Applying Decision Analysis to Component Reuse Assessment

Michael S. Russell
The Anteon Corporation

Reusing and combining components of existing systems to create new ones provides a cost effective and flexible method for developing new systems, and is one of the keys behind component-based architecting. However, achieving these benefits in the real world is never easy. The challenge to system developers is not only determining which reusable components to evaluate for possible incorporation into the new system, but also defining what constitutes a reusable component for that system. This article proposes a methodology for applying decision analysis to support component reuse assessment.

New system development has been driven toward component reuse by many factors, including the emergence of rapidly changing technology, faster development timelines and limited budgets, the inability of *stove-piped* legacy systems to deal with information-/network-centric warfare, and the emergence of new system acquisition policies. Reusing components of existing systems is a viable method to overcome these factors. Effective reuse allows system development to stay current with technology, react to tightening schedules and budgets, and manage development risks.

Realizing the benefits of component reuse for the federal government is one of the U.S. Chief Information Officers (CIO) Council's focus areas. The council is currently developing overarching guidance to define component-based architecting for the federal government and the role of component reuse within it [1]. However, the CIO council's objective is not to dictate explicit reuse assessment procedures, rather, it hopes to provide the high-level guidance necessary to set out the scope of such a program.

Similarly, the U.S. Navy is in the midst of developing a component-based shipboard computing environment called the Open Architecture Computing Environment [2]. The program seeks to specify a broad range of reusable components and entire applications upon which to build new systems. In both cases, and for the engineering community at large, a standardized reuse assessment process would be beneficial.

Upon implementing a component reuse assessment process, the first question that comes to the developer's mind is usually: "Which components should I evaluate for reuse potential?" For almost any system development effort, there are an overwhelming number of reusable components that might be considered, and attempts to exploit these components through using traditional engineering approaches have been disappointing [3]. A structured and tailorable decision-making process, incorporating both qualitative and quantitative analysis, is needed to filter out components to be evaluated, select the right reuse candidates, and justify reuse decisions.

This article advances the current state of research in the application of decision-making processes for component reuse by focusing both on the actual evaluation processes and the filtering mechanisms that must be in place to support a successful process application. Most programs have an extremely large group of potentially reusable hardware and software components from which to choose. Being able to objectively filter this group and develop a shorter list of the most likely reuse candidates for in-depth evaluation is critical to meeting budget and schedule constraints.

"... the assessment process is designed to be used as a decisional aid to the development team, not to dictate the decision."

Methodology

The methodology expands upon previous work sponsored by the Software Productivity Consortium (SPC) [4], the Institute of Electrical and Electronic Engineers (IEEE) [5], and others [3, 6, 7]. The SPC's Comparative Evaluation Process (CEP) is a method for quantitatively evaluating software reuse components. IEEE Standard 1517 defines reuse evaluation considerations as part of a software engineering life cycle, and lays out a method for establishing a software code reuse library.

The methodology uses aspects of the CEP and life-cycle implementation guidance from the IEEE. It adds qualitative assessments as an initial reuse candidate filter, extends the method to add hardware component assessment, and defines assessment criteria categories covering functional, technical, and programmatic evaluation issues.

The methodology is comprised of the following steps: bounding the evaluation effort, identification of candidate components, defining evaluation criteria, evaluating alternatives, and analyzing assessment results.

Bounding the Evaluation Effort

The first step is to define the scope of the reuse effort. This presupposes the creation of an operational concept, requirements specification, architecture, or other statement of expected system functionality, interfaces, and deployment concept. These specifications form the framework upon which reuse decisions are built. In the specification, the developers will have allocated desired system functional requirements to objective or conceptual hardware and software system components. This allows the assessment team to generate a list of candidate components during the next step.

During a typical system life cycle, the system specification will normally stop shy of identifying design-level details or mandated implementations that overly constrain the developers design space, while including enough detail to ensure compliance with desired technical standards, legacy interfaces, and other constraints. However, heavy reuse of existing components mandates some changes to this approach. The specification for a system featuring reuse will have many of its design elements predetermined from the start.

These constraints should be identified early in specification development if known. Most likely, some reusable components will be known at this time and will influence specification development. However, it is just as likely the developers will not know or only have a general idea about what types of reusable components they want to incorporate into the system. This mandates a system life-cycle model that allows the specification to be modified after the best reuse component candidates have been identified, assessed, and selected for incorporation into the system. Additionally, the integration of many reusable components may bring up compatibility and interface problems that were not readily apparent even after several system design iterations. So a program manager desiring to use any level of reuse must be willing to revisit the original specifications as needed to ensure the resulting system meets the customer's requirements.

Identification of Candidate Components

Individual system requirements, allocated through the systems architecture, could potentially be met with different combinations of hardware, software, and business processes. Programmatic requirements to maximize using reusable components complicates system design by adding additional variables such as proprietary protocols, hidden or inaccessible algorithms, and emergent functionality that appears after the system is wired together.

Additionally, developers have to determine which portions of the system are best built using reusable components, which should be based on custom development and how business processes are supported by each. Together, potential combinations of process and technology form a multi-variable quandary for system developers.

Identification of candidate-reusable components should start by deciding what required functionality should be instantiated by reusable components. The functionality to be replicated forms the basis for the initial selection of candidate reuse components. Second, programmatic or legal requirements such as a mandate to use *component X* for all new systems development, will flesh out the initial list.

The initial list of candidates will take some research and may result in an extremely large set of components. In some cases the desired functionality may be replicated using reusable hardware only, reusable software only, or some combination of both. At this point in the process, the developers should not filter out potential reusable components on an ad-hoc basis; rather, developers should seek to identify as many potential component options as possible to support a variety of systems design options and objective decision-making.

Category	Definition
Functional Requirements	Those requirements that outline the expected behavior of the system. This behavior is required for the system to perform its intended purpose.
Non-Functional Requirements	Those requirements that address expected behavior or other aspects of the system that are not required for the system to perform its intended purpose but serve as an enabler.
Technical Requirements	Requirements that specifically define technical constraints and interfaces that the components must adhere to.
Programmatic Requirements	Requirements arising from budgetary, schedule, or resource constraints. For instance, a component that will not be available until after the system is delivered would not meet a schedule requirement.

Table 1: *Evaluation Criteria Categories*

Defining Evaluation Criteria

There are four categories of criteria that will be used to assess each reuse candidate. Due to the variety of reuse candidates being assessed, some candidates may not receive an evaluation in each category; however, this will be the exception. Furthermore, these categories serve as the starting point. Since each project is different, additional categories may need to be defined to evaluate required component functionality – or functionality that the component should not exhibit. Table 1 defines these categories.

"The determination of which evaluation category takes precedence will be different for each system ..."

Functional and non-functional requirement criteria are derived from the system's specification document or other document that outlines the system's requirements. If the criteria were derived from a functional decomposition, assessments will be generated for the lowest level of decomposition (the leaf node level). Ensure the matrix contains the entire functional decomposition; otherwise, dependencies relationships between functions will be missing, and the assessment results may become skewed.

Technical and programmatic category criteria are derived from industry standards, government policies, best practices, and other documents. This set of criteria tends to focus more on the hardware and physical interoperability of the reuse candidates. For reuse candidates that can be separated into hardware and software components, separate assessments will be conducted. Conversely, reuse candidates that cannot be separated into hardware and software components will be assessed as one system. The reusability criteria in these sections generally try to answer the following types of questions:

- Are the reuse candidates compatible with existing government and industry technology standards and mandates?
- Are the reuse candidates compatible with existing organizational standards, processes, and other organizational-specific mandates?
- Are the reuse candidates mature and stable enough to ensure their long-term viability as a part of the system's design?
- Are the reuse candidates sufficiently documented to allow rigorous analysis of their functionality, interfaces, and potential for integration with other components?
- Have all schedule and budget issues associated with the reuse candidates been considered and documented?

Next, each evaluation criteria is assigned a weight. This weight is used to judge the relative importance of a specific criterion to other criteria regardless of category. For example, if the system has five functional requirements, only four may be deemed critical giving them a weight of 0.9. Table 2 contains the criteria weights and definitions.

A system specification typically addresses many requirements that are *nice to have* versus essential for system success. For instance, the ability of a word processor program to check the spelling of a document may be critical (a score of 0.9), while the ability to check document spelling in real time while the user types

Table 2: *Criteria Weight Definition*

Weight	Definition
0.1	Failure to meet this criterion is minimal, or its impact on the system can be mitigated.
0.5	This criterion is an important contributor to system capability, but not essential.
0.9	This criterion is critical to system success

Component Name: (My Component)			Reviewer: (Name)
Criteria Number	Functional Requirements Category	Assessment Result	Reviewer Notes
1	Component Cost		
2	Size of User Community		
3	Maturity		
4	Integration Cost		
5	Schedule Delay Imposed by Using That Component		

Figure 1: *Initial Assessment Matrix Example*

the document may not be critical (a core of 0.5). The correct weighting for each criterion may be derived from many sources, including:

- Customer requirements and expectations.
- System requirements volatility.
- Technical maturity of the system domain.
- Statements of *must have features versus nice-to-have features*.

Once the criteria have been defined and weighted, evaluation matrixes listing each criterion will be generated. The purpose of the matrix is to standardize the evaluation process, thus providing a clearer and more defendable reuse evaluation result and component recommendation. Example matrixes are discussed in the next section.

Evaluating Alternatives

Initial Assessment: Qualitative

The initial assessment is qualitative and used as a mechanism to filter the set of reuse candidates to select the two or three candidates that offer the best match to the system's requirements. The actual number selected will depend on the amount of resources the program can expend to support more detailed assessments. To conduct the initial assessment, the evaluator must first identify the modules (subcomponents, software segments) of the reuse candidate. The individual modules, rather than the component as a whole, should be assessed against the criteria. The objective is to understand which portion of the reuse candidate fulfills the criteria.

The evaluator will fill out a separate criteria assessment matrix for each reuse candidate. For instance if there are four candidates, the evaluator should have four completed matrixes at the end of the assessment. The assessment matrixes will be used to help the system developers filter though the complete set of reuse candidates, thus providing them with a rationale for choosing the most likely reuse candidates for further analysis. Figure 1 contains an example initial assessment matrix.

To record the assessment, a *1* is put into the *Assessment Result* column if the reuse candidate fulfills the criteria; a *0* is inserted if not. In general, the criteria should be rather loosely interpreted, as this qualitative assessment is designed to determine only if the reuse candidate should be assessed in more detail later on.

After the evaluation team has finished the initial assessments on each candidate component, the system's stakeholders will use this evaluation as a decisional aid to select the most likely reuse candidates. The selected reuse candidates will undergo a more thorough evaluation during the detailed, quantitative assessment.

Detailed Assessment: Quantitative

The detailed assessment is quantitative in nature and is used to select the optimal reuse candidate to fulfill each system requirement. Each reuse candidate selected for further assessment during the initial analysis will be evaluated again using one of the methods in Table 3. The selected assessment method will have a significant bearing on the overall evaluation of that reuse candidate, and becomes one of the variables that figures into the component's overall evaluation. For instance, a hands-on evaluation of a candidate component will render better information than simply reviewing marketing literature on the same component.

For large reuse candidates, candidates with many subsystems, or candidates that encompass a mixture of hardware and software, it is conceivable that multiple assessment methods may be justified. Another situation in which multiple assessment methods might be used is when the reuse assessment must be conducted under a compressed timeline. In this case, the evaluators may decide to conduct hands-on testing on the most critical functionality that the reuse candidate must exhibit.

To perform the detailed assessment, the evaluator will determine a score based on the assessment results that indicate the extent to which the reuse candidate fulfills the requirements and metrics of each system criterion. The scores and their definitions are included in Table 4.

This assessment will result in two scores: a criteria score and a composite score. The criteria score is a weighted assessment of the reuse candidate's ability to meet the requirements of each individual criterion. The composite score is the overall score for each category. The process for deriving the criteria score is as follows:

$$\textbf{criteria score} = \textbf{(criteria score)} \times \textbf{(criteria weight)} \times \textbf{(assessment method)}$$

The composite score is derived in this way:

$$\textbf{composite score} \sum_i = \textbf{(criteria score}_i\textbf{)}$$

where:

Table 3: *Assessment Methods*

Assessment Method	Weight	Definition
Verified	1.0	Verified by the evaluator using hands-on examination in a lab environment.
Demonstrated	0.7	Witnessed by the evaluator in a focused demonstration by an experienced user.
Observed	0.5	Seen by the evaluator but not studied in any depth.
Reported	0.3	Described by a user, associate, or vendor, or seen in vendor or third-party literature.

Table 4: *Reuse Candidate Scoring*

Score	Definition
0.1	No capability to meet the criteria demonstrated.
0.3	Meets <50% of requirements and/or customization not possible.
0.5	Meets 50% of requirements and customization is possible.
0.8	Meets 90% of requirements and customization is possible.
0.9	Meets all system requirements.
1.0	Exceeds system requirements and allows further growth opportunities.

$$\textbf{criteria score}_i = (\textbf{criteria score}_1 \square \textbf{criteria score}_n)$$

The component's detailed assessment can be captured in a matrix similar to Figure 2.

Component Name: (My Component)			Reviewer: (Name)		Component Source:	
Criteria Number	Functional Requirements Category	Quantitative Assessment	Criteria Weight	Weighted Assessment	Assessment Method	Criteria Score
1	Component Cost					
2	Size of User Community					
3	Maturity					
4	Integration Cost					
5	Schedule Delay Imposed by Using That Component					
				Composite Score		

Figure 2: *Evaluation Matrix Example*

Analyzing Assessment Results

The final step in the process is to analyze the assessment results and select the reusable components that become part of the new system. At this point, the system developers will have to make a critical determination. What is more important: integration flexibility, functionality, programmatic mandate adherence, or some other concern?

For instance, a candidate that implements a required function, but faces integration challenges, may be chosen over a competing candidate that does not implement the function as well but is easier to integrate into the overall system. The determination of which evaluation category takes precedence will be different for each system, and will result in category-specific weights – i.e., functionality is twice as important as programmatic mandate adherence.

The candidate score represents the summation of the functional, non-functional, technical, and programmatic categories with program-specific category weighting. It is used along with the other component scores as an aid to the system developers so they can decide which reuse candidates to incorporate into the system. The procedure used to calculate the scores is the following:

$$\textbf{candidate score} = \sum_i (\textbf{composite score}_i) \times (\textbf{category weight})$$

where:

$$\textbf{composite score}_i = (\textbf{functional composite score, non-functional composite score, technical composite score, programmatic composite score})$$

At this point, determining the best reuse candidate seems as simple as selecting the reuse candidate with the highest candidate score; however, this can be misleading. The process is only designed to guide the selection of components by mathematically assessing those components and providing a means by which a reasonable comparison between components can be made. As such, the assessment process is designed to be used as a decisional aid to the development team, not to dictate the decision. The benefit of the process to the decision maker is a structured, repeatable, and defendable process on which to base system design decisions.

Conclusion

The Missile Defense Agency's National Team for Command, Control, and Battle Management successfully implemented this approach as part of their block 2004 systems development [8]. This project demonstrated that the ability to objectively judge the suitability of individual components is a critical part of the design process. Achieving the full benefits of component-based architecting depends upon making objective and optimal reuse decisions. System developers must implement decision analysis into their component reuse assessment process, and use this assessment to guide their component reuse decision-making process.◆

References

1. Chief Information Officers Council. Component-Based Architectures and the Federal Enterprise Architecture; draft v1.6. Washington, D.C.: Federal Enterprise Architecture Components Subcommittee, 2003.
2. Program Executive Office, Integrated Warfare Systems. Open Architecture Computing Environment Design Guidance, v1 (Interim). Washington, D.C.: U.S. Navy, 2003.
3. Albert, C., and L. Brownsword. "Evolutionary Process for Integrating COTS-Based Systems: An Overview." Technical Report CMU/SEI-2002-TR-009. Pittsburgh, PA: Software Engineering Institute, July 2002.
4. Phillips, B., and S. Polen. "Add Decision Analysis to Your COTS Selection Process." CrossTalk Apr. 2002 <www.stsc.hill.af.mil/crosstalk/2002/04/phillips.html>.
5. Institute of Electrical and Electronics Engineers. IEEE 1517-1999: IEEE Standard for Information Technology – Software Life Cycle Processes-Reuse Processes. New York: IEEE Standards Board, Jun. 1999.
6. Kang, K., et al. "A Reuse-Based Software Development Methodology." Special Report CMU/SEI-92-SR-4. Pittsburgh, PA: Software Engineering Institute, Jan. 1992.
7. Griss, M. Architecting for Large-Scale Systematic Component Reuse. Palo Alto, CA: Hewlett-Packard Company Laboratories, 2000.
8. National Team for Command and Control, Battle Management, and Communications. NTB Program Directive: Component Reuse Assessment Process. Washington, D.C.: Missile Defense Agency, Nov. 2002.

About the Author

Michael S. Russell is chief architect and technical director for the Anteon Corporation's Center for Missile Defense. He currently supports the U.S. Navy's Open Architecture Program, and has served as lead architect on numerous federal, Department of Defense, and industry enterprise architecture efforts. He is a visiting lecturer with the Federal Enterprise Architecture Certification Institute, and is a member of the International Council on Systems Engineering's central Virginia chapter. Russell has taught courses in Systems Engineering and Architecture Development for the past seven years. He has a master's degree in system engineering from George Mason University.

The Anteon Corporation
2231 Crystal DR
STE 600
Arlington, VA 22202
Phone: (703) 864-1258
Fax: (703) 521-1027
E-mail: mrussell@anteon.com