a process framework in the SEI technical report *Basics for an Assembly Process for COTS-Based Systems (APCS)* [Carney 03]. An instance of the APCS process framework can be found in the technical report *Evolutionary Process for Integrating COTS-Based Systems (EPIC)* [Albert 02].

The Fundamental Change

If you try to follow the traditional custom-development approach when you implement a COTS-based system, you are likely to encounter the following scenario:

- You define your requirements (taking into account your system context).
- You or your contractor defines an architecture and design to satisfy the defined requirements.
- You or your contractor explores the marketplace to find products that will provide the functionality needed and fit within the defined architecture.
- You find no appropriate COTS products.

The fundamental change necessary with a COTS-based systems approach is the simultaneous exploration of the system context, potential architectures and designs, and available products in the marketplace (see Figure 2). “System context” represents requirements (functional and non-functional), end-user processes, business drivers, operational environment, constraints, policy,

® Carnegie Mellon is registered in the U.S. Patent and Trademark office.

cost, and schedule. "Architecture[2] and design" represents the software and system architecture and design, and "marketplace" represents available and emerging COTS technology and products, non-developmental items (NDI), and relevant standards.

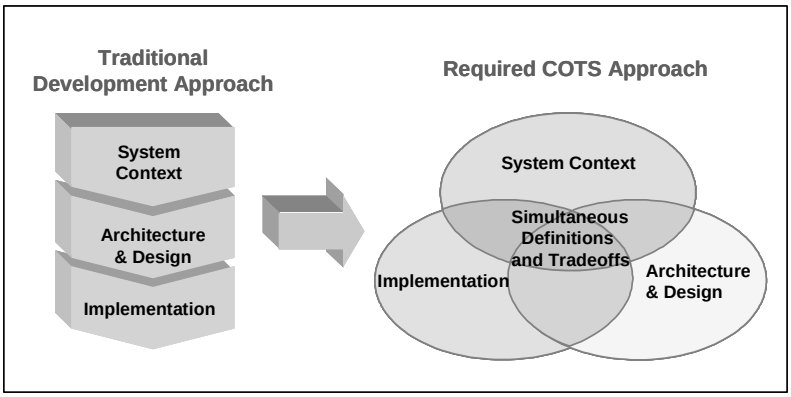


Figure 2: The Fundamental Change

A COTS-based approach requires a carefully reasoned selection of alternatives from among the various options and tradeoffs. Engineering activities must support this concept, and its effects permeate everything you do. As a result, the acquisition strategy, contractual activities, and business activities must all support this approach too.

The conceptual COTS-based approach developed by the SEI is iterative. Typically you go through the process below several times, gathering more information and eliminating or adding alternatives each time, until a viable solution remains. The steps in the process are as follows:

1. Assess and plan. This step includes developing effort estimations, setting goals, identifying stakeholders, and other typical planning activities. At the end of each iteration, accomplishments are assessed to set the goals for the next iteration.
2. Gather information. This step includes defining requirements, learning about COTS products, and understanding design constraints and risks.
3. Analyze. This step includes considering the entire body of knowledge that has been gathered about the products and the system, noting emerging compatibilities and identifying conflicts.
4. Negotiate. This step includes reaching an accepted understanding among the various parties involved in the system by resolving divergent expectations and points of contention.

[2] Architecture is the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time [Garlan 95].

5. Construct. This step includes implementing the selected solution for the current iteration. This can be any sort of model, partial implementation, or implementation subject to further analysis.

Figure 3 places COTS product evaluation within the context of this conceptual CBS process.

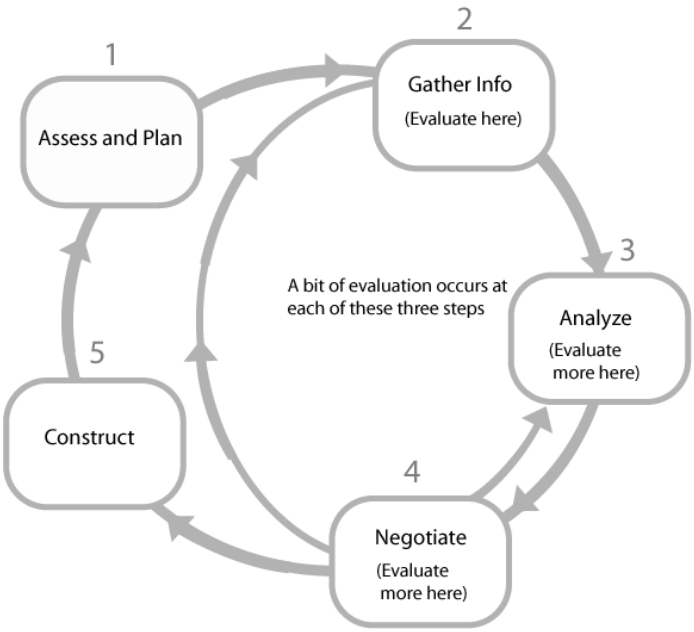


Figure 3: A Good CBS Process

This process is highly iterative. Expect to make multiple passes prior to defining the system to be delivered and even after system delivery. COTS product evaluation plays an important role in several steps, most obviously the "gather information" step. However, evaluation is often necessary during the "analyze" step to support understanding of alternatives and during the "negotiate" step to respond to stakeholder concerns regarding the alternatives. The overall process—and COTS product evaluation—continues for the entire system life cycle. A good evaluation is necessary but not sufficient for COTS-based systems success!

Building a COTS-based system is like completing a puzzle. Filling each hole in the puzzle requires an evaluation. The shape and size of the remaining hole changes each time a piece is added, and the choice of appropriate pieces becomes more constrained as pieces are selected. Finding the right pieces (products) to build a COTS-based system, however, is more difficult because the puzzle (the overall system) has no definite form or single solution. The shapes of the

required puzzle pieces (the needed product characteristics) also change as your understanding of the system changes. The changing nature of the COTS marketplace adds another complication: the puzzle pieces, and therefore the holes themselves, change over time.

COTS product evaluation is performed at many points in a CBS project. When a project is first starting, there are many unknowns and few constraints and commitments. At this early stage, the primary goal of evaluation is to discover information about a class of products and relate that information to the system at hand. However, as the project progresses, more decisions are made that constrain product selections. At later stages of a project, evaluation tends to involve matching products against well-known objectives. This journey from many unknowns to many commitments continues even after the system has been built.

1.3.2 Strategies for Effective Evaluation

There are six strategies that are often employed in successful evaluations.

Effective Evaluators Consider the Context

Because each system is unique, it is not sufficient to select a COTS product simply because it appears on a pre-approved list of acceptable products, is in someone's opinion the best product, or worked well in the context of another system. The more closely the evaluation reflects the actual usage of a product, the more accurate it will be.

The context of an evaluation encompasses the conditions under which evaluation of a COTS product is carried out. The results of an evaluation carried out in an environment that models the system will be more trustworthy than the results of an evaluation that involves a paper exercise or faith in a product because it is generally considered to be the “best of breed.”

Remember that other COTS products form part of the system context. While this is particularly important when products are intended to work closely together, it also can be important when products are intended to work independently. As you perform evaluations and pick products and other system components, your understanding of the system context will change. By selecting one COTS product, you are sometimes making other important product decisions, because the selected product works best with (or works only with) certain other products. The set of products that are used together is called an *ensemble*. An ensemble should be evaluated as a unit.

Effective Evaluators Account for Uncertainty

Regardless of how cautiously a COTS product evaluation is completed, the potential for error still exists. The system context might not be completely understood when the evaluation begins,

or the criteria and data gathering techniques used might be faulty. Since the potential for error cannot be eliminated, the evaluation must accommodate uncertainty.

The use of a traditional risk management approach, applied to the COTS-based system evaluation, can help manage such uncertainty. Possible risks must be documented and mitigation strategies outlined as appropriate.

Effective Evaluators Conduct Comprehensive Evaluations

Evaluating the functional capabilities of a product is not enough. Other factors to evaluate include business aspects, vendor qualifications, quality attributes, and integration compatibility. Just as it is unwise to buy a house based solely on price or square feet, it is also unwise to select a COTS product based on a single characteristic or an overly limited number of characteristics. On the other hand, it is impractical to evaluate all the possible qualities and characteristics of complex COTS products, so a sufficient but practical set of criteria must be defined.

Defining broad and sufficient criteria, however, does not help in meeting the goal unless enough resources are committed to measure products against those criteria. Many projects fall short of their goals because they do not expend enough effort in evaluation.

Effective Evaluators Know That Evaluation Is Continuous

Evaluation is not “once and done.” Technologies advance and change. Products are upgraded. Requirements change. Most engineers know that expectations placed on systems change over time—what engineer has not bemoaned changing requirements? These factors make COTS product evaluation an important activity at many points in the system life cycle.

The back and forth nature of continuing evaluations may seem tedious for some people. If something is done right the first time, why should it be repeated? The answer is that if everything is known from the start, then the evaluation can be done right the first time. If there are uncertainties at the start (as is almost always the case with COTS products and COTS-based systems), then it is very difficult to get all the necessary information on the first attempt. Repeating tasks as necessary is not a waste of time, since each repeated step increases the evaluators’ understanding of the product, the system, and the evaluation activity, moving them closer to their goal.

Effective Evaluators Are Guided by Facts

Opinions and impressions are useful but not sufficient. Far too often a COTS product evaluation is a mixture of unverified assumptions and hunches, which is completely unacceptable in any other engineering discipline.

The only way to achieve predictability in selecting COTS products is to base decisions on verified facts. This means that evaluations must be facts-centric, as opposed to "marketing-data-centric" or "my-hunch-centric." In the words of W. Edwards Deming, "In God we trust, all others must bring data."

Effective Evaluators Follow a Defined Process

Whether the task is frying eggs, building ships, or evaluating COTS products, consistently good results can be achieved only by following a consistent process. This does not mean that product evaluations require a highly complex, exquisitely documented process (although sometimes they do), but it does mean that without some kind of consistent process, evaluation results will vary.

A good evaluation process should be

- cost-effective, yielding maximum results with a reasonable amount of effort
- iterative, to address the evolving nature of the COTS marketplace
- inclusive of multiple criteria, so different aspects of products and systems can be addressed
- simple, to be attractive to users

1.4 The PECA Process

An evaluation process defined by the SEI and National Research Council Canada (NRC), called PECA (Plan, Establish, Collect, Analyze), helps organizations make carefully reasoned and sound product decisions. The process can be tailored by each organization to fit its particular needs, and is flexible enough to be used within many organizations and with many COTS-based development processes.

Although the PECA process was derived in part from ISO 14598 [ISO 99], the process was freely adapted to fit the needs of COTS software product evaluation. The process begins with initial planning for an evaluation of a COTS product (or products) and concludes with a recommendation to the decision maker. The decision itself is not considered part of the evaluation process—the aim of the process is to provide all of the information necessary for a decision to be made.

PECA was named for the four main activities that make up the process:

- **P**lanning the evaluation
- **E**stablishing the criteria
- **C**ollecting the data
- **A**nalyzing the data

As illustrated in Figure 4, the elements in the PECA process are not always executed sequentially. Evaluation events, such as a need for new criteria to distinguish products, unexpected discoveries that lead to the start of a new iteration, or inadequacy of collected data, will direct the process flow. One of the hallmarks of the PECA process is this flexibility to accommodate the realities of COTS-based systems. The process is flexible enough to be used within many COTS-based development processes, but is particularly suited to the process described on page 7. The steps of the PECA process are explained in detail in Sections 2, 3, 4, and 5, and a summary of the PECA process can be found in Appendix A.

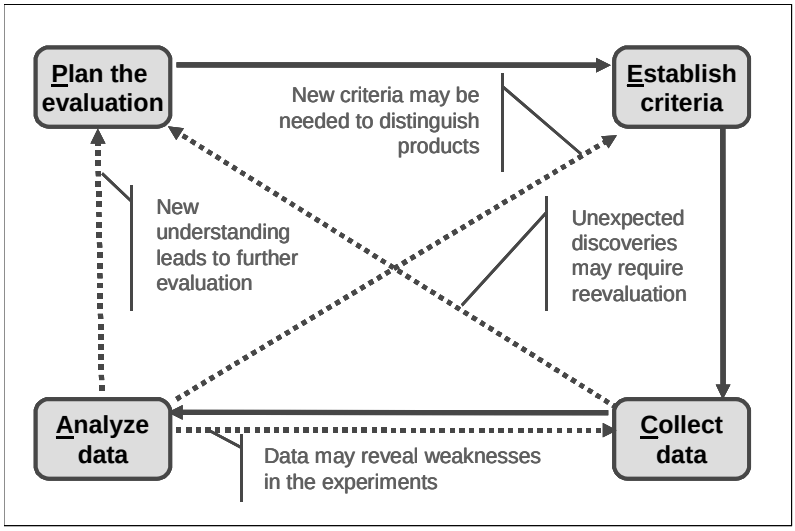


Figure 4: PECA – A Recommended Process

1.4.1 Evaluation Inputs

Two primary inputs to the process are the set of products that will be considered and the system requirements that must be met. However, system requirements alone are not sufficient for making an appropriate choice from among the products because they do not include many important characteristics of COTS products and vendors, such as underlying technology, quality of reputation, and support services offered.

Also, stakeholders may have expectations and assumptions that are not fully documented in the system requirements. For example, users often expect to interact with a product in a certain way, and developers expect to be able to tailor a product in a certain way. Reengineered end-user processes might not be represented in the original requirements either. Almost inevitably, COTS software solutions require particular business processes that affect the way end-users work. For

example, using ERP packages has huge process implications. While highly tailorable, these packages inevitably impose new processes and constrain the existing processes of end users. The processes actually used by people in an organization are rarely the ones found in available documentation, so it will probably be necessary to find out what the users are really doing and why. For more information about business process reengineering—the analysis and redesign of workflow within and between enterprises—see the Electronic College of Process Innovation [DoD 04].

Evaluation Guide

The evaluation guide, which is actually both an input and an output, is where an organization captures its "know how" about performing evaluations. It describes how to perform an evaluation and includes descriptions of processes and techniques. Each COTS product evaluation adds to the organization's knowledge about evaluation, and new information should be recorded in the evaluation guide.

1.4.2 Evaluation Outputs

The type and detail of required documentation depend on the goals of the evaluation. The main outputs of the PECA process are the product dossier, the evaluation record, and the summary with recommendations, although all evaluations do not require all of these documents. These documents are described briefly below; more detailed information can be found in Section 5.4.1, "Documenting Recommendations."

- Product dossier. This document acts as a repository of software documentation and includes discovered facts, assessment results, classifications, and other explanatory material about a specific product. (See also Appendix B.)
- Evaluation record. This record contains a description of the evaluation process. It provides information about the steps used, including effort expended, and lists evaluators' references and expertise. (See also Appendix C.)
- Summary/recommendation. This document includes a summary of the evaluation activity and the resulting findings, along with any messages the evaluation team wants to transmit to the decision maker.

Beneficial Side Effects of Evaluation

Documents are not the only result of evaluations, and might not even be the most important results. During the evaluation, the evaluation team gains deep insights into the system impacts of using COTS products. Ideally, this insight will have a significant impact on perceptions about the system. For example, unrealistic expectations held by stakeholders should be detected, clarified, and negotiated. Requirement and system engineers should use this insight to refine the sys-

tem requirements. Finally, architects and integrators should apply this insight to correct design flaws and adapt the system to work in a COTS product world.

Improving Future Evaluations

Gathering data about successes and failures during a COTS product evaluation process is the best way to improve future evaluations.

- Information about cost and schedule breakdowns can improve future estimations.
- Measures of the effectiveness of criteria can aid in the generation of better criteria.
- Measures of the effectiveness of techniques for evaluating criteria can aid in the development of better evaluation approaches.

1.4.3 Introduction to Evaluation Techniques

While a good process is imperative for the consistent evaluation of COTS products, a successful evaluation relies on more than just a process. In addition, a set of techniques is needed to complete each step in the evaluation. While the evaluation process is a high-level description of what we need to do, techniques are low-level descriptions of how to do it.

Some steps in the PECA process require techniques that might be familiar to you. For example, requirements elicitation is a technique used to draw out stakeholders' expectations, and Gantt charts are a technique used for scheduling [Modell 96]. Other activities require less well-known techniques: scenario-based evaluation can be used for data collection, and the analytical hierarchical process (AHP) can be used for consolidation and analysis.

This report does not include all the techniques that exist, but instead focuses on lesser-known techniques that are more specific to COTS product evaluation. Techniques are included with the step at which we think they will be most useful, although many of the techniques can be used in other steps of the evaluation process as well.

2 Planning the Evaluation

The first step in the PECA process is planning the evaluation. Planning for each COTS product evaluation is different because the evaluation may involve different types of products (from simple to extremely complex) and different system expectations placed on the product (from trivial to highly demanding). Tasks that might be completed during this step include forming the evaluation team, creating a charter, identifying stakeholders, and picking the approach.

2.1 Forming Evaluation Teams

The importance of an effective evaluation team should not be underestimated, since without an effective team, an evaluation cannot succeed. We have unfortunately seen situations where the most junior engineer, with little support from anyone else, was assigned to evaluate products. A lone engineer does not usually have the range of skills necessary to perform a broad-based COTS product evaluation. A good balance of power is also important for a good team. In one situation we observed, the most senior employee (the company president) sat on the evaluation team and biased the results toward his or her preference.

An evaluation team should include people with diverse skills, such as

- technical experts
- domain experts
- contracts personnel
- business analysts
- security professionals
- maintenance staff
- end users

2.2 Creating the Charter

Some very general questions should be answered before an evaluation task begins. These questions include

- What is the evaluation expected to achieve?

- What are the attributes of the evaluation team? Do they have decision authority?
- What are the responsibilities of each member of the team?
- In what ways is the system context flexible?
- How will the evaluation team measure success?
- When should the evaluation finish?
- What constraints must the evaluation team adhere to?

Often, this basic information is not documented, even though the slightest misunderstanding can doom a project. Moreover, people tend to be more productive when they have a clear picture of the ultimate goals of the project. A charter, created by the evaluation team to define the scope and constraints of the evaluation, can record the answers to these questions. The charter should contain

- goals of the evaluation
- scope of the evaluation
- names of team members and their roles
- statement of commitment from both evaluators and management
- summary of factors that limit selection
- summary of decisions that have already been made

2.2.1 Creating the Charter: Example

An example of a simple charter is shown in Table 2. A family might create a charter like this when selecting a new camcorder.

Table 2: Charter Example

Scope of evaluation	The ten best-selling digital non-professional camcorders.
Statement of evaluation goals	Select by consensus a camcorder within budget that fulfills stated needs.
Team members and roles	Sally and Bob are evaluators and decision makers.
Explicit statement of commitment	The new camcorder will be purchased before the birth of the family's new baby.
Summary of factors that limit selection	Budget = $800. Time frame = 9 weeks.
Summary of decisions that have already been made	Camcorder must be digital, miniDVD, and pocket size.

2.3 Identifying Stakeholders

Evaluation stakeholders are individuals or groups with a vested interest in the results of a COTS product evaluation, including all individuals or groups that will be affected by the selection of a particular COTS product in a significant way.

While there are no rules for identifying evaluation stakeholders, errors of inclusion (including additional individuals or groups) are less risky than errors of exclusion (omitting legitimate stakeholders), since errors of exclusion can lead to rejection of the system. Errors of inclusion, however, may make it harder to identify an appropriate COTS product because more people must agree. Practical experience suggests that the main stakeholder group should be limited to approximately seven or eight individuals. If there is a larger number of stakeholders, multiple sessions and management of various groups may be necessary.

Potential stakeholders include

- end users
- system administrators
- operators
- maintainers
- integrators
- architects
- sponsors
- program managers
- vendors

The stakeholders for a particular evaluation might not be a subset of the stakeholders identified for the entire system, since the concerns and expectations relevant to a specific component, COTS product, and vendor are often different from the concerns and documented expectations for the whole system.

2.4 Picking the Approach

During the planning stage, the basic characteristics of the evaluation activity are determined. Some of the parameters of the approach selected include the depth or rigor of the evaluation, the basic strategy for selection, and the filters needed to reduce the number of candidate products.

2.4.1 Depth of the Evaluation

Different situations demand deeper or more superficial evaluations. Some situations require an extremely rigorous evaluation of COTS products; in other situations, an inappropriate product selection carries little risk. Two factors help determine the depth:

- complexity of the evaluation. How likely is it the wrong product will be selected?
- risk of failure. What would be the system impact of making the wrong selection? For example, would it involve significant financial or environmental damage?

If the COTS product being considered will have relatively low technical risk for the system and little involvement with strategic or critical end-user processes, the focus of the evaluation can be on cost and the speed of implementation. A set of low-impact COTS product evaluation criteria can be established and reused (with appropriate tailoring) for all of an organization's COTS products and systems that share these characteristics. The product can then be selected based on an *informed decision*—for example, a decision made after conducting unstructured product research, such as reading marketing brochures.

If the COTS product will have a pervasive impact on the system or support strategic or critical business processes, it should be chosen using a *methodic selection* process, which is preferred in most situations. The PECA process is a good example of a methodic selection process.

If the COTS product could cause the entire business to fail if not executed properly, then a *mathematically formal methodology* should theoretically be used to select the product. In reality, however, this approach is viable only in a limited number of situations and is difficult to use.

2.4.2 First Fit vs. Best Fit

You must also decide whether you will choose the first product you evaluate that satisfies your needs or if you will consider the entire group of products before making a selection.

First fit can be applied at any level of evaluation rigor and should be used when the goal is simply to fulfill a minimum set of needs—when additional "goodness" of a product is unimportant or it is not cost effective to evaluate to determine additional goodness. First fit considers minimum requirements and answers the question "Is it good enough?" This does not imply that the criteria by which products are assessed are any less complete. Although the first adequate candidate is selected without comparison to other candidates' capabilities, all requirements must still be met.

Best fit should be used when there is an appreciable gain in getting more than the minimal amount of some characteristic. For example, in many situations a minimum performance is specified, but better performance adds significant value to a product within the context of the system. Best fit is also the approach to use in situations where no product has all the capabilities being sought in a particular evaluation. Best fit answers the question “How good is it?”

2.4.3 Using Filters

Sometimes it is not reasonable to perform a thorough evaluation on all candidate products because the number of candidates is too large. The goal of filtering is to narrow the range of products that must be further evaluated by developing progressively more selective filters. By starting with a coarse filter that is easy and cheap to use, products that are a poor fit can be eliminated quickly. The remaining products can be evaluated using increasingly discriminating filters. Using this process, the most complex and expensive filters will be used on the smallest number of products.

Filters that focus on easily measured criteria can provide a useful mechanism for winnowing the list of products under consideration. Product price is a coarse filter focusing on easily measured criteria. Market share is another. As the number of candidates to consider is reduced, filters may become increasingly discriminating and complex. A complex filter may involve building a test harness for multiple products to determine which products respond to erroneous inputs in the manner most appropriate for the system.

2.4.4 Picking the Approach: Example

In the example introduced earlier of a camcorder purchase, some issues that might influence the depth of the evaluation include the following:

- A camcorder is an important, long-term investment.
- Buying a “lemon” has an important risk associated: the family might stop using the camcorder, and the investment would be lost.
- The family has no particular expertise in camcorders.

Because of these concerns, the camcorder should be chosen using a methodical selection process. Although a camcorder could easily be selected at random from a store display, a camcorder is a large, important investment, which makes its selection an important decision that requires careful evaluation.

The approach chosen should follow these steps:

- Collect information about features, brands, and models (market survey, discovery mode).

- Reduce the number of camcorders to three or four (filtering).
- Try each of those camcorders to find the best (best fit).

2.5 Estimating Resources and Schedule

Expert opinion, analogy, decomposition, and cost modeling are classic and well understood techniques used for cost evaluation in engineering disciplines. Unfortunately, there are few techniques available specifically for estimating resources and schedules for COTS product evaluations. The Constructive COTS Model (COCOTS) is one of the few attempts to specifically address the costs associated with building a COTS-based system [Abts 97]. The technique does not separately address the costs associated with COTS product evaluation, however. An earned value management approach can be used in an evaluation context; an example is given in *Using EVMS with COTS-Based Systems* [Staley 01].

Some factors that may affect cost estimation include the following:

- level of rigor and risk involved. In general, the more rigorous the evaluation, the greater the short-term cost. However, rigorous evaluations may lower long-term costs in building the system because data obtained during the evaluation effort is useful during system architecting, design, and integration. In addition, a rigorous evaluation may head off the greatest cost of all: a failed system.
- number of candidates being evaluated. The more candidates evaluated, the higher the overall cost.
- evaluators' levels of experience. Evaluation costs are often higher when evaluations are performed by more experienced evaluators because they tend to perform more rigorous evaluations. However, use of experienced evaluators can be expected to reduce costs down the road.

In general, the resources commonly allocated to COTS product evaluation are insufficient. This is particularly the case for COTS products that are at the core of complex systems.

3 Establishing Criteria

Evaluation criteria are the facts or standards by which the fitness of products is judged. Perhaps the most critical step in the entire COTS evaluation process is establishing appropriate criteria. The criteria selected will determine whether you ask the right questions regarding the viability of the product in your system. This chapter provides a simple three-step process, recommendations, and techniques for transforming a set of requirements into a set of product evaluation criteria that can be applied in a COTS product evaluation and selection process. The three steps are

1. Define evaluation requirements.
2. Define evaluation criteria.
3. Prioritize criteria.

These steps are defined in detail below.

3.1 Requirements vs. Criteria

The starting point for virtually all software development efforts is a set of requirements that represent at some level the expectations of those that have a stake in the system. If the organization is considering the use of COTS products as system components, it is natural and appropriate to look first to these requirements as a basis for evaluating and selecting the right products. Unfortunately, for some organizations, an initial focus on system requirements leads them to believe that generating the actual criteria for COTS evaluation is trivial—each requirement is directly transformed into a criterion. However, our experience suggests that this simple transformation is not likely to achieve the desired result—selection of an appropriate COTS product—because of the following:

- System requirements are normally written at an abstract level in order to allow sufficient flexibility for choosing multiple technical solutions. Unfortunately, criteria derived directly from such requirements are too abstract to serve as a way of evaluating products. For example, a requirement such as “The system shall be easy to use” is too abstract to judge whether a product is or is not easy to use. A less abstract and more useful requirement might be “The system follows the Windows interface standard” or “The learning curve for the system shall not exceed two weeks.”
- System requirements are stated in terms of needs, whereas criteria should be stated in terms of capabilities to satisfy those needs. For example, a requirement such as “Information transfer shall be protected from unauthorized access” might be transformed into a criterion such

as “Support for secure sockets or equivalent security mechanism,” which is the expected concrete capability.

- Criteria should be obviously measurable, whereas system requirements are often stated in a manner such that it is not immediately obvious how to measure them. Criteria should include the way in which the expected capability is to be determined in order to facilitate the evaluation process. Such a criterion might be “System will accurately process 500 transactions an hour. This will be verified using a performance testing tool.”
- System requirements tend to be incomplete, hardly ever stating every expectation placed on a COTS product. Often qualities of the component other than required functionality are overlooked. For example, a system requirement regarding the level of effort necessary to apply appropriate tailoring to new releases of a product is rare.

3.2 Defining Evaluation Requirements

Because the expectations placed on a COTS product are not just a subset of the system requirements, the first step in defining evaluation criteria is to establish *evaluation requirements*. Because COTS products rarely align exactly with anticipated system functions (some products combine functions, use different vocabularies, etc.), the mapping between product functions and system functional requirements may not be obvious or straightforward. In addition, COTS products add an entirely new class of concerns regarding licenses, testing, rapid deployment, and control of the content of upgrades that are often not addressed by system requirements.

3.2.1 Sources of Evaluation Requirements

There are many sources of legitimate evaluation requirements that are derived from the context in which a COTS product will execute but are not always addressed by system requirements. These sources include

- Architecture/interface constraints: COTS product decisions are often constrained by other decisions that have already been made. These constraints become evaluation requirements when choosing a COTS product. For example, if a decision has been made to use a certain middleware mechanism, it makes little sense to buy a product that is clearly going to conflict with this technology.
- Programmatic constraints: Time, money, available expertise, and many other programmatic factors may be sources of evaluation requirements that are not captured in system functional requirements. For example, availability of trained staff adept at using a product can be a useful criterion when choosing products, but is not likely to appear in system requirements.
- Operational environment: Not all aspects of the operational environment are included as system requirements. For example, information about the organization that will perform maintenance on the system is frequently omitted. Thus, the operational environment may be a source of additional evaluation requirements.
- Stakeholder expectations: People place additional expectations on products that are not clear from system requirements. For example, users may have a strong preference about the style of the user interface to a product. These expectations are often not captured in system re-

quirements, in part because it is assumed that the user interface can be tailored to expectations. However, the user interface of a COTS product is not as flexible as that provided by custom code. Therefore, it is more important to determine the acceptability of the interface.

All of the previously listed sources are specific to the system under construction. There are also a number of sources that provide lists of potential evaluation requirements that are not system specific.[3] These sources can provide insight into a diverse range of COTS product and vendor characteristics, but no requirement identified from these sources should be used without careful consideration of relevance to the situation at hand. Sources of non-specific requirements include

- product feature checklists
- organizational checklists
- previous evaluations for other systems
- marketplace

Product feature checklists are a standard fare for product comparisons. While they are only as reliable as the source, they are widely used—and misused—in COTS software evaluation. The basic capability provided by a product feature checklist is an itemized list of the features of a class of products. This list is often used as a basis for comparing different products.

In some market segments, organizations have formed a niche market by identifying a set of generally preferred features and characteristics and evaluating products against this set. The most notable example of this approach is *Consumer Reports*, which monthly evaluates consumer items and provides recommendations for preferred products [Consumer 00]. However, in the COTS software domain, the context in which the software will be used is not nearly as consistent as the various contexts for which *Consumer Reports* makes recommendations. Thus, an effective *Consumer Reports* for the COTS product marketplace does not exist today and is not likely in the near term.

Some firms, however, regularly assess products that make up particular market segments; examples of these are Gartner and Ovum [Gartner 98, Ovum 00]. In addition, trade magazines often publish product feature checklists. One should guard against accepting at face value product feature checklists that appear in trade magazines. These are sometimes produced by vendors specifically to present their product in the best light or to highlight some feature that distinguishes their product from those of competitors. It is also important to keep in mind that no generic list can reflect all of the requirements that need to be considered—a generic list will include some features that are irrelevant for the system and other important features will be missed. These lists should not be used as the only evaluation requirements for COTS products.

[3] ISO 14598 includes the concept of "evaluation modules" that are reusable packages containing requirements and other data about an evaluation. In theory, these modules can be reused in different settings. This approach is similar to the reuse of evaluation requirements described here.

Other sources of "reusable" evaluation requirements are organizational checklists that represent a consistent way to ensure that corporate interests are addressed in the evaluation and selection of COTS products. Organization checklists may maintain relevant information about corporate policies, preferred architectures, and expected engineering practices and can provide some uniformity and predictability in selecting products.

In general, corporate need for such checklists is not driven solely by technical considerations: many other concerns are reflected as well, such as legal considerations (e.g., "due diligence" in managing financial assets) and issues of scale (many organizations, many projects, many evaluators). Table 3 identifies the categories of criteria covered in an organizational checklist built by a large aerospace contractor. Appendix E provides a larger compendium of several lists we've encountered. In addition to the requirements covered in these categories, the actual checklist provides a 70-step process for evaluation and selection.

Table 3: Example of an Organizational Checklist

Organizational Checklist for COTS Products
Compatibility with other COTS products in use
Adaptability, flexibility, reliability, maintainability
Impact on system integrity
Impact on system integration
Vendor support
Training
Documentation
Licenses

Evaluations that the members of the team (or the wider organization) have performed for similar purposes are also sources of evaluation requirements. While these requirements are likely to reflect some of the context-specific nature required, it is a mistake to consider them equally appropriate for the new evaluation context as they were for the old. This mistake is reflected in statements such as: "We looked at Product X before and it was pretty good. It will be good for the new system also." A variant of this problem is represented by selecting "pre-approved" products from a product list without further analysis. In essence, such choices indicate that the requirements for COTS products in "this" system are sufficiently close to that of a "benchmark" system and therefore no product evaluation process is warranted—a very bold and risky position.

Finally, a source for evaluation requirements is the marketplace itself. Risk-Driven Generation is a technique that looks at what the COTS marketplace can offer and focuses on the risk intro-

duced by or reduced by product and vendor features[4] [Wallnau 01]. This approach derives a set of potential evaluation requirements from the features of products rather than from system requirements. This set of requirements is pruned by analyzing the risk to the system should the product provide or not provide a particular feature. The steps involved in Risk-Driven Generation are straightforward:

1. Identify interesting product features.
2. Assert risk to the system of a product not exhibiting a feature.
3. Categorize and quantify risk (optional).
4. Identify mitigations (optional).

The overall effect is that expectations regarding system needs are "adjusted" to agree with product capabilities. By focusing on risk, the approach also helps to combat "gold plating" of requirements and "featuritis." If little or no risk results from absence of a product feature, then the feature is not required of products under consideration. Note that this approach may also generate negative requirements—features that the product should *not* have.

3.2.2 Classes of Evaluation Requirements

A problem when evaluating COTS products is to treat the majority of requirements as providing no leeway for negotiation. Stakeholders are the common source of this problem because they are most familiar and comfortable with the end-user processes, user interfaces, and capabilities provided by existing systems and have high expectations that the new system will provide all of these plus the latest "whizz-bang" technologies. This is almost inevitably a mistake because virtually all COTS products are compromises suited to multiple organizations and will very probably work in a way that is different from the way to which stakeholders have grown accustomed. In the most severe case, this approach will render all COTS products unacceptable, since none can meet the expectations and specifications[5] desired by users. If the organization is committed to using COTS products, it should strive for achieving sufficient flexibility in requirements.

In reality, identified evaluation requirements will fall in one of two general classes or categories:

1. *Hard requirements* that are mandatory—if the product does not meet these requirements, then the product or the system must be augmented in some way or the product is unsuitable. The augmentation could take the form of external custom code that handles functionality

[4] The Risk-Driven Generation approach identified in *Building Systems from Commercial Components* by Wallnau et al. refers to criteria. However, the use of the term *criteria* in the document is consistent with our use of the term *requirement.*

[5] In some cases, where organizations are dead set against the use of COTS products, this approach is often intentional.

the COTS product does not provide or the selection of another product that provides the missing capability.

2. *Negotiable requirements* that are flexible—if a product does not demonstrate the preferred characteristic, then it is not automatically excluded. The options in this case are to adjust the requirement to some degree or to tailor the product or the other components in the system to address the requirement.

We expect that the majority of requirements fall somewhere towards the center of a spectrum between totally inflexible and completely flexible, where there is some room for negotiation but in the end the requirement (or some version of it) must somehow be met. In many cases, the implementation expressed by the COTS product, though it may not completely match the original need, will be an acceptable requirement. This has even been shown in several cases involving requirements that were thought to be controlled by legislative mandate (laws). In these cases, the driving force behind the requirement was the "traditional" approach of manual processes or legacy systems, rather than the actual letter of the law. Several state and federal organizations have suggested to the SEI that many laws have more leeway than people who have grown accustomed to current procedures tend to believe. In all cases, the organization should carefully research any perceived legislative mandates to determine whether the letter of the law (rather than some traditional interpretation) is the limiting factor.

Errors Compiling Evaluation Requirements

There are two types of errors that can occur when compiling evaluation requirements: errors of inclusion and errors of exclusion. Errors of inclusion are caused when non-applicable requirements are included in the evaluation requirement set. Errors of inclusion have two unfortunate effects. First, they tend to reduce the total amount of time and effort that is focused on the evaluation of the product against necessary and important requirements. Thus, the organization may focus its evaluation efforts on determining whether a product supports a feature of questionable value, while spending less effort determining whether a product's vendor can appropriately support the product in a certain context. Second, errors of inclusion can eliminate suitable COTS products simply because they do not support the unnecessary and unimportant "requirement." Perhaps the best example of an error of inclusion is the tendency to want every "cool" capability—even if the capability does not add any appreciable value in the present context. A good technique to combat this tendency is to consider the risk to the system mission should the feature be absent or provided in a different manner (e.g., risk-driven requirements generation, as covered in Section 3.2.1, pages 24-25). If there is no risk or it is minimal, then the requirement may be unnecessary.

Errors of exclusion involve omitting evaluation requirements that are crucial to the fitness of the product and vendor within the system context. The main danger of errors of exclusion is that they can lead to the selection of an unsuitable product. For example, neglecting to consider the vendor's ability to support a product within a certain context may lead to the selection of an un-

supportable product. Errors of exclusion are often caused by insufficient understanding and an oversimplification of the problem. An iterative approach to building COTS-based systems can help mitigate this risk. As understanding about the problem grows, it is almost inevitable that requirements that were initially overlooked will be identified.

3.3 Defining Evaluation Criteria

Completed evaluation requirements represent what is needed from a COTS product. However, these requirements are not normally phrased in a manner that leads to straightforward and unambiguous analysis of products. Prior to beginning the actual evaluation of products, evaluation criteria must be generated from evaluation requirements. Evaluation criteria are the manifestation of the practical approach chosen for determining whether a COTS product has the characteristics that are required for the system.

3.3.1 Characteristics of Good Evaluation Criteria

A good criterion for evaluation and selection of a COTS product consists of two elements:

1. A clearly measurable statement of the capability necessary to satisfy a need—called a capability statement
2. A means for assessing and assigning a value to the product's level of compliance with the capability—called a measurement method

A good criterion needs both the capability statement and a measurement method. However, in some cases the details of the measurement method may not be fully known when the criterion is first defined. In this case, these details must be filled in as they become known. Table 4 contains an example of a fully defined criterion for the evaluation of vendor support.

Table 4: Example of a Requirement, Capability Statement, and Measurement Method

Requirement • The COTS vendor shall provide extensive support for the COTS product. Capability statement • COTS vendor support shall include - 24x7 help desk - On-site installation/training support - Online error reporting Measurement method • A product support survey will be provided to potential COTS vendors. Support claims will be verified by exercising help facilities where possible and by contacting current product users to determine the quality of vendor support.

Several additional characteristics should be considered when forming criteria:

- Assessability: If data cannot be gathered to support the measurement method selected, then the criterion is not good. For example, “quality of engineering” is not a good criterion if there is no way to gather data indicating whether the product is well architected, designed, and implemented.
- Ability to discriminate: If all COTS products display a required characteristic, then the characteristic is not useful in identifying which products are superior. For example, the presence of a graphical user interface will not normally discriminate between modern word processors.
- Non-overlapping: If criteria overlap, then the associated product characteristics can be factored into deliberations multiple times. This can lead to misleading results and possibly wasted effort.
- Significance: If a criterion is not valuable in the context of the system, then it should not be used. For example, the long-term stability of a vendor is irrelevant if the product will only be used as a short-term or interim solution.

3.3.2 Techniques for Defining Evaluation Criteria

Perhaps the most common approach to generating evaluation criteria is through straightforward decomposition of evaluation requirements. Each evaluation requirement is decomposed into one or more evaluation criteria until the performance of a COTS product can be directly and unambiguously measured.

There are a variety of sources for pre-existing decompositions of important software characteristics that potentially can serve as the basis for criteria. Among the more common are decompositions found in organizational checklists and in the software literature. Other good sources are the various ISO and IEEE standards that address particular concerns about software. For example, ISO provides a list of software quality characteristics (e.g., functionality, reliability, usability) and suggests a potential decomposition of the quality characteristics into constituent parts [ISO 91]. The standard also provides suggestions for measurement methods. Appendix D provides a summary of such characteristics.

Of course, no general list can reflect all of the criteria you need to consider—a general list will include some criteria that are irrelevant and will miss other criteria that are important. The *Goal Question Metric* (GQM) technique provides an approach that assures system specificity by decomposition of requirements into capability statements and associated measurement methods [Briand 94]. With GQM, the user develops a triad containing

1. the *goal*, which in this case is to fulfill an evaluation requirement
2. the *question*, which is a capability statement
3. the *metric*, which is the standard of measurement, determined either by direct measurement or decomposition into other GQM triads

GQM has been adopted by a number of product evaluation techniques, including PORE [Ncube 99] and OTSO [Kontio 96]. Table 5 provides an example of GQM.

Table 5: Example of the Goal Question Metric Technique

Goal (Requirement)	Minimum impact of learning curve on end-user performance.
Questions (Capability statement)	• Are the text and graphics visible? • Is there an intuitive interface? • Are the training materials effective?
Metrics (Measurement standard)	• Readability of data by user with aging eyesight • Reaction of user to a sample set of common tasks • Retention of information via a formal exam

3.3.3 Measurement: Common Problems

Several common problems must be addressed in defining measures for criteria. These include selecting the measurement method, minimizing bias and error, and normalizing measurement scales.

Selecting a Measurement Method

For any realistic COTS evaluation scenario, the need for context-specific and detailed information must be balanced against budget and schedule concerns. This problem is reflected in a variety of ways, but in almost all evaluation situations, a decision must be made between hands-on analysis of COTS products and analysis of various forms of documentation (either provided by the vendor or by a third party).

Hands-on inspection is by far the preferred way of determining what the software product really does (as opposed to what the vendor claims it does) and how well the COTS software will fit into the system. For all but the most rudimentary evaluations, we believe hands-on experiments should be performed and data gathered. However, hands-on inspection is far more difficult, time-consuming, and expensive than document inspection. It can involve procuring hardware as well as additional software and the construction of evaluation harnesses to mimic portions of the executing systems.

Another critical decision involves the type of data to be gathered. For a given capability to be measured, either qualitative measurements (such as *bad, fair, good*) or quantitative measurements (such as *2.34 miles, 4.64 nanoseconds*) might be appropriate. There is sometimes a ten-

dency to believe that quantitative measures are "better" than qualitative ones, but the reality is that both can provide useful information for a COTS decision. The choice of qualitative vs. quantitative depends on a range of factors, including the specific capability, its relative importance, and the resources available for its measurement.

Reducing Bias and Error

The goal of measurement is to produce fair, unbiased results and to frame those results in a way that supports the ability to reason about them. Unfortunately, even with the best criteria, measurement bias or error can affect the quality of the decisions made about COTS products.

One form of measurement bias is the result of unintended influences on measurement from factors outside product characteristics. For example, with qualitative measurement, individual evaluators may have preexisting bias against one vendor, causing them to view all characteristics of the vendor's products in a poor light. A more subtle case involves differences between individuals in interpreting qualitative scales. For example, to one individual, fair may be an acceptable rating for a product, while to another individual rating the same product, fair may mean the product is totally unacceptable.

One way to increase consistency across individuals rating products against qualitative criteria is to provide clear guidelines that

- emphasize the need to judge all criteria separately and without bias
- directly state what various qualitative ratings mean (e.g., *bad* is unacceptable, *fair* is marginally acceptable, and *good* is fully acceptable)

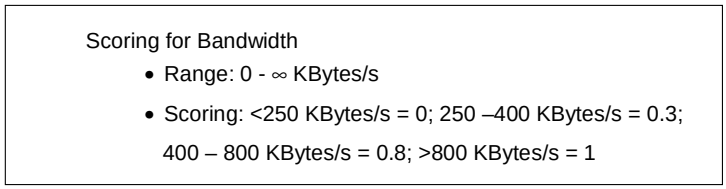
A second form of error results from quantitative measurements that are systematically or randomly biased or invalidated. For example, a common flaw emerges when a single run of a benchmark is performed and a slight variance in the timing of input data significantly alters the performance measured. This could lead to a false impression that one product is superior to another. In general, poor setup of the evaluation environment (e.g., the way in which the system/environment/product is configured) and poor selection of test data lead to this form of measurement error. This problem is normally addressed by developing consistent evaluation environments, sufficiently accurate test harnesses, and multiple test executions.

Managing Multiple Scales: Normalization

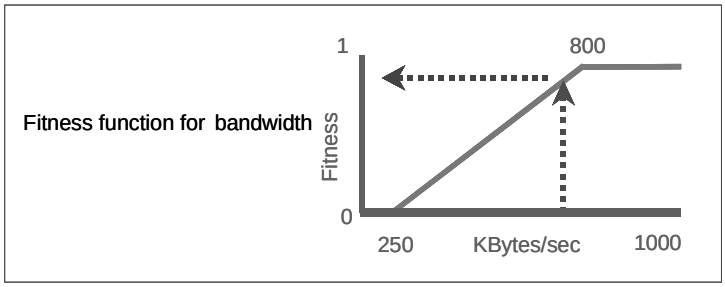
As already indicated, data in its "natural" form has different units of measurement and different scales. For example, the quality of a vendor's product support may be measured using terms such as poor, fair, good, or excellent, while the cost of the product is measured in dollars. However, in order to understand the total picture and compare products that perform at different lev-

els on different criteria, a consistent scale must be assigned. The process that converts all criteria measurements to the same scale (e.g., a common measure of fitness) is called *normalization.*

Figure 5 illustrates two approaches to converting continuous data to a normalized "fitness" scale. In the top part of the figure, various ranges within the continuous data field are converted to fitness values (e.g., 0, 0.3, 0.8, and 1). This means, for example, that products performing at 250 KBytes/sec and 400 KBytes/sec are identically fit (fitness = 0.3 for both products). In the bottom of the figure, there is a continuous relationship shown between performance and fitness. Thus, a product performing at 400 KBytes/s (kilobytes per second) is more fit than one performing at 250 KBytes/s. A product performing at greater than 800KBytes/sec receives the same maximum score.



A: Discrete Scoring Ranges



B: Continuous Scoring Ranges

Figure 5: Scoring Ranges and Functions

A common mistake organizations make in using normalized data involves violations of the basic premises of measurement scales[6] [Fenton 91]. Organizations sometimes try to convert ordinal data (that is, data in which only the order is significant) to interval data (that is, data where the interval between units is equivalent). For example, they convert price using a mapping similar to the following:

- More than $1200 is poor, which equates to 1.
- Less than or equal to $750 is fair, which equates to 2.

[6] Fenton provides an excellent discussion of this topic in particular and the fundamentals of software metrics in general.

- Less than or equal to $500 is good, which equates to 3.
- Less than or equal to $350 is excellent, which equates to 4.

While the mapping itself is perfectly valid as long as it is recognized that the new scale (1,2,3,4) is still ordinal, it is invalid to make assumptions that treat the result as interval data (e.g., in this case, assuming a rating of 4 is twice as good as 2). This commonly occurs when organizations try to perform mathematical operations on ordinal data that has been converted into a numeric representation. For example, evaluators may try to determine the *mean* of the data, but the operation is meaningless for ordinal data.

3.4 Establishing Priorities

Normally, some COTS product evaluation criteria (i.e., those derived from negotiable requirements) are more important than others. In order to reflect this relative importance, priorities must be assigned to criteria to allow reasoning about products that have different strengths and weaknesses.

Unfortunately, there is no simple formula for identifying the priority of a given criterion. The actual priority assigned tends to reflect a composite of many different factors. For example, the value assigned to a criterion may reflect its relevance when compared to others, the expense of meeting the criterion in another way if the COTS product does not provide a solution, and the risks to the system if the criterion goes unmet.

A common way of expressing priority is by assigning weights to the criteria. Weights express the relative importance of normalized values of different criteria, answering questions such as, "How many units of this criterion would we trade for a unit of this other criterion?"

3.4.1 Techniques for Weighting Criteria

A variety of techniques can be used to assign weights to criteria. All involve making a judgment based on understanding of the system. Three popular approaches include

- unstructured weighting. One or more people determine weights based on their common understanding of the system and their experience. This is probably the most popular method, but not necessarily the best.
- Delphi technique. Individuals use their own approach for deriving initial weights. The Delphi technique helps the team converge on a single weight and is a popular method for gaining team consensus [Dalkey 63, Linstone 75].
- pair-wise comparison. A judgment is performed by comparing pairs instead of whole sets of criteria. Criteria are ordered in pairs, and the team agrees on the relative importance of the criteria in each pair. Pair-wise comparison is at the core of the Analytic Hierarchy Process

(AHP) [Saaty 80]. Tools such as Expert Choice [EC 00] support AHP and pair-wise comparison by computing weights for each criterion from the total set of pair-wise comparisons.

The Delphi technique and Pair-wise comparison are described in more detail in the following sections.

Delphi Technique

In the Delphi technique, a questionnaire is given to several people, asking their informed opinions on the subject. Replies are tabulated, and the questionnaire is distributed again, this time with all the opinions attached to the questionnaire. People read each other's opinions and answer the questions again. This process might continue for three or four cycles [Linstone 75]. As information is exchanged, people incorporate each others' perspectives and information into their thinking and collectively arrive at a fairly accurate understanding of the critical issues to consider in their decision-making process. The following steps make up the basic method:

1. Clarify what information you need and why you need it.
2. Determine who the participants will be. The Delphi should be sent to multiple levels within the organization, since diversity tends to lead to more comprehensive information that is more accurate.
3. Determine a time line for the process.
4. Design the questionnaire. Include the purpose, a description of the process, the time line, and precisely worded questions.
5. Send out the questionnaire.
6. Compile the information.
7. Repeat as needed, with the responses to the questions included with the questionnaire.

Should there be several rounds with no consensus, the group position may be determined by averaging.

Pair-Wise Comparison

A technique that has gained some popularity is pair-wise comparison. Pair-wise comparison is at the core of the Analytical Hierarchy Process (AHP), which provides a means of making decisions or choices among alternatives, particularly where multiple criteria are involved. In pair-wise comparison, the relative importance of each criterion against every other criterion is determined, often in a group session that was preceded by individual "homework." In the case where criteria are grouped into categories, such as when one does criteria decomposition, each criterion is compared to the other criterion in its group. The process is repeated for every level in the hierarchy. Figure 6 diagrams the pair-wise comparison process for a set of four criteria (color, speed,

security, warranty). As a rating for each possible pairing is determined, a formula is applied that identifies inconsistencies (e.g., color two times as important as speed, speed two times warranty, but warranty two times color). Finally, when no inconsistencies remain, weights for the criteria are computed.

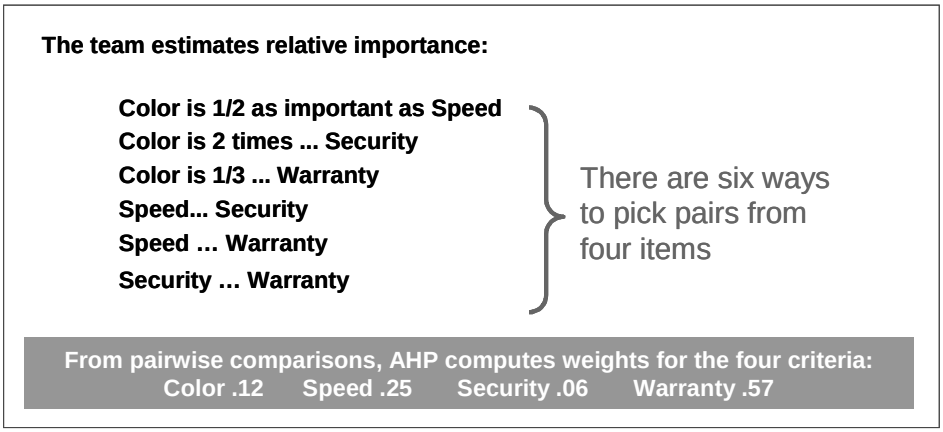


Figure 6: Pair-Wise Comparison

In general, a group whose members have done their individual homework can confer on up to 40 pairs an hour. However, when identifying the relative importance of criterion pairs, it is critical that the imperative to assign a value does not overwhelm the more important discussion of the factors that should be considered in establishing weights.

4 Collecting Data

Much of this report so far has dealt with deciding what data to collect. This section deals with actually collecting the data—the third step in the PECA process. Collecting data provides a basis for analysis. Good data collection is simple, repeatable, measures what it was intended to measure, and captures information in a form suitable for analysis. Accurate data collection is one of the keys to successful COTS software evaluation, yet the act of collecting data is full of surprises—a few good ones and more than a few bad ones. This is one reason for applying an iterative approach to building COTS-based software systems.

Remember that the quality of the evaluation (and the resulting selection) is only as good as the data collected. Confidence in the final results can be improved by ensuring that the data collection methodology is as accurate as possible.

4.1 Results of Collecting Data

The most obvious result of collecting data is the discovery of how products stand against the criteria. However, this is by no means the only result, and in some situations may not even be the most important result. For example, improved understanding of the COTS product marketplace and the system context is invaluable and should be documented.

COTS product evaluation data collection has some similarities with software testing. In software testing the goal is to discover whether the software behaves as expected. Sometimes, the measures taken during COTS product evaluation are exactly the same and help you verify that a COTS product does something in the expected way. However, in other situations there is no concept of expected behavior. In these situations, you might want to discover how well a product can complete a task or what the product is capable of doing. Here, measurements are less defined and more open-ended than those in software testing.

As products are tested and data is collected, you realize how far your understanding was from reality. Some unpleasant discoveries might include

- actual product capabilities different from your expectations
- unexpected interactions and architectural mismatches
- weak vendor responsiveness

- products that don’t even install

A new evaluation iteration may be needed to accommodate changes in your assumptions. Typically, these realizations arise more often in the first iteration of a COTS product evaluation process. As long as the team members expect to find surprises and have mechanisms to deal with them, the evaluation can advance. One very successful program that employed dozens of COTS products adopted the adage, “Pick a (COTS) horse and ride it until the legs fall off—but be prepared to switch horses.”

4.2 Techniques for Data Collection

Different criteria and situations require different data collection techniques. The techniques you choose will be determined in part by the degree of confidence you need in your results. For example, the technique for determining the value of a critical criterion will likely be more rigorous than one used to determine the value of a requirement that carries with it little risk.

The closer the data collection technique comes to the execution of the COTS component in your specific system context, the higher the degree of confidence you can have about how the product will perform in your actual system. You might want to use multiple data collection techniques for a single criterion. For example, data collection techniques for “transactions per second” might include

- trusting the provided documentation
- extrapolating from pre-existing benchmarks
- performing a specific experiment in the operational environment

This report classifies data collection techniques into three categories: literature reviews, vendor appraisals, and hands-on experiments.

4.2.1 Literature Reviews

Just as it sounds, techniques in this category involve the review of literature about specific products. This information can be gathered from many sources. The Internet is a good way to find a plethora of information. User newsgroups and Web pages can be invaluable sources for discovering product strengths and weaknesses. However, Internet information is often inaccurate, so beware of unchecked gossip and rumor, and assess the source and context of the information.

Reports from outside evaluators are also extremely popular sources of information. These reports can be found in magazines and newsletters devoted to the product’s market segment. While there is often value in those reports, they do not address the particular context of your system.

Therefore, the data provided and conclusions reached at best can provide you with only some of the information you will need, and at worst might not apply to your context at all.

Vendors also provide information about products through user manuals, marketing brochures, release notes, version histories, and vendor references. When reviewing these materials, beware of misleading marketing-oriented terminology. For example, two products we studied claimed to implement the SQL standard, but SQL scripts written for one were not compatible with the other. It is also a good idea to follow up on references. We have seen situations in which alleged users were not really involved with the product beyond having requested an evaluation copy.

4.2.2 Vendor Appraisals

Characteristics and capabilities of the vendor must be considered in the evaluation because they affect the use of COTS products. Just as you must live with the characteristics of the selected COTS product (both good and bad), you must also live with the characteristics of its vendor. A vendor appraisal is the analysis of a vendor organization's processes, personnel, and organizational capabilities in the context of how they affect the COTS product.

Sources of information for use in vendor appraisals might include

- vendor business and capability information gathered from interviews, vendor literature, or capability evaluations
- independent financial analyses, such as Standard & Poor's
- strategic information and lists of users provided in trade journals and by vendors
- customer compliments and complaints, found on Web sites or in user groups

4.2.3 Hands-on Techniques

Unless the product you are evaluating is trivial, you *will* need to use hands-on techniques to evaluate it.

Hands-on techniques are beneficial because they can help to

- verify claims
- determine interactions with other components
- determine feasibility of proposed architectures and designs
- determine performance, reliability, and other characteristics *in your context*
- identify assumptions made by the product

Beware of "hands-on" evaluations that are devised by the vendor. They tend to reflect processes that work particularly well and avoid processes that are not so smooth—in any case they are not

your processes. Determining what the basic product contains and what features require tailoring beyond the basic capacity is also difficult. At the very least, such evaluations should be done against real data you provide.

Some hands-on techniques include test beds, product probes, prototypes, and scenario-based evaluations.

Even if information about a product gathered in this phase does not directly relate to how well the product meets the criteria, it should be recorded. This information might include the architecture and design implications of the product, the limitations or conditions placed on its use, or new options for using the product. Evaluators should also note the degree of confidence they have in their data and list any deficiencies in their assessment methods, evaluation requirements, or criteria.

Test Beds

A test bed is the infrastructure required to conduct hands-on experiments. Using a test bed is particularly important when multiple COTS products will be integrated into a larger system. In these cases, evaluations entail more than one product or one set of products. The test bed can be used to mix and match products to determine how they interact.

The test bed should recreate the target system to the greatest extent feasible, with real system components used whenever practical. If real components are not available, simulations may be necessary. Test beds also require support tools, such as test generators, debuggers, performance monitoring software/hardware, test harnesses, and “sniffers.” A test bed complements many other techniques identified in this section.

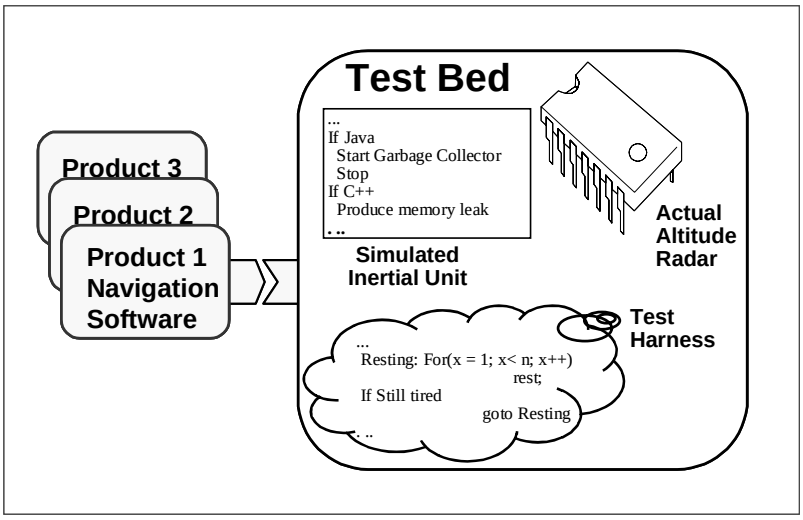


Figure 7: Test Bed

The diagram in Figure 7 illustrates a test bed used for testing COTS software packages for navigation. The test bed is composed of some real components (altitude radar), some simulated components (inertial unit), and code to control the execution of the test bed (test harness).

Ideally, different navigation software packages should be "plugged" into the test bed to compare their characteristics. In practice, such "plug and play" is uncommon, and the test bed will have to be modified to accommodate each navigation software package.

Product Probes

In a product probe, specific features of a product are investigated through guided experimentation to illuminate critical performance characteristics. For example, a probe could be conducted to find out how a product reacts to invalid data or a large volume of data.

Types of guided experimentation that might be used during a product probe include

- error testing
- stress testing
- data stream fault insertion
- testing of reactions at or near the boundaries

Guided experimentation has much in common with system testing. In fact, software-testing tools (such as those that interrupt data streams and inject erroneous data) can be used in guided ex-

perimentation. Discovery activities, which can also be used in a product probe, are far less structured. They involve using the product to "see what happens." Discovery can be made repeatable (and therefore more useful) if the sequence of activities and events is recorded. Even "playing around" is not really playing—you do it for a purpose and need to apply some discipline.

Prototypes

A prototype is a small-scale version of a system used to demonstrate critical features and design decisions. Prototypes are extremely useful in evaluating COTS products because they allow experimentation with products in the context in which they will be used.

Prototypes can be used to observe

- interactions between major components of the system
- interactions between users and the system
- performance of critical product features
- product or system behavior in areas of high technical risk (for example, integration with legacy systems or interoperability with external sources)

For product evaluation, prototypes are often focused on a relatively small set of specific characteristics. We use the phrase *model problems* to refer to these narrowly focused prototypes where the consumer is usually the designer or architect [Wallnau 01].

Scenario-Based Evaluations

A scenario is a step-by-step description of a function that a system must perform. Scenario-based evaluation is an ideal way to focus a product evaluation on the system context, since each scenario is derived from a function that the system must accomplish and can be initially defined without reference to product capabilities.

Scenarios are defined based on use cases, which are sequences of interactions between an actor and the system. An actor is an external entity (person or other system) that communicates with, but is external to, the system. The use case describes the sequence of interactions between the actor and the system necessary to deliver the service that satisfies the goal. It also includes possible variations of this sequence, such as alternate sequences for satisfying the goal or sequences that may lead to failure [Malan 99]. Use cases (and the actors) see the system as a "black box," as illustrated in the following excerpt from *An Example of Object-Oriented Design: An ATM Simulation* [Bjork 98].

> A session is started when a customer inserts an ATM card into the card reader slot of the machine. The ATM pulls the card into the machine and

reads it. (If the reader cannot read the card due to improper insertion or a damaged stripe, the card is ejected, an error screen is displayed, and the session is aborted.) The customer is asked to enter his/her PIN, and is then allowed to perform one or more transactions, choosing from a menu of possible types of transaction in each case. After each transaction, the customer is asked whether he/she would like to perform another. When the customer is through performing transactions, the card is ejected from the machine and the session ends. If a transaction is aborted due to too many invalid PIN entries, the session is also aborted, with the card being retained in the machine. The customer may abort the session by pressing the Cancel key when entering a PIN or choosing a transaction type.

A scenario must reflect a task without including the technical details of how the task should be accomplished. For example, a scenario might suggest that one step in the task would be the permanent storage of data, but should not state that the data will be stored in a database. A single scenario will normally spawn multiple test cases and procedures.

A five-step process for conducting scenario-based evaluations is provided below.

Step 1: Define a scenario by isolating a particular process or subprocess. The scenario might involve multiple use cases.

Step 2: Define specific test procedures for the scenario. Include expected outcomes.

Step 3: Establish an environment that simulates the area of inspection.

Step 4: Insert a product into the environment.

Step 5: Run tests and record results.

Other Techniques

Many other techniques can be used to collect data. Benchmarking, product insertion, and demonstration are three that should be used with caution.

Benchmarks are commonly associated with the measurement of performance, although other types of benchmarks are also available. Unfortunately, benchmarks provided by vendors are notoriously unreliable.

Product insertion is a useful technique when the COTS product will be used with little or no integration with other system components. In such cases, the COTS product can be deployed to a

limited and controlled subset of the users. Product insertion is particularly useful as a technique to analyze the impact of the product on end-user processes.

Demonstration is used by vendors eager to show off their products. Vendor demonstrations are normally free, but rarely touch on product limitations or weak points. Perceptive customers can sometimes infer product weaknesses by observing areas that the vendor avoids. A useful strategy is to ask the vendor to perform the demonstration on data that you provide.

5 Analyzing Results

The fourth step in the PECA process is analyzing the results. Data collection typically produces a large number of facts, checklists, and other types of data. This raw data must be consolidated into information that can then be analyzed, since analysis is required for reasoning about the data collected.

5.1 Consolidating Data

Raw data can be transformed into information that can be analyzed by abstracting and consolidating it. Consolidation means that some detailed information inevitably will be lost. A balance must be struck between the need for easy understanding, which requires a high level of consolidation, and the risk of losing too much information. For example, applying a weighted aggregation technique to condense values for all criteria into a single overall fitness score risks losing the data that suggests strengths and weaknesses with respect to an individual criterion. With high levels of consolidation, two very different products can appear to be virtually identical. Consolidation does not compare products—it simply makes sense of data.

When consolidating data, don't be fooled by the apparent formalism. Sometimes, the apparent mathematical surety of aggregated scores fools evaluators into believing that the results represent immutable facts ("Product A is the best"). However, even a numerical score is not necessarily a fact, and the results must be viewed from a variety of perspectives. Fitness is necessarily a judgment—in scores, weights, and normalization. This is not a bad thing, since human judgment is often critical to the completion of the most complex tasks. But keep in mind that most of your data is ultimately based on judgment.

5.2 Techniques for Consolidating Data

Selection of a product should never be based solely on data consolidated using a single technique, since consolidation masks details and says nothing about individual requirements. Two techniques for consolidating data are all-to-dollars and weighted aggregation.

5.2.1 All-to-Dollars Technique

The all-to-dollars technique treats every piece of data as adding to or subtracting from the bottom line. Product deficiencies are converted into costs, and product excesses are converted into benefits. Thus, there is some cost associated with a problem (such as poor performance) and some value associated with meeting a requirement. Determining values and costs is tricky and often requires you to think about the cost of alternative ways of delivering needed services.

5.2.2 Weighted Aggregation

Weighted aggregation is an extremely popular technique adopted from quantitative decision-making. See Figure 8 for an example of weighted aggregation.

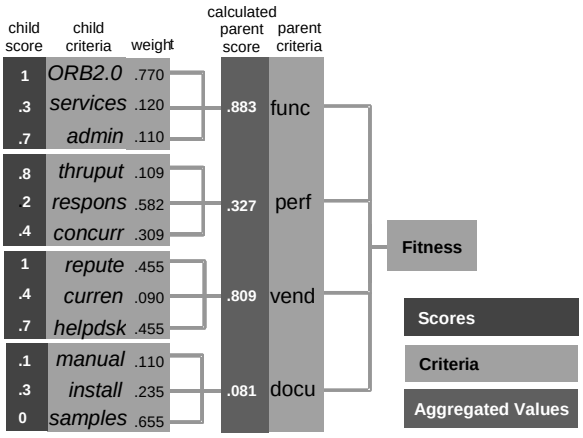


Figure 8: Weighted Aggregation

Observe that the sum of weights at each level of the hierarchy is equal to one. In general, the score of a parent criterion is the sum of the scores of the child criteria times the weights of the child criteria.

score **parent** = Σ (score **child** * weight **child**)

Remember that the score and weight of a child criterion are often subjective; therefore the answer may be quantitative, but fitness is still subjective.

5.3 Techniques for Analyzing Data

Analysis is a creative task, and the best technique is simply sound and careful reasoning. However, there are techniques that can help to guide and support this reasoning. This section outlines three helpful techniques: sensitivity analysis, gap analysis, and cost of fulfillment. Each deals with a particular aspect of the analysis, and these techniques can be applied together.

5.3.1 Sensitivity Analysis

Every evaluation is subject to uncertainty. You might wonder, "Are my measured values accurate? What if my assumptions concerning weights are invalid? How dependent are the results on a particular criterion?" This uncertainty creates risk.

Sensitivity analysis is a valuable but unfortunately underused technique for analyzing results. It involves determining the impact of changing assumptions on resulting scores. For example, if you have reason to believe that measurements are not completely accurate, you can investigate what would happen to the overall fitness of the product if other plausible scores were assigned. Even more commonly, relative weights of criteria may not be exact or may need to be changed with changing circumstances (for example, if the maintenance organization decides to outsource maintenance work on the system). By varying weights, you can determine the impact of these changes on overall product fitness.

By performing sensitivity analysis, genuine insight into the decision being made can be obtained. In addition, conflicts between various stakeholders that have been previously set aside can be analyzed in an objective fashion.

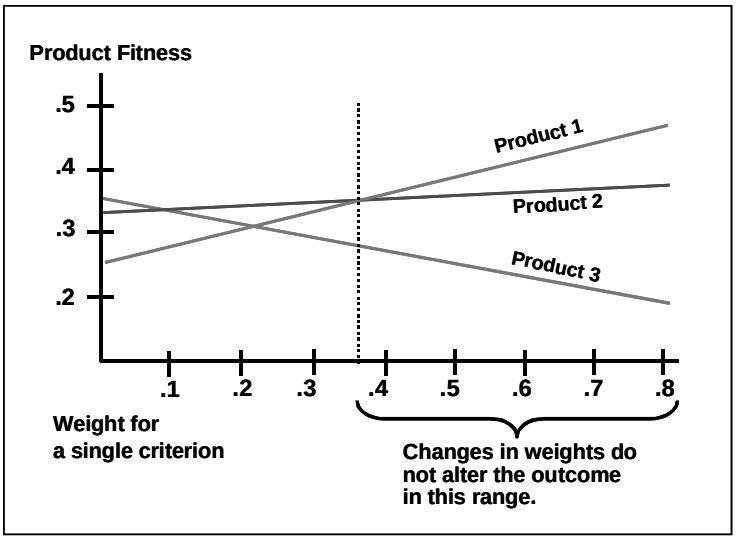


Figure 9: Sensitivity Analysis

In the example shown in Figure 9, the weights applied to a particular criterion have been varied and the resulting change in the overall fitness of the three products has been graphed. Note that if the weight applied to the criterion is reduced below approximately .36, then the product that was more fit at higher weights (Product 1) becomes less fit than a previously inferior product (Product 2). If the weight applied to the criterion is reduced enough, Product 1 becomes the least fit product.

In the real world, weights applied to criteria may change for many reasons: requirements for the system may change, the maintenance strategy for the system may be modified, or the weighting of the experts may become suspect if bias is discovered.

5.3.2 Gap Analysis

Gap analysis compares the measured values associated with criteria to the required values. It provides a sense of "best" and helps to identify significant issues in trying to meet the corresponding requirement.

One important use of gap analysis is to determine the differences between end-user processes and the processes assumed by each COTS product. Major failures have resulted from insufficient effort to determine the gap between as-is and COTS-supported processes.

A gap analysis typically uses a matrix with candidates listed across the top and criteria down the side. Cells can be filled with text describing how well a product feature provides the function. The resulting matrix shows how well a product fulfills the criteria.

As an example, this kind of matrix can help you understand the impact of different choices on end-user processes. First, document the as-is processes. Next, understand the process imposed by each candidate COTS product. Finally, understand the differences using the matrix. Placing competitors side by side in this sort of comparison can help evaluators reason about relative weaknesses and strengths. It can also illuminate overall patterns and provide some sense of global superiority.

A "Yes/No" matrix provides broad understanding of where gaps exist between the product and the required capabilities (see the matrix on the left in Figure 10). However, an understanding of the consequences of the gap must also be developed. A matrix that contains some measure of fitness (shown on the right in Figure 10) can provide a useful starting point. Additional information is still needed to identify the effort required to bridge the gap—the "cost of fulfillment."

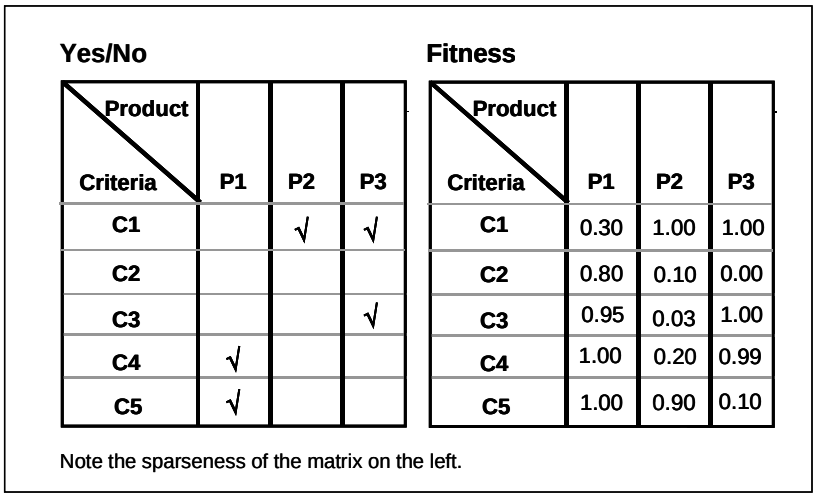


Figure 10: Gap Analysis – 1

The most useful matrix would be one that provided some information on the actual level of fulfillment of the criteria. Figure 11 shows a third matrix that provides this kind of information. Note that the entries are textual; that is, they do not have to be numerical scores. This data allows us to base our calculation of cost of fulfillment on specific information gleaned during evaluation.

Gap Information

Product / Criteria	P1	P2	P3
C1	Limited Java support	Complete solution	Complete solution
C2	Inaccurate math	Precision only to 2 decimals	No Math engine
C3	10% < required reliability	No reliability figures available	Complete solution
C4	Complete solution	Vendor out of country	Vendor Canadian
C5	Complete solution	Linux platform required	Windows only

Figure 11: Gap Analysis – 2

5.3.3 Cost of Fulfillment

Cost of fulfillment involves determining the implications to the system if a product is selected by considering the work that must be done to the system to fulfill deficits in the product. In this case, deficit does not necessarily refer to a product flaw; it may be a capability that is required in the system that the product does not demonstrate. A deficit may also be caused by an overabundance of features (because users must be protected from extraneous features) as well as a paucity of features. "Cost" is not necessarily stated in terms of dollars; it could be in time (delays) or in shifted risks. Cost of fulfillment is analogous to cost of repair in *Building Systems from Commercial Components* [Wallnau 01].

Understanding the cost of fulfillment requires assumptions about the use of the product in the system, including such elements as

- architecture and design
- impact on maintenance
- business considerations

This requires expertise outside the scope of evaluation. Evaluators must team with others to determine possible approaches to the fulfillment and their estimated costs. Figure 11 shows three potential situations that might occur as a result of an evaluation of a product.

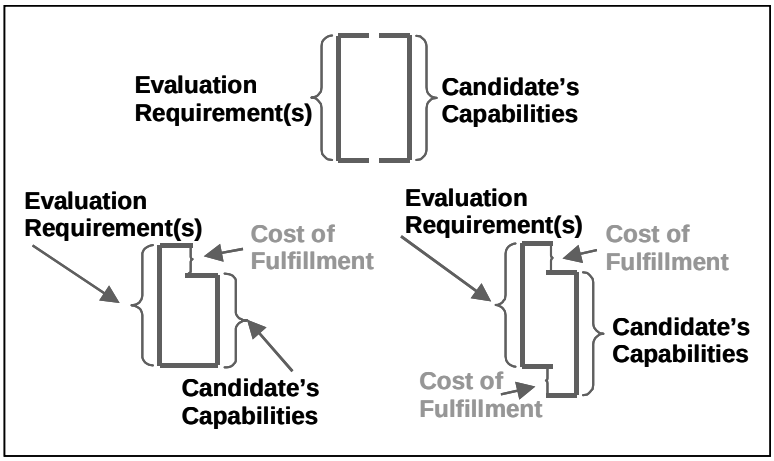


Figure 11: Cost of Fulfillment – 1

The diagram at the top describes a situation where the product capabilities and the requirements match exactly. In this case there is no cost of fulfillment. This situation is highly unlikely. The diagram in the lower left describes a situation where the product does not fully meet the requirements. The only cost involved here is the cost required to meet the remaining portion of the requirements.

The diagram in the lower right describes a situation where a product does not completely meet the requirements but also provides capabilities that are outside of the boundaries of the requirements. In this situation there will be two types of costs involved:

- the cost required to meet the requirement
- the cost required to handle (that is, manage, hide, etc.) non-required functionality

In one approach to determining cost of fulfillment, the following steps are to be repeated for each product:

Step 1: Identify each gap.

Step 2: For each gap, determine

a. one or more fulfillment strategies (e.g., negotiate with the vendor, modify other system components, add custom code, negotiate requirements)

b. the cost to implement each strategy (costs include architecture/design, maintenance, process changes, etc.)

Step 3: Select the preferred set of fulfillment strategies.

Step 4: Calculate total cost of fulfillment for the product.

Based on this analysis, select the preferred product for recommendation.

Keep in mind that the cost to fulfill a deficit has many facets. Not only must new capabilities, wrappers, and filters be created, but also the total cost will likely include training, process change, documentation, updated maintenance strategies, and many other individual costs. Also note that a "deficiency" may actually entail features that the product provides that must be hidden or protected against.

Each evaluation matrix shows the deficiencies for that particular set of products and criteria. For each product in the set, you must calculate the cost of fulfillment strategies. There will probably be a number of different strategies for each product.

As an example of the effort required to undertake a cost of fulfillment calculation for one evaluation, consider the matrix in Figure 12.

Product / Criteria	P1	P2	P3
C1	Limited Java support	Complete solution	Complete solution
C2	Inaccurate math	Precision only to 2 decimals	No Math engine
C3	10% < required reliability	No reliability figures available	Complete solution
C4	Complete solution	Vendor out of country	Vendor Canadian
C5	Complete solution	Linux platform required	Windows only

Figure 12: Cost of Fulfillment – 2

Each highlighted cell requires at least one cost of fulfillment determination, and multiple fulfillment strategies could be investigated for each cell as well. In addition, your solution may be affected by interactions between potential components and conflicts between different fulfillment strategies.

5.4 Making Recommendations

Making a recommendation requires much more than a simple statement like “Product A is the best for our goals.” The goal of a recommendation is to provide enough information to a decision maker to help them select a product. Each decision maker is different, and evaluators need to provide all the information decision makers might want or need to make an informed decision.

The information required can vary according to the type of organization and the characteristics of the decision makers. For example, a bank has fiduciary responsibility for people’s money. The emphasis placed on an evaluation by one bank’s decision makers was on the demonstration of due diligence so that any decision made could be supported and justified to bank investors. This emphasis “flavored” the recommendations.

Evaluators learn many lessons about the use of a product in a system during an evaluation. They also develop highly informed opinions about system architecture and strategies for tailoring, wrapping, testing, and maintenance. All of this information should be documented, even though it might not be possible to recommend any of the evaluated products.

In cases where the recommendation is to reject all products, evaluators often learn critical information regarding alternate strategies for the system. This information might suggest custom-building portions of the system, or it might identify changes to requirements that would allow COTS products to be used. It is the evaluator’s responsibility to document and convey this information as well.

5.4.1 Documenting Recommendations

As stated in Section 1.4.2, the primary outputs used to document recommendations for decision makers are the product dossier, the evaluation record, and the summary with recommendations.

Creating a Product Dossier

The information necessary to understand and use a product is collected in the product dossier. Appendix B provides a template for this document. A dossier should be started for each product when it is first considered for evaluation. As more information about the product is collected, the product dossier is extended. Since the product dossier represents the growing record of a product’s consideration, the dossier for a product rejected early may contain only preliminary information, while a dossier for the chosen product may be quite detailed.

For products that are rejected, the product dossier documents the rationale for rejection. This information is useful during subsequent product evaluations for the same system, and is invalu-

able should the product be considered again during future evaluations. For products that are selected, the dossier maintains the rationale for the selection, but also serves as a central source of information for the team that will architect, design, and build the system around the product, and for those tasked with integrating the product into the system.

The product dossier should include the following information:

- product and vendor information
  - documentation, sources of information, points of contact
- product limitations
  - limitations on use of the product, including features not to use
  - design strategies that make best use of products and compensate for limitations
- discovered facts
  - differences between product and documentation
  - interactions and behavior
  - observed product features
- information about use of product
  - procedures for installing, configuring, and maintaining the product
  - ramifications of product use on system architecture

Creating an Evaluation Record

An evaluation record is an essential item for project tracking, preservation of historical data, and process improvement. See Appendix C for an evaluation record template. The evaluation record describes what the evaluation team has done. The evaluation record should include the following information:

- evaluation plans
- names of personnel responsible for the evaluation
- dates and details of meetings and evaluation tasks
- environment or context in which a product is evaluated
- detailed information about product versions, configurations, and customizations
- results of all of the evaluation activities
- rationale for decisions made
- lessons learned that may be useful for subsequent evaluations

Creating a Summary with Recommendations

Evaluators should present their results in a summary with selection recommendations. Some possible topics to cover in the evaluation summary include

- analysis of fitness
  - observed product features and limitations
  - performance against criteria for all candidates
  - behavior and interactions with other components
  - results of gap analysis, fulfillment strategies, and fulfillment costs
  - results of sensitivity analysis
- analysis of evaluation deficiencies
  - need for further evaluation
  - confidence in results
- discovered facts
  - differences between product and documentation, for example

The analysis of fitness describes how well candidate products fit into the intended context. It might also include an analysis of performance against criteria for all candidates and an analysis of gaps and fulfillment strategies. It might also identify lessons learned about appropriate system architecture and design strategies relative to the products.

The analysis of evaluation deficiencies describes the limitations of the evaluation and the data gathered. It might also identify the degree of confidence in the results and whether any further evaluation should be carried out. In some cases, discovered facts of particular importance may be highlighted in the evaluation summary.

Some possible topics to cover in the evaluation recommendations include

- recommended products or alternate solutions, with rationale and system implications
- if the recommendation is *to buy*, recommendations regarding
  - architecture, design, implementation
  - tailoring or wrapping
  - integration and system testing
  - maintenance and support
- if the recommendation is *not to buy*
  - alternatives (such as build, change requirements, etc.)

6 Conclusion

Some individuals believe that following any documented process is ill advised, particularly when the end goal is to save time and money. Our experience in analyzing troubled programs is that too often highly informal COTS product evaluation processes share the blame for failure. The process described in this report is a *means* of performing COTS product evaluations and not an end in itself. Expect to tailor this process to fit your own situation, and do not let it limit your options in getting good data and making an informed recommendation.

Regardless of the COTS product evaluation process you adopt, remember that evaluation is an ongoing activity. Your organization will need to evaluate new product versions and potentially identify product replacements over the life of your system. If you have a foundation of good evaluation processes and practices, along with good documentation of the characteristics of products and the rationale for decisions, you have a good start at making COTS products work for you.

Appendix A Step by Step Description of the PECA Process

1. Plan the Evaluation (p. 15)

1.1. Define Charter (p. 15)

a. Form the evaluation team.

b. Obtain commitment of both the evaluators and management.

c. Create a charter for the current evaluation effort.

d. Plan the evaluation.

1.2. Identify Evaluation Stakeholders (p. 17)

a. Locate the relevant evaluation stakeholders.

b. Identify their expectations.

1.3. Establish Level of Depth (p. 18)

a. Identify the component's criticality.

b. Identify evaluation complexity.

c. Define the level of depth that will govern the evaluation.

1.4. Identify Approach and Resources (p. 20)

a. Pick a general approach and candidates for the evaluation.

b. Estimate resources required for the evaluation.

c. Develop schedule based on previous estimations.

2. Establish Criteria (p. 21)

2.1. Compile/Negotiate Evaluation Requirements (p. 21)

a. Find subset of system requirements relevant to this evaluation.

b. Normalize the language.

c. Distinguish between negotiable and non-negotiable requirements.

2.2. Define Criteria (p. 27)

a. Create and document a set of criteria sufficient to enable selection.

b. Prioritize criteria.

3. **Collect Data** (p. 21)

3.1 Select measurement techniques.

3.2 Design measurements.

3.3 Obtain data about the products by executing the planned measurements.

4. **Analyze Results** (p. 43)

4.1. Consolidate Data (p. 43)

a. Translate raw data into a basis for recommendation.

4.2. Analyze and Document Results (p. 51)

a. Provide the decision-maker with foundations to choose a product.

b. Document lessons learned for future evaluations.

c. Transfer product facts to system integrators.

Appendix B Product Dossier Template

1. **Product Identification**
   a. Name
   b. Version number, rev number, patches installed, etc.
2. **Vendor Contact Information**
3. **Product Description**

   [*Summary of what the product does and what it is being considered for/how it is used in the system*.]
4. **Product Status**

   [*Current state of decisions made regarding use of the product, whether it has been selected, is being used, actively maintained, or being replaced/retired*.]
5. **State of Evaluation, Testing, Certification**
6. **Vendor Data (includes raw and processed information)**
   a. Financial
   b. Business
   c. Engineering
7. **Product Data (includes raw and processed information)**
   a. Basic characteristics
   b. Standards
   c. Hardware/software configuration required
   d. Functional capabilities
   e. Nonfunctional capabilities

      [usability, supportability, interoperability, reliability, security, etc.]
   f. Interactions and behavior
   g. Performance
   h. Documentation
   i. Licensing
   j. Architecture
   k. Noted discrepancies between the product and its documentation

8. Product Limitations

a. Product deficiencies

b. Limitations on product use

9. System Relationships, Tailoring, and Modifications (includes raw and processed information)

a. System configuration

b. System adaptation

c. System integration

d. Product and system tailoring and modification

e. Design strategies for using product

10. Product Usage History

a. Dates considered, used, retired

b. Bugs/problems reported

c. Disposition of bugs/problems

d. Queries to vendor or third parties for support

e. Changes/updates to configurations and tailoring
[capture rationale, changes, and results]

f. Preventative/other maintenance performed

11. Dossier Usage History

a. Who, what, and why record of access to Dossier components

b. Errata or inconsistencies found
[additional information required]

Appendix C Evaluation Record Template

1. Charter

1.1. Background

a. Date of effort

b. Evaluation team members and qualifications

c. Facilities and resources used

1.2. System Stakeholders

Stakeholder	Requirements	Sponsorship, administration	Contractual information	Technical information	Other

1.3. Approach

a. Depth

— complexity

— risk of failure

b. First fit vs. best fit

c. Number and type of filters

d. Other

2. Criteria Record

Criterion	Negotiability Non applicable Very negotiable Negotiable Barely Negotiable Hard Requirement	Capability Statement	Measurement Method

3. Results Record

Criterion	<product1 name>		<product2 name>	
	Measurement Results	Repair Strategy(s) and Cost of Fulfillment	Measurement Results	Repair Strategy(s) and Cost of Fulfillment

4. Assessment of Evaluation Effort

a. Limitations or deficiencies

b. Rationale for all decisions made

Appendix D Criteria Classification

Vendor Characteristics
Organizational stability
Financial stability
Nationality
Ease of access
Independence
Reputation
Support infrastructure
Engineering approach
Maintenance approach
History

Product Characteristics
First shipment date
Install base
Market share
Market trend
Customer references
End-of-life plans
Availability of training
Access to hotline
Availability of consultants
Delivery method

Hardware Configuration
Type
Memory requirements
Disk requirements
Other storage media
Communications

Standards
DoD standards
Industry standards
Organizational standards
Confidence in adherence to standards

Software Configuration
Operating system
Communications
Database
Related applications
Known compatibility problems

Functionality
Suitability
Accuracy
Security

Usability
Intended use and users
General operability
Skill level required
Responsiveness
Robustness
Help capabilities
Error assist/recovery
Understandability
Learnability

Supportability
Self diagnostics
Disclosure of subcontractors
Effort of upgrade
History of upward compatibility
Site installation support
Site operation support
Tool support required
Analyzability
Installability
Replicability
Preventive maintenance

Interoperability
Data model/format
Support for data access
Support for control by other applications
Infrastructure utilized
Infrastructure commonality

Reliability
Test regimen
Test coverage
Types/frequency of faults
Recovery from faults
Mean time between failures

Performance
Benchmarking results
Time-related behavior
Resource behavior
Surge capacity

Adaptability/Flexibility
Customization approach
Customization effort
Portability
Scalability

Documentation
Availability of design and maintenance documents
Customization
Quality

Training
Materials
Courses
Customization
Policy on reproduction

Licenses
Standard use and maintenance licenses
Site licensing
Quantity discounts
Transferability of license
Development/runtime licensing
License bases (per seat, CPU, other)
Data rights
Escrow
Rights granted on discontinuation of product

Appendix E Generic Organizational Checklist

1. Vendor Contact Information
2. Vendor Financial Evaluation
3. Product Background
4. Product Standards
5. General Business Application
    a. Logical model of business functions
    b. Data dictionary
    c. Online features
    d. External information
    e. Training
    f. Testing
    g. Documentation
6. Product Configurations
7. Physical Characteristics
8. Environment Factors
9. Product Usability
10. Product Customizations
11. Product Reliability
12. Product Supportability
13. Integration
14. Presentation Integration
    a. Appearance and behavior
    b. Interaction paradigm
15. Data Integration
    a. Interoperability
    b. Non-redundancy
    c. Data consistency
    d. Data exchange
    e. Synchronization

16. Control Integration
    a. Provision
    b. Use
17. Process Integration
    a. Process step
    b. Process event
    c. Process constraint
18. Platform Integration
    a. Hardware coupling
    b. Operating system coupling
19. Fees and Terms

Acronym List

4GL	fourth generation language
AHP	Analytical Hierarchical Process
ATM	automated teller machine
BPR	business process reengineering
C3/C4	command, control and communications/ command, control, communications, and computers
CBS	COTS-based system
CORBA	Common Object Request Broker Architecture
COTS	commercial off-the-shelf
DCIS	Defense Commissary Information System
DoD	Department of Defense
ERP	enterprise resource planning
GQM	Goal Question Metric
KBytes/s	kilobytes per second
MBV	Model-Based Verification
MRP	Manufacturing Resource Planning
NDI	non-developmental item
NRC	National Research Council, Canada
OTSO	Off-the-Shelf Option
PECA	Plan/Establish (criteria)/Collect (data)/Analyze

PIN	personal identification number
PORE	Procurement Oriented Requirements Engineering
RDCG	Risk-Driven Criteria Generation
RMC	Canadian Forces Military College
SEI	Software Engineering Institute
SQL	Structured Query Language
SSL	Secure Socket Layer
TLS	Transport Layer Security

Glossary

Aggregation

Collection of particulars into a whole mass or sum.

Acquirer

An organization that acquires or procures a system, software product, or software service from a *supplier* (ISO definition). As used in this paper, it also refers to an individual person within such an organization who participates in acquisition or procurement.

Commercial off-the-shelf (COTS)

Available commercially and directly off-the-shelf.

Commercial off-the-shelf (COTS) product

A product that is

- sold, leased, or licensed to the general public
- offered by a *vendor* trying to profit from it
- supported and evolved by the *vendor*, who retains the intellectual property rights
- available in multiple, identical copies
- used without *modification* of the internals

Consolidation

The transformations applied to raw data in order to condense it and extract useful information.

Cost of fulfillment

The effort required to bridge the gap between the evaluation *criteria* and the capabilities of the products being evaluated.

COTS-aggregate system

A system composed of multiple *COTS products* from multiple *suppliers*, integrated to collectively provide system functionality.

COTS-based system

Any system partially or completely constructed using *COTS products* as integral components.

COTS product evaluation

The examination of *COTS products* for the purpose of determining the products' *fitness* for use in a particular context.

COTS-solution system

A system composed of one substantial *COTS product* or product suite from one *vendor,* tailored to provide significant system functionality.

Criteria

The factors or standard by which the *fitness* of products is judged. Criteria are derived from *evaluation requirements*.

Custom code

Code that has been created by and/or is under the control of the *acquirer*.

Decision maker

Person or team responsible for selecting a particular *COTS product* based on the results of the *COTS product evaluation*.

Depth (of evaluation)

The level of detail, accuracy, and comprehensiveness of an evaluation.

Document (for literature review)

Written material relevant to *COTS products*, such as product documentation, Web-based reports, etc.

Ensemble

A unit or group of complementary parts that contribute to a single effect.

Evaluation requirements

Requirements that apply to the current *COTS product evaluation*; they are derived mainly from the *system requirements*.

Evaluator

A person responsible for executing some or all of the *COTS product evaluation* process.

End users

Those who will use the *COTS-based system* in the intended *operational environment*.

Fitness

The degree to which a *COTS product* meets the needs of a system.

Market Survey

Initial and often informal exploration of the COTS marketplace or market segment in order to locate a set of suitable *COTS products* and *vendors* for further evaluation.

Modification

Changes to the internal capabilities of a *COTS product* that are not part of the *vendor*'s original intent; for a software product, this is generally any changes to the *vendor*'s source code.

Negotiability of a requirement

Degree to which a particular *requirement* can be weakened or removed when the *requirement* cannot be fulfilled. A *negotiable requirement* can be altered, weakened, or removed; a *non-negotiable requirement* must be fulfilled as stated.

Operational Environment

The physical, political, and business environment within which the deployed system must function.

Requirement

A need or necessity. A requirement may be *negotiable* or *non-negotiable*.

Risk

Exposure to the chance of injury or loss; the potential for a situation or set of circumstances to develop into a problem.

Sponsor

The individual or office providing the funding for the development of the *COTS-based system*.

Stakeholder

Someone with vested interest in the results of a *COTS product evaluation* or on whom the selection of a particular *COTS product* will have an appreciable effect. It includes but is not limited to *end users*, *system integrator*s, *acquirers*, *sponsors*, and *decision makers*.

Supplier

An enterprise (not necessarily commercial) whose purpose in producing a product may or may not include making it commercially available for the use of others; examples of suppliers include

government, academic institutions, shareware providers, and not-for-profit enterprises, in addition to *vendors*.

System context

Those entities, parties, or circumstances that place *requirements* and constraints upon the *COTS products* to be integrated: denoting such factors as *requirements* (functional and non-functional), *end-user* processes, business drivers, *operational environment*, constraints, policies, budgets, schedule limitations, and *stakeholders*.

System integrator

An enterprise, not necessarily commercial, whose purpose is to compose various components into a system. In the case of *COTS-based systems*, this could entail such activities as installation and *tailoring* of *COTS products* and creation of "glue code" to make components integrate properly.

System requirement

A need or necessity that the finished system must fulfill or a trait it must exhibit. A system requirement may be *negotiable* or *non-negotiable*.

Tailoring

Changes to COTS software product functions along parameters that are predetermined by the *vendor*. In particular, tailoring is distinguished from *modification*, as it does not change the basic product or its capabilities in any way unintended by the *vendor*.

Vendor

A commercial enterprise whose purpose in producing a product is to offer it for sale, lease, or license in the marketplace.

Bibliography

[Abts 97] Abts, Christopher M. & Boehm, Barry. *COCOTS (Constructive COTS) Software Integration Cost Model: An Overview* (USC-98-520). Los Angeles, CA: USC Center for Software Engineering, University of Southern California. http://sunset.usc.edu/publications/TECHRPTS/1998/1998_main.html.

[Bjork 98] Bjork, R. C. *An Example of Object-Oriented Design: An ATM Simulation.* (CS320 Course Notes). Wenham, MA: Gordon College, 1998. http://www.math-cs.gordon.edu/courses/cs320/ATM_Example/default.html.

[Briand 94] Briand, L.; Morasca, S.; & Basili, V. R. *Goal-Driven Definition of Product Metrics Based on Properties*. (CS-TR-3346, UMIACS-TR-94-106) University of Maryland, Computer Science Technical Report, December 1994. http://www2.umassd.edu/SWPI/ESEG/cs-tr-3346.pfd.

[Brownsword 00] Brownsword, L. & Place, P. *Lessons Learned Applying Commercial off-the-Shelf Products: Manufacturing Resource Planning II Program.* (CMU/SEI-99-TN-015 ADA 379746). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/02.reports/02tr005.html.

[Cambridge 95] *Cambridge International Dictionary of English*. Cambridge, UK: Cambridge University Press, 1995.

[Carney 03] Carney, D.; Place, P.; & Oberndorf, P. *Basics for Assembly Process for COTS-Based Systems* (APCS) (CMU/SEI-2003-TR-010, ADA 413706). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr010.html.

[Consumers 00] Consumers Union of U.S. Inc. *Consumer Reports*. http://www.consumerreports.org/ (2000).

[Constantine 00] Constantine, L. L. & Lockwood, L. A. D. “Structure and Style in Use Cases for User Interface Design.” Published in M. van Harmelen (ed.), *Object Modeling and User Interface Design*, New York, NY: Addison-Wesley, 2000. http://www.foruse.com/articles/structurestyle2.htm.

[CORBA 96] *The Common Object Request Broker: Architecture and Specification*, Version 2.0. Framingham, MA: Object Management Group, 1996. http://www.omg.org.

[Dalkey 63] Dalkey, N. C. & Helmer, O. “An Experimental Application of the Delphi Method to the User of Experts.” *Management Science 9*, 3 (April 1963): 458-467.

[DoD 04] Department of Defense, Business Process Reengineering (BPR). Information available through: http://www.defenselink.mil/nii/bpr/bprcd/ (2004).

[EC 00] Expert Choice, Inc. *Expert Choice 2000 Quick Start Guide and Tutorials*. Pittsburgh, PA: Expert Choice, Inc., 2000.

[Fenton 00] Fenton, Norman E. *Software Metrics: A Rigorous Approach*. London, UK: Chapman and Hall, 1991.

[Garlan 95] Garlan, David & Perry, Dewayne. “Introduction to the Special Issue on Software Architecture.” *IEEE Transactions on Software Engineering*, April 1995.

[Gartner 98] Gartner Group, New Bern, NC. http://www.thegartnergroup.com/ (1998).

[ISO 91] International Organization for Standardization (ISO). ISO/IEC 9126:1991 – *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for Their Use*. Geneva, Switzerland: ISO/IEC, 1991.

[ISO 99] International Organization for Standardization (ISO). ISO/IEC 14598-1:1999 – *Information Technology – Software Product Evaluation*. Geneva, Switzerland: ISO/IEC, 1999.

[ISO 00] International Organization for Standardization (ISO). ISO 9000:2000 – *Quality Management Systems – Fundamentals and Vocabulary.* Geneva, Switzerland: ISO, 2000.

[Kontio 96] Kontio, J. "A Case Study in Applying a Systematic Method for COTS Selection," *Proceedings of the International Conference on Software Engineering*. Berlin, Germany, March 25-29, 1996. Washington, D.C.: IEEE Computer Society, 1996.

[Linstone 75] Linstone, H. A. & Turoff, M. *The Delphi Method: Techniques and Application*. New York, NY: Addison-Wesley, 1975.

[Malan 99] Malan, R. & Bredemeyer, D. *Functional Requirements and Use Cases*, draft white paper, 1999. http://www.bredemeyer.com/pdf_files/functreq.pdf.

[Meyers 01] Meyers, B. C. & Oberndorf, P. A. *Managing Software Acquisition: Open Systems and COTS Products.* New York, NY: Addison-Wesley, 2001.

[Modell 96] Modell, Martin E. *A Professional's Guide to Systems Analysis*, 2nd. Ed. Columbus, OH: McGraw Hill, 1996.

[Ncube 99] Ncube, C. & Maiden, N. A. M. "PORE: Procurement Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm." *Proceedings of the 2nd International Workshop on Component-Based Software Engineering.* Los Angeles, CA, May 6-22, 1999. Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University, 1999. http://www.sei.cmu.edu/cbs/icse99/papers/11/11.htm.

[Ovum 04] Ovum, Boston, MA. http//:www.ovum.com (2004).

[Saaty 80] Saaty, T. L. *The Analytic Hierarchy Process*. New York, NY: McGraw-Hill, 1980.

[Staley 01] Staley, M. J.; Oberndorf, P.; & Sledge, C. *Using EVMS with COTS-Based Systems.* (CMU/SEI-2002-TR-022, ADA403815). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. http://www.sei.cmu.edu/publications/documents/02.reports/02tr022.html.

[Wallnau 01] Wallnau, Kurt; Hissam, Scott; & Seacord, Robert. *Building Systems from Commercial Components*. New York, NY: Addison-Wesley, 2001. http://www.sei.cmu.edu/cbs/bscc/bscc.htm.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2004	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
A Process for COTS Software Product Evaluation	F19628-00-C-0003

6. AUTHOR(S)

Santiago Comella-Dorda, John Dean, Grace Lewis, Edwin Morris, Patricia Oberndorf, Erin Harper

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2003-TR-017

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2003-017

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The growing use of commercial software products in large systems makes evaluation and selection of appropriate products an increasingly essential activity. However, many organizations struggle in their attempts to select appropriate software products for use in systems. As part of a cooperative effort, the Software Engineering Institute and National Research Council Canada have defined a tailorable commercial off-the-shelf (COTS) software product evaluation process that can support organizations in making carefully reasoned and sound product decisions. The background fundamentals for that evaluation process, as well as steps and techniques to follow, are described in this report.

14. SUBJECT TERMS	15. NUMBER OF PAGES
COTS, CBS, PECA, commercial off-the shelf, COTS-based system, evaluation, criteria	90

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL