COTS Software Evaluation Techniques

John C. Dean, CD, B.Sc, M.Math
National Research Council Canada
Software Engineering Group
Building M-50, Montreal Road
Ottawa, Canada K1A 0R6
+1 613 991 6975
John.Dean@nrc.ca

Dr. Mark R. Vigder, PhD
National Research Council Canada
Software Engineering Group
Building M-50, Montreal Road
Ottawa, Canada K1A 0R6
+1 613 991 6972
Mark.Vigder@nrc.ca

ABSTRACT
Employing Commercial Off-the-Shelf (COTS) software products as components in large-scale long-lived systems has been proposed as a way to reduce both implementation and operating cost for the user communities. While this may be the case, the actual benefits have not been confirmed. However, there is factual evidence that some of the suggested cost savings will be offset by the need to address a new set of issues that are raised by the inclusion of COTS components. One of these is the need to evaluate candidates COTS systems early in the development life cycle. Our research is concentrated in the area of physical evaluation of candidate products, that is, actual testing of the products themselves.

The purpose of this paper is to present a discussion of proposed evaluation techniques used to select COTS software components for systems development, to describe appropriate testing techniques for COTS candidates, and to propose an evaluation system which will provide support to ensure timely selection of suitable COTS products.

Keywords
Commercial Off-The-Shelf, COTS, software, evaluation

1 INTRODUCTION

In modern COTS-based systems development we need to evaluate the candidate COTS components at an extremely early stage in the development process. At this stage requirements are generally less than completely defined and often provide only the most general guidance to the evaluator. As with any modern system, the requirements evolve over time. The fundamental difference in a COTS based system is that COTS capabilities have been shown to influence requirements[3,4,10] decisions and thus the evaluation process is inextricably linked to requirements definition.

Some of the proposed COTS evaluation methods have proven to be less than successful because they are based on traditional development paradigms which, while applicable to systems built from first principles, have not been able to easily accommodate COTS software components. Many of these paradigms rely on a highly structured requirements definition and specification that sets the criteria for COTS selection. As such they are slow to react to the fast changing commercial marketplace

Other proposed evaluation processes depend on the pre-qualification of COTS components. With these schemes the developer selects from lists of qualified or certified components which have undergone extensive generic laboratory testing. These components are then incorporated into the current development. The developer must rely on in-context evaluation to ascertain specific knowledge about each candidate COTS software product.

An alternative methodology is one in which the COTS software selection and evaluation influences and is conducted concurrently with the requirement definition process. This approach has advantages in terms of cost and time because it results in a more directed evaluation of components and because it reduces implementation complexity.

2 EVALUATION OF COTS PRODUCTS

Oberndorf et al [15] provide a general background discussion of the issues involved in selecting and evaluating COTS products. In particular, they stress that in-context evaluation is necessary for any reasonable hope of successful evaluation. In context evaluation implies that evaluations are conducted within the scope of the systems to be conducted as opposed to out-of-context evaluation that is conducted against a set of generic criteria.

Current literature provides a number of methods for the evaluation of COTS components. Each of these methods emphasizes one or more critical aspects of COTS software evaluation. This section will discuss highlights of these proposed techniques. This is not meant to be a recommendation as to the validity of these methods, but only an overview. The overriding goal is to identify those aspects of the methodologies that might be useful in developing an integrated approach to evaluation. The information is drawn from a broad range of fields, some of which have different goals than COTS-based systems development, but the information is still pertinent.

2.1. COTS-based Integrated System Development (CISD) method

Tran and Liu [16] propose, within the CISD model, a two stage COTS selection process. The first stage is product identification, where candidates are identified and classified. The data for this stage is gathered via vendor documentation, personal experience or other means. The results are a list of potential candidates. The second is evaluation, where the final candidates are chosen (and unsuitable candidates eliminated). In this stage the authors depend on concrete techniques. They state that the COTS evaluation phase requires the extensive use of prototyping techniques. They argue that prototyping is the only way to practically evaluate a COTS candidate within the systems context. They define three critical stages of the evaluation phase; functionality, interoperability, and performance. In the functionality phase the candidates are tested in isolation to confirm that the functionality of the COTS product is applicable to the current application. In the interoperability stage, the candidates are evaluated to ensure their ability to co-exist with other components of the system, both COTS-based and custom developed. The performance evaluation stage consists of a quantitative analysis of the effect of the COTS component on the overall performance of the system.

The final aspect of the methodology is a management evaluation that considers the less tangible aspects of integrating the COTS product. These include such things as training, cost, vendor capability, etc. At the end of this process a final selection of COTS products is made.

The authors also discuss different approaches to evaluation based on constraints such as development time and cost. Two that they highlight are the Comprehensive Evaluation (CE) approach and the First-Fit Evaluation (FE) approach. The result of CE is a list of the most optimal COTS product sets while the result of FE is the first product set which fulfills the requirements. They state that FE is the more cost-effective approach.

Note that this methodology depends on having a relatively complete predefined set of requirements since the product identification stage is dependent on COTS candidates meeting the requirements. The methodology in general is a waterfall-style process in that each stage depends on the results of its predecessor.

2.2. Off-The-Shelf-Option (OTSO)

Kontio et al.[8,9], present a multi-phase approach to COTS selection which begins during requirements solicitation. With their approach the decision to incorporate COTS into the system has been predetermined and thus the OTSO method is only concerned with the actual selection process, not with implementation. The phases are the search phase, the screening and evaluation phase and the analysis phase. In the search phase COTS candidates are identified. At this time the requirements are not fully specified and, in fact, may be quite vague. The screening and evaluation phase narrows the field of potential candidates.

During both these phases the extension of understanding of the product capabilities provides feedback to the requirements definition process. This results in a refinement or modification of known requirements as well as the introduction of new requirements. Evaluations are always performed against a set of evaluation criteria which are established from a number of sources, including the requirements specification, the high level design specification, the project plan, etc.

The final phase of the selection process is the analysis of the results of the evaluation. This leads to the final selection if COTS products for inclusion in the system.

The central theme to the OTSO method is the construction of a "product evaluation criteria hierarchy". This hierarchy serves as a template for situation specific criteria definition.

The conclusions that the authors reach are that criteria definition must be revisited for each project because each project evolves in a different environment at different times. This again implies that evaluation is context-dependent. The OTSO process is iterative because the requirements are both refined and defined throughout the course of the evaluation stage.

2.3. Checklist Driven Software Evaluation Methodology (CDSEM)

Jeanrenaud and Romanazzi[6] present a methodology for evaluating software that employs checklists, which they use to determine a quality metric for each item in the checklist. The process is metric based and provides a numerical result that describes the suitability of the component. This approach is very attractive because it quantifies the evaluation results, however the authors base some of their discussion on the availability of source code and access to individual modules, neither of which are usually available in a COTS product. They also depend heavily on the vendor documentation and demonstrations for supporting data as opposed to in-context, practical evaluation. This may lead to the adoption of unsuitable candidates.

Mcdougall and Squires[13] present arguments why this approach is not necessarily effective as a selection process.

2.4. Procurement-Oriented Requirements Engineering (PORE)

Maiden and Ncube[10,11] propose a template approach to requirements definition that depends on evaluating COTS products. They initially suggest requirements need to be reasonably defined in order to be able to start evaluating COTS products. The process they describe, however, is one in which requirements are defined in parallel with COTS component evaluation and selection.

Within their discussion of lessons learned they highlight that software prototypes are useful in developing knowledge concerning COTS products and their interactions within the overall system. They stress that the selection process needs to proactively evaluate the actual product and not rely exclusively on the vendor-supplied documentation or demonstration.

Although they are directed towards requirements acquisition, the sample templates give a preliminary view of some of the steps needed to perform a justifiable evaluation of candidate COTS applications.

3 TESTING TECHNIQUES FOR COTS EVALUATION

Evaluation of candidate products requires that we adopt some technique to prove the capabilities that interest us. In traditional software development there are two accepted methods of testing software products. They are white box and black box testing. It is not clear that both of these techniques can be applied effectively in the case of COTS software-based systems since both the available documentation and the goals of COTS evaluation are different.

With COTS-based systems there are a number of unique constraints on our ability to conduct effective testing. In general we assume that we have no access to the source code or, in the case where it is available it cannot be modified. This means that we cannot internally instrument the executable. Most vendor documentation that is available consists of user manuals and advertising materials and is not directed at evaluating the operation of the system. For example, it does not describe the behaviour of the system in response to abnormal input. Finally, in COTS-intensive systems much of our use of these products is under non-standard conditions so the testing focus must be skewed towards unique situations.

These constraints influence the goals we are attempting to accomplish with COTS evaluation. Much of our test strategy is directed towards discovery of behaviour under system imposed conditions. We also need to confirm that the product adheres to specifications supplied by the vendor and that it can operate within the system environment, particularly as this pertains to product interoperability. We want to determine if we are able to mask out unwanted functionality as well.

3.1. White Box Testing

White box testing relies on the ability of the tester to examine the internal operations of the software at the source code level. One of the accepted white box testing methods is basis-path testing where an attempt is made to exercise each independent path through a code module. There are a number of interesting ways of determining the independent paths. This type of testing is usually undertaken during actual software development while code is being actively constructed. This corresponds well to the concept of verification testing which confirms that functionality of a system is implemented correctly

3.2. Black Box Testing

Black box testing is designed to allow the tester to treat each code module as a unit which can be defined by its' inputs and outputs (the interfaces to the module) without regard to the route by which an input is transformed into a particular output. With this method visibility into the internal workings of the code module is not necessary and thus the source code is not required. An example of the methods used during black box testing is boundary value analysis where inputs are supplied to the module under test which represent valid, invalid and boundary values. The outputs are then measured and accepted if they fall in the expected output range. The black box type of testing is normally carried out during system integration or after the completion of the coding of a module. This type of testing also is seen during acceptance testing and is considered to be the foundation of validation testing which confirms that the software actually performs the required functions.

The physical testing of COTS candidates is necessarily constrained by the fact that the source code is not available. Some of the testing is for discovery of undocumented features and/or bugs while other testing involves confirming or denying the published vendor data and specifications. Both of these can be seen to be a special case of validation; the first because we are trying to increase our understanding of the candidate under evaluation, and the second to attempt to confirm the vendors claims as to the effectiveness of the COTS product. The various black box techniques seem to be ideal for these purposes since we do not have the visibility into the system that white box testing requires.

Test Methods

One of the recommended methods for evaluating COTS products is to employ scenario-based testing methods. With this method a portfolio of scenarios is created. Note that the scenarios represent typical operating procedures for the system that is to be constructed, not for the COTS product under test. Test procedures are developed based on the scenarios and each candidate is evaluated against the criteria. In this case the initial scenarios are reasonably easily established using the preliminary operational requirements definitions. The results of this type of testing will be confirmation that the qualified candidates perform appropriately in the system context.

Another method that has been suggested by Voas[17] is the use of fault injection techniques. This is particularly effective when access to the internal operations of a product is restricted. The method consists of inserting erroneous values into the data and/or control stream and observing the results. This technique is a good example of evaluating for discovery, that is, to determine unknown or unexpected reactions of the product under evaluation.

4 A PROACTIVE EVALUATION TECHNIQUE

The technique that we have developed during the implementation of our prototype relies neither on a strict requirements definition nor on pre-qualification of COTS products. Rather, it combines the most effective processes of all of the above models.

4.1. The Concept

We begin with a generalized statement of requirements in which we describe only the overall concept of the system to be developed. This initial requirements definition draws much from the operational needs of the users and less from the technical descriptions. We take this approach because we do not, at this early stage, want to eliminate any possible solution to the problem. This allows great flexibility in selection of appropriate COTS software. Only during the detailed evaluation stage do we establish more restrictive technical criteria. Using our prototype system as an example, only after surveying the marketplace for appropriate tools to transfer data, did we select the hypertext transport protocol as the primary transport mechanism.

The next step after gathering initial requirements is to survey the marketplace to determine which candidate COTS components exist that exhibit capabilities compatible with the generalized requirements. We are not attempting to find all the available candidates but only a reasonable selection. Choosing the initial candidates based on generic capabilities somewhat eliminates the competitive aspects of the survey. Should we find a candidate that appears to be an ideal fit we could select that component without further comparison. Our experience has been that we can find one specific component in about ninety percent of the cases without requiring a pre-qualification stage. During the market survey we continue to redefine the requirements based on the knowledge we gain about available products. The information may lead to the addition of or possibly to the removal of requirements.

After choosing the candidates, we analyze existing documentation to determine what advertised capabilities exist that we might require within our proposed system. At this stage we would assume that any of the candidates under consideration could perform adequately in the role. The COTS component is tested to ensure that it indeed performs within its documented parameters. Exceptions are noted but these exceptions do not necessarily eliminate the component. That would occur only if an exception would cause significant harm within our usage context. We do not attempt to assess the undocumented features of a component. This preliminary evaluation is only to determine that the documentation is accurate and that the candidates actually perform as documented.

We then begin a more detailed evaluation of the component by creating a system based test harness and exercising the component within the context of our application. Maiden[10] argues the case for a scenario based testing process as a reasonable and effective testing mechanism. Likely operational scenarios are determined and documented and the candidate is then subjected to operation under the established scenario

Much of this evaluation is conducted using prototype implementation. The actual test suites that are applied to the components are derived from the requirements. Test suites are designed to examine the limits of the product under test. The prototypes are made successively more capable until we are confident that all of the needed functionality of the COTS component has been examined. This evaluation is meant to establish the operating bounds of the component and to enable us to begin to refine our requirements to fall in line with the capabilities of the candidate. We also attempt to determine ways to mask out currently unneeded capabilities.

Finally we define any local enhancements which may be needed to supplement the capabilities of the component. The enhancements are necessary to provide for critical requirements that cannot be implemented using COTS components. These will be implemented in the wrappers and/or the glue code of the system. If a critical requirement is such that an entire subsystem needs to be implemented in-house, that subsystem will be designed and coded so that it can be integrated as a COTS product.

Advantages

The advantages to this approach are significant, particularly in the early stages of development. By restricting the evaluation and testing to the specific needs of the current system we eliminate the direct pre-qualification requirements completely. This allows us to concentrate our efforts on deriving a limited set of tests that exercise only the interesting capabilities of the candidate.

In-context testing ensures that the candidate is suitable for this particular project. The testing is extremely focussed and definitely goal-directed. The actual test cases are designed to exercise only those aspects of the COTS component that will be used for this application. The testing relies heavily on a Black Box approach since the internal operation of the component is usually unknown. Even if we do have access to source code, the goal is to use the COTS software without modification and therefore we must assume that White-Box techniques will not provide useful information. An expanded form of Boundary Value Analysis can supply all the information that we require. This is not to suggest that the testing process is somehow incomplete. We follow the same rigorous approach that we would when planning and implementing testing for a traditional development model but here we emphasize the integration aspects of testing.

We evaluate only existing, current versions, because our evaluation takes place closer to implementation. This ensures that we do not have to repeat the evaluation prior to implementation. We also only evaluate those components that we realistically believe can be used in our system. We select a subset of available components for evaluation by a process of examining candidates only until we feel we have a representative selection. If one candidate appears to be ideal (i.e. it is capable of filling the majority (the 80-20 rule) of the requirements) we do not seek other alternate solutions unless there is some constraint applied. This reduces the number of components that must be evaluated. It also leads to a truly independent assessment of the candidate's capabilities because it reduces the competitive and/or comparative nature of the pre-qualification process.

A further advantage is that we can use the acquired product understanding in future projects to aid in selecting appropriate candidates. We construct our understanding via evolution rather than monolithically. We also gain an understanding of the critical aspects of a candidate COTS product.

Finally this approach fits requirements to COTS capabilities. This leads to a more comprehensive match between the COTS components and the final system requirements. This is not to say that only the capabilities exhibited by the COTS components will appear in the final product; rather, it forces the system integrator to consider carefully whether a particular stated requirement is actually necessary or whether it can be eliminated. Those requirements outside the capability sphere of the candidate COTS products, but deemed necessary, will be met by constructing a component in-house. Implementation of these in-house components follows traditional development processes, except that the component is then integrated into the system in a similar manner to a COTS product. This ensures architecturally consistency throughout the system.

5 CONCLUSIONS

This paper outlines a number of proposed evaluation and selection techniques for choosing appropriate COTS software products for incorporation in large-scale systems. The advantages and disadvantages of each are outlined. We have then proposed a process for the evaluation of COTS software products that takes advantage of the best processes of the different methods as well as introducing new techniques.

Current testing practices as applied to conventional development were examined and their applicability to COTS development highlighted. It is obvious that Black Box techniques are mandatory during in-context evaluation of software but also that the goals of testing are somewhat different from the traditional ones. Scenario based testing provides a good basis for evaluating candidate products. The results obtained from evaluation testing can be used as validation data for system testing.

6 ACKNOWLEDGEMENTS

The research described by this paper has been jointly supported by the Defence Research and Development Branch of the Department of National Defence Canada and the National Research Council Canada.

7 REFERENCES

1. Evan E. Anderson, A Heuristic for Software Evaluation and Selection. Software Practice and Experience, 19(8):707-717, 1989.
2. Jean Bergeron et. al.. Detection of Malicious Code in COTS Software. A Short Survey. In International Software Assurance Certification Conference, ISACC'991998.
3. John C. Dean and Mark R. Vigder. System Implementation Using Off-the-shelf Software. In 9th Annual Software Technology Conference1997.
4. G. Fox and S. Marcom. A Software Development Process for COTS-based Information System Infrastructure. In Fifth International Symposium on Assessment of Software Tools and Technologies, pp133-142, Jun 1997.
5. David Garlan and Robert Allen and John Ockerbloom. Architectural Mismatch or Why it's hard to build systems out of existing parts. In 17th International Conference on Software Engineering, pp179-185, 1995.
6. J. Jeanrenaud and P. Romanazzi. Software Product Evaluation: A Methodological Approach. In Software Quality Management II:Building Software into Quality, pp59-69, 1994.
7. I.M. Klopping and C.F. Bolgiano. Effective evaluation of off-the-shelf microcomputer software. Office Systems Research Journal, 9(1):46-50, 1994.
8. Jyrki Kontio. A Case Study in Applying a Systematic Method for COTS Selection. In 18th International Conference on Software Engineering, pp201-209, 1996.
9. Jyrki Kontio and Gianluigi Caldiera and Victor R. Basili. Defining Factors, Goals and Criteria for Reusable Component Evaluation. In Proceedings of CASCON '96, pp17-28, Nov 1996.
10. Neil A.M. Maiden and Cornelius Ncube and Andrew Moore. Lessons learned during the requirements acquisition for COTS systems. Communications of the ACM, 40(12):21-25, Dec 1997.
11. Neil A. Maiden and Cornelius Ncube. Acquiring COTS Software Selection Requirements. IEEE Software, 15(2):46-56, Mar 1998.
12. Jean Mayrand and François Coallier. System Acquisition Based on Software Product Assessment. In

18th International Conference on Software Engineering, pp210-219, 1996.

13. Anne McDougall and David Squires. A Critical Examination of the Checklist Approach in Software selection. Journal of Educational Computing Research, Vol 12(3):263-274, 1995

14. National Research Council COTS Web Site Available at: http://wwwsel.iit.nrc.ca/projects/COTS/COTSpg.html

15. Patricia Oberndorf and Lisa Brownsword and Ed Morris and Carol Sledge. Workshop on COTS-Based Systems. SEI Special Report CMU/SEI-97-SR-019. 1997.

16. Vu Tran and Dar-Biau Liu. A Risk-Mitigating Model for the Development of Reliable and Maintainable Large-Scale Commercial-Off-The-Shelf Integrated Software Systems. In Proceedings of the 1997 Annual Reliability and Maintainability Symposium, pp361-67, Jan 1997.

17. Jeffrey Voas. Error Propagation Analysis For COTS Systems. Computing and Control Engineering Journal, 8(6):269-72, Dec 1997.