

# Metrics-Based Framework for Decision Making in COTS-Based Software Systems

Sahra Sedigh-Ali and Arif Ghafoor  
School of Electrical and Computer Engineering  
Purdue University  
West Lafayette, IN 47907-1285  
{sedigh,ghafoor}@ecn.purdue.edu

Raymond A. Paul  
Department of Defense, OASD/C3I  
Pentagon  
Washington, DC 20453  
ray.paul@osd.pentagon.mil

## Abstract

*The growing reliance on Commercial-Off-The-Shelf (COTS) components for developing large-scale projects introduces a new paradigm in software engineering, which requires the design of new software development and business processes. Large scale component reuse leads to savings in development resources, enabling these resources to be applied to areas such as quality improvement. These savings come at the price of integration difficulties, performance constraints, and incompatibility of components from multiple vendors. Relying on COTS components also increases the system's vulnerability to risks arising from third-party development, which can negatively affect the quality of the system, as well as causing expenses not incurred in traditional software development. We aim to alleviate such concerns by using a framework based on software metrics to accurately quantify factors contributing to the overall quality of a COTS-Based System (CBS) and guide decisions regarding quality and risk management.*

## 1 Research Description

The paradigm shift to Commercial Off-the-shelf (COTS) components appears inevitable, necessitating drastic changes to current software development and business practices. Quality and risk concerns currently limit the application of COTS-Based Systems (CBSs) to non-critical applications. New approaches to quality and risk management will be needed to handle the growth of CBSs [10, 1, 2, 4, 3]. With software development proceeding at Internet speed, in-house development of all system components may prove too costly in terms of both time and money, as witnessed by the outsourcing trend currently present in commercial software development. Large-scale component reuse and COTS component acquisition can generate savings in de-

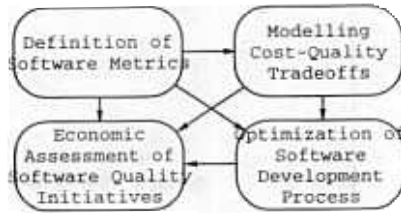
velopment resources, which can then be applied to quality improvement, including enhancements to reliability, availability, and ease of maintenance.

Prudent component deployment can also localize the effects of changes made to a particular portion of the application, reducing the ripple effect of system modifications. This localization can increase system adaptability by facilitating modifications to system components or integration code, which are necessary for conforming to changes in requirements or system design. COTS component acquisition can reduce time to market by shifting developer resources from component-level development to integration. Increased modularity also facilitates rapid incremental delivery, allowing developers to release modules as they integrate them and offer product upgrades as various components evolve.

These advantages bring related disadvantages, including integration difficulties, performance constraints, and incompatibility among products from different vendors. Further, relying on COTS components increases the system's vulnerability to risks arising from third-party development, such as vendor longevity and intellectual-property procurement. Component performance and reliability also vary because component-level testing may be limited to black-box tests, and inherently biased vendor claims may be the only source of information [5, 1, 4]. Such issues limit COTS component use to non-critical systems that require low to moderate quality. Systems that require high quality cannot afford the risks associated with employing these components [7, 11, 6].

We believe that a rigorous framework for decision making in software management is vital to the development of reliable and safe software systems. We propose to alleviate quality and performance concerns by using software metrics to guide decisions regarding quality and risk management in a CBS, accurately quantifying various factors contributing to the overall quality of a CBS, and identifying and elim-

inating sources of risk. We illustrate the use of metrics in guiding decisions throughout the software life cycle and determining whether software quality improvement initiatives are financially worthwhile [8, 9]. Our research addresses the issues of cost and quality management in CBSs.



**Figure 1. Software Quality Management Research Areas.**

As in any development or manufacturing process, software quality is achieved at a cost. Our research uses metrics to quantify the concept of quality, investigates the tradeoff between cost and quality, and uses the information gained to guide quality management. We focus on four main areas in software quality management: metrics definition, modeling of cost-quality tradeoffs, optimization of the software development process, and economic assessment of quality improvement initiatives. These areas are interrelated, as depicted 1.

Our framework begins with the collection of software metrics and development of cost-quality models for CBSs. We have preceded development of the framework with the definition of a set of metrics representing various aspects of quality in CBSs [8, 9]. We also develop techniques for measuring and estimating software metrics data, as well as investigating the relationships among the metrics and their effect on the cost-quality models.

Our research also includes the development of analytical techniques and formal models for quality improvement decisions. In this portion of the research, we formulate software development decisions as multiple objective optimization problems and utilize statistical decision theory concepts such as Markov Decision Processes to model uncertainty regarding the components, competing products, and other environmental factors. We then apply the analytic techniques developed to guide various management decisions in the software engineering process.

Our research objectives include: evaluation of the relationship between software metrics, determination of optimal software release time based on cost-quality tradeoffs, identification of cost and quality bottlenecks in system design, risk analysis based on software metrics, and sensitivity analysis of cost and quality models with respect to a particular quality metric.

## References

- [1] B. Boehm and C. Abts. COTS integration: Plug and pray? *IEEE Computer*, 32(1):135–138, Jan. 1999.
- [2] P. Brereton and D. Budgen. Component-based systems: A classification of issues. *IEEE Computer*, 33(11):54–62, Nov. 2000.
- [3] L. Brownsword, T. Oberndorf, and C. A. Sledge. Developing new processes for COTS-based systems. *IEEE Software*, 17(4):48–55, Jul. 2000.
- [4] J. Hopkins. Component primer. *Comm. of the ACM*, 43(10):27–30, Oct. 2000.
- [5] J. McDermid. The cost of COTS. *IEEE Computer*, 31(6):46–52, Jun. 1998.
- [6] P. Rodriguez-Dapena. Software safety certification: A multidomain problem. *IEEE Software*, 16:31–38, Jul./Aug. 1999.
- [7] N. F. Schneidwind. Methods for assessing COTS reliability, maintainability, and availability. In *Proc. of the Int'l Conf. on Software Maintenance (ICSM '98)*, 1998.
- [8] S. Sedigh-Ali, A. Ghafoor, and R. A. Paul. Metrics-guided quality management for component-based software systems. In *Proc. of the 2001 Int'l Conf. on Computer Software and Applications (COMPSAC 2001)*, pages 303–308, Oct. 2001.
- [9] S. Sedigh-Ali, A. Ghafoor, and R. A. Paul. Software engineering metrics for COTS-based systems. *IEEE Computer*, 34(5):44–50, May 2001.
- [10] C. Sledge and D. Carney. Case study: Evaluating COTS products for DoD information systems. *SEI Monographs on the Use of Commercial Software in Govt. Systems*, June 1998.
- [11] J. M. Voas. The challenges of using COTS software in component-based development. *IEEE Computer*, 31(6):44–45, Jun. 1998.