Timing the Testing of COTS Software Products*

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

Abstract
The use of Commercial-Off-The-Shelf (COTS) software products in the construction of distributed large-scale systems implies the need for a new approach to systems testing. The evaluation process for COTS candidate selection begins at the same time as the establishments of initial requirements. This position paper presents arguments for the establishment of a testing protocol in the early stages of evaluation and for the implementation of effective testing techniques that are appropriate to a COTS-based system. Test planning and implementation must account for the use of results obtained during the evaluation process.

Keywords
COTS, product evaluation, testing, test planning

1. INTRODUCTION

In every discussion of the implementation of COTS-intensive software systems it quickly becomes obvious that early evaluation of candidates COTS products is one of the key aspects of the system development life cycle. The definition of requirements for the system as a whole can be significantly affected by the results of this evaluation. Thus the success of the entire development depends on an accurate understanding of the capabilities and limitations of the individual COTS. This dependency can be quantified by implementing a test suite that uncovers interoperability problems, as well as highlighting individual characteristics. These tests represent a formal evaluation of the COTS capabilities and, when considered within the overall system context can represent a major portion of subsystem testing. Given that this is the case, true testing of the system can start before the requirements are strictly specified because the test suite created for evaluation can form a basis for formal testing of the system.

This position paper will outline some of the basics of current evaluation approaches and suggest appropriate modifications that allow for the use of evaluation results as test cases. Note that, in this discussion, we are concerned specifically with evaluation and testing of the COTS products and not with the testing of wrapper or glue code that may be required. In addition, while this discussion is applicable to distributed systems, it is equally pertinent within the context of general COTS software development. Section 2 of the paper outlines a number of current approaches and describes the advantages and disadvantages of each. Section 3 discusses the applicability of current testing techniques to COTS-solution or COTS-intensive systems. Section 4 provides some conclusions and suggestions concerning open research questions.

2. EVALUATION OF COTS PRODUCTS

Oberndorf et al.[1], 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 means that the candidate product is evaluated in the context of the current development as opposed to generic evaluation. In-context evaluation lends itself to the development of specific test procedures and cases whose results, in the long term, may be used as data for the demonstration of compliance during formal testing.

In order to acquire a reasonable level of confidence in the results of any evaluation there must be a rigorous methodology in place to guide the testing of the product. The methodology must include the accepted testing and test documentation practices. This would include the establishment of test cases and the recording of test results.

Current literature provides a number of methods that could be applied to the evaluation of COTS products.

---

*NRC Report #41625

This section will discuss highlights of these techniques. This is not meant to be a recommendation as to the validity of these methods, but only an overview. 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.

COTS-based Integrated System Development (CISD) method

Tran and Liu[2] 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 outcome of the stage is a list of potential candidates. The results of this stage are of less value as reusable test results because they are in many cases based on hearsay rather than empirical results. For example, at this stage vendor documentation that states that the product meets certain constraints might be used as a classification criterion without being confirmed.

The second stage is product *evaluation*, where the final candidates are chosen (and unsuitable candidates eliminated). In this stage the actual evaluation depends on concrete test criteria and accepted testing techniques. Extensive use of prototyping techniques is recommended. Employing prototyping is the only way to practically evaluate a COTS candidate within the systems context. There are 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 suitable 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 this stage 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.

Different approaches to evaluation based on constraints such as development time and cost are examined. Two that are highlighted are the Comprehensive Evaluation (CE) approach and the First-Fit Evaluation (FE) approach. With the CE approach multiple products must be evaluated on a somewhat comparative basis. All candidates are subjected to the same evaluation criteria. The result of CE is a list of the most optimal COTS product sets.

In the Fe approach the result of FE is the first product set which fulfills the requirements. FE is the more cost-effective approach because the evaluation is limited to a subset of competing COTS products. Employing FE automatically implies that some candidates may not be evaluated at all.

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. This is a drawback when the requirements are less than fully defined. The methodology in general is a waterfall-style process in that each stage depends on the results of its predecessor.

The quantifiable results obtained during stage two of the CISD process provide initial data that can be relied on during later testing as a starting point. These results can be used as is and those tests need not be re-run during formal testing.

Off-The-Shelf-Option (OTSO)

Kontio et al.[3], present a multi-phase approach to COTS selection which begins during requirements solicitation. With this approach the decision to incorporate COTS into the system has been mandated 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. The central theme of the OTSO method is the construction of a "product evaluation criteria hierarchy". This hierarchy serves as a template for situation-specific criteria definition.

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. Evaluations are always performed against a set of evaluation criteria that are established from a number of sources, including the requirements specification, the high level design specification, and the project plan.

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.

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

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.

Checklist Driven Software Evaluation Methodology (CDSEM)

Jeanrenaud and Romanazzi[4] present a checklist-based methodology for evaluating software. A quality metric is determined 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 some of the discussion is based on access to source code and individual modules, neither of which are usually available in a COTS product. The method also depends 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. The output of the process is also a list of the relative merit of the competing COTS products.

Mcdougall and Squires[5] present arguments why this approach is not necessarily effective in the selection process. In addition, this method appears to concentrate on providing a global assessment of the quality of the product under consideration rather than acquiring specific data concerning test results. Thus the data may not be as valuable to the system testing process.

Procurement-Oriented Requirements Engineering (PORE)

Maiden and Ncube[6] 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 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. With this method there may be possibilities for forming a basis set of test results, however more work needs to be undertaken to validate the process.

In summary, each of these methodologies exhibits desirable qualities with respect to evaluating COTS products. The most promising appear to be those that rely on flexibility of requirements and the ability to adjust requirements to meet the capabilities of the COTS candidates. In addition, the most valuable evaluation results, as applicable to testing data, are those derived from the methodologies that use non-comparative evaluation suites.

3. TESTING TECHNIQUES FOR COTS

No matter what the final goals of testing, the actual testing of COTS candidates is necessarily constrained by the fact that the source code is not available. Some of the testing needed is discovery based, that is, searching for evidence of undocumented features and/or bugs. Other testing involves confirming or denying the published vendor data and specifications. A third aspect is testing to determine how well a product fits within the system environment. Each of these can be seen to be a special case of validation. In the first we are trying to increase our understanding of the candidate under evaluation, in the second we confirm the vendors claims as to the functionality of the COTS product, and in the third case we determine interference effects between products competing for resources.

Another aspect of testing is the need to determine our ability to mask out unwanted features of the COTS products and to ensure that these features are not allowed to interfere with the operation of the system as a whole. This differs from the traditional approach because a developer would not normally choose to incorporate unwanted functionality in a system. COTS products, on the other hand, can be expected to contain a significant level of this type of functionality because the product is being used under unforeseen conditions. The effectiveness with which superfluous functions can be hidden is a significant concern.

Evaluation of candidate products requires that we adopt some technique to examine and verify 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[7]. The following discussion outlines some aspects of the approaches.

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 that are generally based on counting decision points within the source code that define the number of branches. 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 (for example, values are calculated correctly).

It is apparent that white box testing is not appropriate in the case of COTS evaluation because we do not have the visibility into the source code. We must assume that the COTS product vendor has already conducted this type of testing. If so then our testing must be restricted to the confirmation of results rather than the examination of the method of achieving those results.

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 techniques used during black box testing is boundary value analysis where inputs representing valid, invalid and boundary values are supplied to the module under test. The outputs are then measured and accepted if they fall within 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 (according to the specification). Black box testing can also be used to assess interfaces between various products.

Given that the nature of a COTS solution is that the emphasis is on the integration of multiple "black-box" components, the majority of final testing takes the form of integration testing. In this case the various black box techniques seem to be reasonable for our purposes because, with individual COTS products, we do not have the internal visibility that white box testing requires.

4. CONCLUSIONS AND RECOMMENDATIONS

We have discussed a number of proposed evaluation and selection techniques for choosing appropriate COTS software products for incorporation in large-scale systems. Each of these has advantages and disadvantages when used as a basis for test suites. The two methodologies that exhibit the most promise as vehicles for acquiring test data appear to be CISD and PORE. These methodologies need to be reassessed in order to determine where processes can be adapted to provide the specific test data needed in later stages. Developing extensions to assist in recording test result data should be investigated further.

Current testing practices as applied to conventional development were briefly examined and their applicability to COTS development highlighted. It is obvious that black box techniques are mandatory during in-context evaluation of software. The approach to testing COTS-based systems is different than for traditionally implemented systems because the quite often only the functional aspects of the COTS products can be quantitatively measured. However, test data ranges may need to be expanded in order to reach the same level of confidence in the reliability of the system.

The results can provide a foundation for system validation and employed as data to support regression testing. Results obtained during evaluation do not need to be reconfirmed during system testing because we change neither the internal code of the COTS products nor the context in which they are integrated into the overall system.

The unique nature of a COTS-based system provides the developer an opportunity to gain earlier insight into the validation of the overall system because test results are obtained early in the development process. Early test planning is even more critical with these systems in order to take advantage of these results.

Bibliography

[1] Patricia A. Oberndorf, Lisa Brownsword, Ed Morris and Carol Sledge, *Workshop on COTS-Based Systems (Technology and Product Evaluation Breakout Group)*, SEI Special Report CMU/SEI-97-SR-019, November 1997. (http://www.sei.cmu.edu/publications/documents/97.reports/97sr019/97sr019chap03.htm)

[2] 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*, Proceedings of the Annual Reliability and Maintainability Symposium, 1997

[3] Jyrki Kontio, Gianluigi Caldiera and Victor R. Basili, *Defining Factors, Goals and Criteria for Reusable Component Evaluation*, Proceedings of CASCON '96, November 1996

[4] A. Jeanrenaud and P Romanazzi, *Software product evaluation metrics: a methodological approach*, Software Quality Management II: Building Quality into Software/Edinburgh UK, 1994

[5] Anne McDougall and David Squires, *A Critical Examination of the Checklist Approach to Software Selection*, Journal of Educational Computing Research, Volume 12, Number 3, 1995

[6] Neil A. Maiden and Cornelius Ncube, *Acquiring COTS Software Selection Requirements*, IEEE Software, Volume 15, Number2, 1998

[7] Roger S. Pressman, Software Engineering: A Practitioner's Approach, 4th Edition, McGraw-Hill, Pg., 454, 1998