Commercial Off-the-Shelf (COTS) Software in Safety Systems

J. A. Scott
G. G. Preckshot

This paper was prepared for submittal to the 6th International Conference on Nuclear Engineering (ICONE-6)
San Diego, CA
May 10-15, 1998

January 25, 1998
DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.
COMMERCIAL OFF-THE-SHELF (COTS) SOFTWARE IN
SAFETY SYSTEMS

John A. Scott, (510) 423-7655, scott7@llnl.gov
G. Gary Preckshot, (510) 423-0902, preckshot1@llnl.gov
Fax: (510) 422-9913
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550

6th International Conference on Nuclear Engineering
ICON-6
San Diego, California
May 10-15, 1998
Copyright 1998, ASME
Abstract

Many safety systems, such as those in nuclear power plants, are systems for which the consequences of failure can be severe or catastrophic. These systems must be developed, implemented, and maintained in ways that provide assurance that catastrophic consequences will be prevented. This paper discusses various aspects of the question of using commercially available software in these systems. Risk, grading, and system assessment are discussed, and relevant standards are summarized. Recommendations for addressing key issues are given.

1. Introduction

1.1 Discussion of the Issue

Systems with potentially severe or catastrophic failure consequences, e.g., nuclear power plant safety systems, must be developed, implemented, and maintained in ways that provide assurance that catastrophic consequences will be prevented. Approaches may include techniques such as fault avoidance (keeping faults out of the system), fault tolerance (designing the system to tolerate faults without progressing to severe consequences), or diversity and defense-in-depth (providing additional measures to prevent or mitigate the consequences of system failure). Developers of these high-integrity systems focus on achieving safety goals even if extensive resource utilization is necessary. Historically, guidance for developing high integrity software has been based on custom developments using state-of-the-art software and safety engineering practices, both at the system and software levels.

Recently, economic pressures have resulted in a trend toward streamlining and increasing organizational efficiency. Because they are expensive, system development activities are a focus of efforts to increase efficiency and reduce costs while maintaining adequate quality (Poulin, 1993). In particular, it is important to allocate relatively scarce, skilled professionals to unsolved safety problems rather than to those already solved. While the costs associated with mass-produced electronic hardware have been decreasing steadily for many years, costs for labor-intensive software development have been continually increasing (for example, see (Griss, 1993)). Consequently, software costs account for a growing percentage of system development costs, which creates considerable pressure to improve efficiency and productivity in software development activities.

1.2 COTS & In-house Software Reuse

Effective reuse of previously developed software (PDS) is a potential contributing factor for increasing software development productivity while maintaining quality. PDS can be obtained from previous software development efforts within an organization (in-house software reuse) or from commercial sources (use of COTS software). In the arena of nuclear safety systems, PDS may consist of existing vendor software developed according to a 10 CFR 50 Appendix B program, other vendor in-house software, or commercially available software. In the latter two cases, the processes for evaluating PDS acceptability are essentially the same. It has also been noted that there are a number of inter-related factors, including software reuse, associated with software productivity (Blackburn, 1996). Software reuse, while important, is not the only driver of software productivity increases. Therefore, it should not be assumed, a priori, that software reuse is a panacea or that reusing PDS will necessarily be beneficial in a system development effort.
In-house software reuse has long been a goal of software development organizations. Early efforts tended to focus on code reuse, typically via subroutine libraries, and were not usually successful on a large scale. More recently, the advent of object-oriented software techniques offers the hope of achieving higher levels of software reuse. Recent experience (Griss, 1993) indicates that effective software reuse requires a planned and carefully managed systematic process that departs somewhat from the traditional software development approaches. Griss notes: “While library-based reuse has been fairly successful with stable, well-understood, low-level application areas such as user interface libraries, math libraries, and statistical packages, this has not yielded a major change in the way most people develop software.” More recent efforts attempt to achieve software reuse on a scale larger than that of a single organization. One example is the Asset Source for Software Engineering Technology (ASSET) facility established under ARPA’s Software Technology for Adaptable, Reliable Systems (STARS) program. One of ASSET’s goals is to provide a national marketplace for reusable software products. As implied above, the benefits of effective in-house software reuse have associated costs. In addition to the costs associated with the establishment of a systematic reuse program, Poulin (1993) notes that “Developing software for reuse by others requires additional development, validation, and support investments that must be weighed against the benefits to outside organizations and any cost recovery plans.”

The above discussion serves to illustrate two points:

1. While the reuse of PDS seems like a simple, straightforward exercise, it cannot be done effectively without significant planning and investment, and
2. There must be a careful cost-benefit analysis in order to understand the overall value of reuse, i.e., low cost and high benefit cannot be assumed.

Employing commercial off-the-shelf (COTS) software in the development of high integrity digital systems is another form of software reuse. As with reuse of in-house software, a planned and systematic implementation process is needed and must demonstrate that the PDS is adequate for its intended use. Also, the decision to use COTS software should be supported by a cost-benefit analysis. In addition, several new concerns are added beyond the technical issues of in-house software reuse, such as license issues, arranging for long term support, and the difficulties of assessing the quality of a commercial product. In many low-risk applications, it is practical to incorporate this type of software into systems. In a number of other potential applications, e.g., reactor safety systems, it requires a considerable leap of faith to believe that safety systems developed by knowledgeable reactor instrumentation engineers can be replaced by COTS software. Yet the modern trend toward more and more engineering being done in software packages makes this an inevitable suggestion.

1.3 Risk

Treatment of risk is a key difference between high-consequence systems, such as safety systems, and those of lower consequence. Risk, or risk exposure, is usually taken to be the product of consequence times the probability of occurrence. For high-consequence systems, the extent of a high-consequence failure and its probability of occurrence are difficult to predict. Assuming low probabilities for “rare” events may be optimistic. Human beings are constantly surprised to discover interdependencies between things they thought were independent. The implication for COTS software used in safety systems of nuclear power plants is that considerably higher standards of proof of suitability will be required than would normally be the case for typical industrial hazards. This makes risk grading a useful gauge for COTS assessment activities.

1 Software reuse is a superset of code reuse involving software items other than code, such as software designs.
There are a number of other considerations related to risk exposure. Any system architecture using COTS products must be tolerant of product failure. This is true for any software product used in this venue, because nobody has yet discovered how to build error-free software, or, for that matter, write error-free specifications. Also, custom made software is built expressly for a specific application, falls under the same engineering controls as the larger system, and is designed for maintainability by the target industry. In contrast, COTS products are usually not designed for narrow applications, but instead must appeal to a broader market by commercial necessity. There is usually a target price, a time-constrained market window, and a set of features to be delivered for the target price. To broaden the appeal of the product, it is often multi-configurable, and the developer is responsive to the demands of the most lucrative segment of his clientele. Compared to the custom developed product, the COTS product is more likely to have extra features that may surface as unused or unintended functions in a safety system. In high-risk applications, therefore, COTS products cannot be the last line of defense.

1.4 Regulatory Perspective

The general goal of regulatory bodies is to ensure that adequate levels of safety are achieved when high-consequence systems within their purview are developed. Meeting this goal can require that ‘best engineering practice’ be employed. Within the software community, considerable effort has been expended to establish guidance regarding the application of maturing software engineering practices to new developments (e.g., see the software engineering standards from the IEEE or ISO/IEC). Since, in most cases, software development methods cannot guarantee the absence of faults, this guidance typically calls for comprehensive, extensive application of currently accepted software engineering practices. In cases with potential catastrophic consequences, additional systems measures such as diversity and defense-in-depth might be needed. Guidance issued by the U. S. Nuclear Regulatory Commission (NRC) on the review of digital instrumentation and control systems in safety systems of nuclear power plants is contained in the recently revised Chapter 7 of the Standard Review Plan (USNRC, 1997).

The pressures to employ COTS software in high-integrity systems create a new set of difficult questions for regulators. Aside from technical questions related to the safe integration of a COTS software product into a system, other difficulties are also encountered due to various competitive or management concerns. These concerns tend to interfere with access to records and other information needed for product assessment. If the desired records and information are unavailable, the regulator (and developer) is then faced with either rejecting the COTS product for use in a high-integrity system or with making a determination about the acceptability of a COTS software product based on acceptance criteria that are different than those used for new developments. There is a danger in the search for alternatives. It must be shown convincingly that alternative acceptance criteria provide confidence equivalent to that obtained from the processes applied to new developments since, if it is possible to achieve this confidence with less costly new alternatives, the existing processes used in new developments will quickly be abandoned. Unhappily, the search for alternatives to the assurance techniques used for new developments frequently leads to issues that are still subjects of research in software engineering.

2. The Assessment Process

2.1 Need for Assessment

Use of COTS software in larger application software systems (such as application code or operating systems), as stand-alone system components, or as software tools (such as compilers), requires a determination that the software is acceptable for its intended use. Making this determination, if it can be made for a given product, requires the use of a planned and documented assessment process that addresses a wide range of technical considerations at the system and software levels. It should be noted that an assessment process does not add quality to a product and that it may result in a determination that a given product is inadequate for its intended use. In the nuclear power plant environment, 10 CFR 21 defines dedication as an “acceptance process undertaken to provide reasonable assurance that a commercial grade item to
be used as a basic component will perform its intended safety function and, in this respect, is
deemed equivalent to an item designed and manufactured under a 10 CFR 50, Appendix B,
quality assurance program.” In this case, the definition of commercial grade item includes both
COTS software and the reuse of vendor in-house software not developed to the requirements of
10 CFR 50, Appendix B. Also in this case, as noted in (EPRI, 1996), the assessment process
itself is subject to Appendix B requirements, underscoring the need for a planned, documented,
and carefully managed assessment of proposed COTS software.

2.2 Available Guidance

As a result of the recent trend toward cost reduction and productivity improvement, the topic
of software reuse has been extensively discussed in various industries including the software
engineering, defense, and nuclear communities. Numerous articles have appeared in the literature
and standards organizations have updated, or are currently updating, their guidance on the topic.

In the U. S. nuclear arena, references to the topic include the standard, IEEE 7-4.3.2-1993
(IEEE, 1993), and reports by the Electric Power Research Institute (EPRI, 1996) and by
Lawrence Livermore National Laboratory (LLNL) (Preckshot and Scott, 1996)2. In general, the
guidance from all sources is similar but the perspectives are different. IEEE Std 7-4.3.2-1993
presents requirements for software issues, including COTS, within the context of the nuclear
the context of “the relationship between the existing methods for commercial dedication and the
issues that should be addressed for software-based equipment.” The LLNL report focuses on
technical evaluation criteria synthesized from guidance in a variety of standards. U.S. NRC
review guidance on COTS is contained in (USNRC, 1997).

3. Important Aspects of an Assessment Process

A number of important aspects of an assessment process for determining the acceptability of
COTS software have emerged from the various investigations of the topic. The following
sections identify and discuss key topics from the perspective of COTS software usage in high-
integrity systems.

3.1 Grading of the Assessment Effort

Efficiency dictates that efforts to achieve safety should be commensurate with risk exposure.
The rationale for grading is that extraordinary efforts should not be made to manage an
insignificant risk, but if a significant risk is present, appropriate risk management efforts should
be employed. For high integrity systems, it is important to make the assessment effort
commensurate with importance to safety. This has two elements: consequence grading, which is
based on the impact associated with potential failure, and grading with respect to the relative
importance to safety of constituent components. The former is related to the overall extent of risk
reduction effort necessary to prevent the realization of hazards. Within the context of the
consequence category, the latter is related to the maximum allowable probability of component
or subsystem failure.

2 This information is presented here to identify important references; it is not within the scope of this paper to
present or compare the specifics contained in the references.
Four consequence levels are typically recognized:

- Failures with the potential for significant off-site consequences, such as consequences having a major impact on the public.

- Failures with the potential for significant on-site consequences. Possible examples are consequences involving on-site fatalities, loss of mission, or economic disaster for the owner/operator.

- Failures with the potential for only localized consequences. Possible examples are consequences involving a small number of recoverable injuries or survivable economic impact on the owner/operator.

- Failures having negligible impact, such as minor delays or inconveniences.

For high-integrity systems, the highest category is usually the one of interest. When grading components or subsystems with respect to relative importance to safety, consideration of the specific application area might allow specific types of subsystems to be placed in categories for which varying levels of assurance would be required. An example of grading with respect to importance to safety is found in the IEC 1226 standard (IEC, 1993) in which categories A, B, C, and Unclassified are established for types of systems found in nuclear power plants. Examples of nuclear systems in each category are, respectively,

- Safety shutdown systems

- Control systems

- Some warning and alarm systems

- Non-safety systems

The general benefit of applying a valid and effective grading process is that each system will receive the level of assurance effort that is appropriate for that system. Systems that have some importance to safety will be treated as such and not as uncontrolled developments. Systems that are unimportant to safety will not have excessive assurance efforts applied. Grading is especially important with respect to determining the acceptability of COTS products. As the magnitude of potential consequences decreases or as the safety role of a particular subsystem decreases, the potential impact of unknowns related to the alternative acceptance criteria discussed above is lessened. Therefore, the developer and regulator have more flexibility with respect to the basis for acceptance of the COTS software product.

### 3.2 System-Level Assessment

The issues facing a safety-system designer who would like to use a COTS product fall into a natural order if the rigor of the assessment process is to be commensurate with the risks associated with system use. A goal of any assessment process should be to eliminate unsuitable options as soon as possible in order to reduce time spent on untenable candidates. This requires starting from the viewpoint of the overall system and its risks, deriving the functional requirements the COTS product must satisfy, and then proceeding to product-specific issues. Risk identification comes first, followed by product identification, and finally determination of the rigor needed in the assessment steps to provide adequate confidence that the COTS product is appropriate for its intended application.

The first step taken in the design of a high integrity system should be some sort of hazard analysis that identifies potential high-consequence failures. For example, an architecture may be proposed in which one or more COTS products play a role in detecting, preventing, or mitigating failures. Probabilistic risk assessment (PRA) is a typical type of analysis that would be used to identify the functions the system must perform. This analysis uses fault trees to model subsystems, and event trees to model accident progressions. One of the outcomes of the analysis
should be identification of the safety functions for which specific subsystems are responsible. For subsystems incorporating COTS products, the safety functions the products must provide become the basis for assessment of candidate products. The PRA should also determine whether diverse means will be necessary to ensure the performance of a critical set of safety functions.

Once the environment for the COTS product has been established in terms of the safety functions to be performed and the nature of the product’s interfaces to other system components, the next step is to assess the COTS product itself in order to establish sufficient confidence that it will perform the intended safety functions, and that it will interface with other system components in ways that will not lead to unsafe conditions. It is crucially important at this point that the COTS product be under strict configuration control. The product must be precisely identified and all pertinent documentation must be clearly linked to the exact version and configuration of the product being assessed. The assessment cannot be done on a product that is essentially a “moving target.” With respect to the performance of safety functions, the assessment seeks to determine that the product characteristics provide sufficient confidence that the functions will be accomplished. Quality attributes such as correctness, performance, and reliability are assessed. With respect to system interactions, it is necessary to determine that the product will not interfere with the performance of system safety functions. Interference could occur because of abnormal behavior within the COTS product or because of inappropriate responses to external abnormal conditions or events. The presence of unused functions (legitimate features of the COTS product that are not needed for the safety-related application) in most COTS products requires that special assessments be performed to ensure that inadvertent activation of these functions is prevented or, if they are activated, that they do not interfere with system safety functions.

3.3 Product Assessment vs. Process Assessment

Careful examination of a COTS product is essential to assess its acceptability for use in a high-integrity system. Three sources for information should be used in the assessment: product development records, operating experience data, and post-development product testing. Product development records include specifications, code, V&V records, etc. Experience data can provide information about reliability, correctness, and performance; however, care must be taken to ensure that the data are valid. Data based on other versions or different configurations may not be applicable. Post-development product testing can be performed to obtain additional assessment data. This testing might be done to address minor deficiencies in development records or to address special requirements associated with the planned use of the COTS software in the system.

It is also important to assess the development processes that were applied during product development. This assessment helps to establish the validity and value of the product development records. Product information is not very useful without confidence that the processes that generated the information are comparable to commonly accepted practices for developing high-integrity software. Various methods have been developed to assess these processes, such as the SEI CMM (Software Engineering Institute Capability Maturity Model), the ISO 9000 certifications, or the Air Force’s Software Development Capability Evaluation. Certifications from such assessments are of interest in COTS assessment but do not substitute for a COTS assessment process. It is important to note that certifications from past assessments are perishable and that those past assessments will not encompass everything of interest with respect to COTS acceptability determinations. In addition to software development process assessment, a number of organizational indicators can also provide important information (Lawrence and Preckshot, 1994) and should be considered in the assessment process.

3.4 Difficult Issues in Assessing COTS Software

A number of issues that are particularly difficult to address have surfaced with respect to the determination of the acceptability of a COTS software product for use in high-integrity systems.
Missing Documentation

Considerable controversy rages about the acceptability of various approaches to replacing, reconstructing, or substituting for missing COTS product documentation. Some viewpoints focus on the question of applying engineering judgment to these and other assessment questions. It is necessary that engineering judgment be applied carefully to specific, narrowly defined questions, and that the rationale for the judgment be documented and able to withstand critical, external scrutiny. The documentation should be such that comparably qualified reviewers would reach the same conclusion. Engineering judgment cannot be used to justify generic, hand-waving arguments about COTS product acceptability.

One proposed alternative to the lack of documentation is operational experience. It seems intuitive that experience data derived from extensive usage of the same version of a product in similar applications would indicate that the product is acceptable for the intended application. There are several problems with this assumption. First, configuration data is rarely supplied and version data is often missing. Therefore, the statistical validity of the data is unknown. Large bases of experience data usually span releases and configurations of a given product. The second problem is that circumstances surrounding the occurrence, monitoring, and recording of failures are often vaguely reported. These activities are not uniformly controlled; in fact, there might be some motivation for COTS vendors to limit publication of negative experience. Another key issue regarding operating experience is that extensively used products can still have crucial faults that could cause problems in safety systems. The tendency is to consider operational experience to be similar to random testing. In this regard, operational experience suffers from the same shortcoming as testing: testing cannot prove the absence of faults.

A second proposed alternative acceptance criterion is using post-development testing to compensate for a major lack of product development documentation. However, as noted above, testing cannot confirm the absence of faults in software. In addition, it is difficult to use testing to detect unintended functions. Testing can, however, demonstrate that intended functions are implemented and that anticipated error conditions are handled properly.

A third approach to coping with missing data is to perform data reconstruction with methods such as reverse engineering. This can be a difficult task requiring as much effort as doing a new development and, even if accomplished, it is not clear that the reconstruction process is error-free.

In general, the approach of compensating for missing information by using information from other sources involves research questions or data validity problems. One must know exactly how one source of information replaces information from another. Until this is known, there is considerable risk in accepting a COTS product for use in a high-integrity system based on such techniques.

However, compensation is more acceptable as risk exposure decreases.

Unexpected Functions

A key element of an assessment process for high-integrity systems is the verification that system safety analyses have been performed and that the functions and characteristics important to safety that are applicable to a COTS component have been determined. It is then possible to verify from testing or examination of vendor testing records that these intended functions will be accomplished by the COTS component. Unfortunately, software too often behaves in unexpected ways, i.e., performs unexpected functions. Unexpected functions are either unused functions (functions intended by the developer but not intended to be used in the application) or unintended functions (unplanned functions arising from design errors or implementation errors). It is much more difficult to assure that no unintended functions will be performed since software testing is ineffective against this risk. The probability that unexpected functions would be present in a new software development for a specific application is much less than for the COTS software product, since a typical goal for commercial products is to maximize features and configurability while minimizing development time. The concern is that unexpected functions
might affect the performance of system safety functions if the functions are inadvertently activated. Unexpected functions are a concern in system-level COTS product assessment, discussed above.

**Operational Profiles**

In order to evaluate the COTS software in performing its assigned safety functions and interacting with the overall system, it is necessary to understand the operational profile of the software’s role in the system. This profile includes items such as the nature of inputs and outputs, transaction loads, performance requirements, and reliability requirements. For new developments, this information is included in the product specifications and the product is designed to fit the profile. For COTS products, the operational profile assumed in its design might not be known precisely. Consequently, it is difficult to assess where discrepancies between the system design and the specification for COTS usage within that system might appear. This could be particularly important in situations where the software must operate outside of its normal parameters.

**Version Control**

Two key considerations arise regarding version control. The first is that a COTS product to be evaluated for use in a high-integrity system must be identified by version. All of the information pertaining to the COTS item to be used in the assessment must be clearly linked to the specific version to be incorporated into the system. Different versions require separate assessments. It is also possible that a COTS product is so highly configurable that information from several sources regarding the same version will not be equally valid for the COTS acceptability determination.

The second consideration is the question of how bug fixes and new releases will be handled once the COTS software has been integrated into a system. Also, who will perform the modifications, and how will the acceptability of modified software be determined? These are particularly troublesome questions in cases where software was accepted under alternative acceptance criteria because product information was not available. It may be possible to freeze versions for a period of time, but eventually, system evolution or vendor support requirements will generate a need for changes.

**Site-Specific Configuration**

The COTS product must be evaluated in the light of a full range of system configuration questions. New developments would be designed for the specific site configuration requirements. Considerations include site requirements and constraints, platform configurations, configurations of layered or interfacing software, and possible configurations of the COTS product itself.

**Error Reporting**

Error reporting is a key issue with respect to the use of commercial software in safety-related systems. Information on errors encountered in the past is necessary to help in evaluating available operational experience. After the COTS product has been incorporated into a system, two-way communication between the COTS software developer and COTS software users is necessary to ensure that known or recently discovered errors are documented for evaluation of their potential impact on high-consequence systems, and to ensure that additional information can be obtained from the software developer if anomalies are observed during system operations. Long-term relationships of this type can be difficult to establish and maintain with vendors of commercial products.
V. Conclusion

Given the current state of the art in software engineering and the technical, political, and maintenance considerations associated with typical commercial software, it appears that it will be difficult at best to incorporate COTS software products into high-integrity systems. If an effective grading process is in place, however, the tradeoffs associated with the use of COTS products in lower risk categories become more palatable.

Although it appears initially to be much cheaper to purchase COTS software than to perform new developments, the costs associated with an acceptance process and with maintaining the acceptance through necessary software revisions can be quite high. The decision to employ a COTS product, if it can be successfully assessed, should be supported by a cost-benefit analysis indicating overall benefit. Acceptance and maintenance issues, and their associated costs, become less important as importance to safety decreases.

It is possible that some COTS software products can be demonstrated to be adequate for use in high-integrity systems. Some software vendors, such as programmable logic controller vendors, produce software with the knowledge that it will be used in systems with medium to high risks. Some of these vendors use software processes that have been designed to produce high-integrity software. They are generally aware of the types of hazards associated with the systems in which their products will be used and those hazards have been considered in their designs. In order to meet the demands of the high-integrity marketplace, they may be motivated to form long-term partnerships with users and to supply additional reliability documentation. Such vendors may well be in a position to meet applicable acceptance and regulatory requirements for the use of their products in high-integrity systems.

VI. References


