Method for Enhancing the Process of Software Tool Evaluation and Selection: COTS, Heritage, and Custom Software Reviewed

D. J. Equils

Jet Propulsion Laboratory, Pasadena, CA, U.S.A.

For many organizations today, software evaluation and selection to meet the computing needs of that organization is critical. In recent years, the role of software has become essential to the integration of hardware systems of greater and greater complexity. As a result, the choice to purchase Commercial Off-The-Shelf software (COTS), adapt existing heritage or legacy software, or to develop custom software can mean literally the success or failure of project. Yet frequently this decision is delegated to an individual who is not in alignment with the system or organization as a whole. Decisions are too often based solely on personal preference, familiarity and experience with one product with little regard for peripheral information. So how does a project effectively evaluate the software available? What are the factors and criteria that a decision maker should focus on and understand to make the most logical and strategic choice for software solutions? These fundamental challenges can be found throughout organizations of any business in nearly every industry. As software becomes increasingly complex with advancements in Information Technology, the need for accurate tools, which will help decision-makers faced with this decision, also increases. Additionally, as standardization becomes more mainstream, companies in all industries are looking to COTS systems to cut costs, meet demanding schedule needs, and increase robustness. These espoused capabilities of COTS software can give organizations a competitive advantage but only if implemented in the appropriate way, at the appropriate time and with the appropriate resources. However, if a software selection and implementation strategy is executed without proper understanding of the project as a system, then any software decision can lead to cost overruns, delays in the schedule and even project failure. In today's competitive market place, this risk is unacceptable.

Through interviews with software decision-makers and managers, this paper has captured both successful empirical methods and lessons-learned from previous software selection processes. This paper concludes with a cost-effective and strategic approach via template for establishing accurate software selection criteria. With this template, decision-makers will be better prepared to make informed and comprehensive software decisions for specific systems, subsystems, and interfaces. Ultimately, by adopting this methodology, organizations will be able to achieve a Return-On-investment (ROI) through accurate and effective software selection processes.

Introduction

At JPL, all projects have one thing in common; computing needs for both the Flight System, which designs, assembles, and tests the Spacecraft, and the Mission System, which operates the spacecraft to meet its science objectives. These computing needs can be met in one of at least two fundamental ways; Build or Buy. Projects can build software applications and scripts from scratch, designing capabilities to exactly meet the system level requirements. Alternatively, projects can decide to buy or inherit and then adapt an existing piece of software, which can be modified in an attempt to meet the same requirements. The existing piece of software will come in one of two forms: COTS or Heritage.

Each method has pros and cons and this decision can be one of the most critical in a Projects Life Cycle. Additionally, each COTS option must be weighed against each other, comparing strengths and weaknesses. Historically, however, this decision is often made based on two to three criteria and the personal experience and preference of the Project Manager, Mission Operations Manager, and Mission Operations Systems Engineer. As this paper will show, there are scores of issues, which must be thoroughly examined and considered before a decision of this importance can be made. Through examining projects that have used both custom software and adapted COTS and heritage software, the results of those projects will be compared, looking at the software selection process, cost and schedule impacts, software performance, and lessons learned from the decision makers. This paper will attempt to analyze the decision making process through interviews of the managers involved. Where possible, independent verifications will be made of claims from decision-makers and secondary sources, such as documentation and project databases, will be checked.

Finally, with the collected information and lessons learned from the interviews, I have concluded with a template that decision-makers can use to more accurately and effectively evaluate the issues surrounding software selection. This template will include recommendations on ways to make informed and comprehensive software decisions for Custom or Customizable COTS software for specific subsystems.

To begin the software selection and evaluation process, one must answer the following questions. What are the criteria that Projects must consider and understand before making the COTS vs. heritage vs. Custom decision? Which aspects of a Project must a manager examine to understand the advantages of one method over another? What are the characteristics of the software options that need to be analyzed to reveal the "best" fit into the project system? When and under what conditions is it appropriate to inherit existing software from older projects? In which situations is a custom approach more suited for the needs of the project? In the current paradigm of Software Evaluation and Selection, these and other questions are simply not explored to sufficient detail. However, with the template provided by this paper, projects can address these questions that should be considered when making the selection between COTS, Custom, and Heritage software. This list is based on several sources including the information gathered during the interview process, the successes of those processes and my personal experience in software selection.

Of course, this is not an exhaustive list of the factors to consider when making the COTS/Custom/heritage decision nor are all criteria in the template applicable to all missions. For example, there can be external political factors, which drive the decision to adopt a particular software paradigm. This list only attempts to establish the more important issues for a manager to consider when making the software decision. Ultimately, the decision template will serve as a tool for Projects who are faced with the decision to adapt and implement existing COTS or heritage software or build a custom software system. It will be discussed in more detail in section

Literature Search

The first phase of the literature search was conducted using the Pepperdine Electronic Library resource, the IEEE website, the Software Quality Insurance group at JPL, as well as the Google and Yahoo search engines. The purpose of this search was to collect information and lessons learned from organizations in Industry and understand what methodologies and philosophies are the underpinning of this process.

This search clearly showed that there was no one methodology or accepted theory for selecting COTS over Custom software or vise versa. The bulk of the literature search turned up only discussions of the COTS vs. Custom decision and the possible considerations for each choice, typically with COTS or Custom biases. Articles did mentioned potential pitfalls with both approaches and benefits of each that seemed to correspond more with the author's bias. Overall, the results were very inconsistent. Several articles [13, 18] mentioned COTS as the better choice and other articles [17, 12] cited that Custom options could help a project avoid many common pitfalls associated with COTS. Additionally, there were no comprehensive templates or tools to speak of concerning the decision making process of choosing between custom and COTS software or for comparing COTS to other COTS options. In general, I found only high-level discussions on the pros and cons of each of the approaches.

In theory, COTS vs. Custom is an issue of "Faster and Cheaper" vs. "Better" and Software Evaluation seeks to identify which component will provide the best results, in the least time, with the lowest chance of failure. The articles reviewed here though, call into question whether or not COTS is really Cheaper or Faster. If COTS software is "compatible" with a Projects needs and infrastructure, then savings might be achieved in budget and schedule. The question still remains however, what qualifies as "compatible" so that a Project will be able to realize gains for the sacrifices in performance? And what are the criteria that a project will have to consider in order to determine the "compatibility" of a piece of software with the existing design. The next step of the literature search examines the software selection methodology and classification of existing software applications.

Scores of articles (Clarke [4], Hariri [8], Kitchenham [11], Morisio [14], and Paulet [15], and Torchiano [19]) discussed proper methods for software classification and focused on the technical characteristics of software applications.

For example, Torchiano [19] lists the following as key components in the software decision making process: Product Maturity, Market Share, Performance, Safety/Security, Reliability, Hardware Requirements, Product Support, Documentation, Usability, Learnability, Modifiability, Change Frequency, License Type, Cost of Use, Software Requirements, Conformance, and Domain Specificity. A poll of industry experts points to Cost of Use and Usability as the two most important from this list. [19].

Bertoa [2] lists Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability. Each of these is broken down further into specific attributes that one can evaluate quantitatively. In his article, Bertoa does mention that issues such as technical support and license conditions, while not directly related to quality, are important considerations. Bertoa, as with most research focuses on software quality exclusively.

Schneidewind [16] groups all attributes into 3 main categories: Reliability, Maintainability, and Availability. He concludes "The decision to employ COTS on mission critical systems should not be based on development cost alone. Rather, costs should be evaluated on a total life cycle basis and RMA should be evaluated in a system context." [16]. I agree completely.

Hariri [8] lays out what he believes to be the seven key characteristics for understanding the "quality" of the software tool: Ease of Programming, Debugging Support, Customization, Error Handling, Run-Time Interface, Integration with other software subsystems, and Portability. He stresses that Operational Usability should be one of the key factors in evaluating options, reasoning that if it is difficult to interface with the tool, its strengths will not be realized.

The bulk of the literature discussed characterizing the software and the process for testing the application within the system; outlining metrics for evaluation. Nevertheless, this is believed to be a major deficiency of the current mentality as it fails to take into account factors that will likely drive comprehensive decisions on selecting the correct software tool. Factors such as budget and schedule constraints, strength of the software team on the project, the state of project requirements, and political climate surrounding the organization were not discussed. A few articles did mention but do not explore the costs associated with software maintenance or the evaluation of the history and robustness of COTS tools. It is this papers contention that these aspects of the decision must be considered and are as important if not *more so* that the technical aspects of the software tools in question.

Interviews

The interviews were conducted to understand the tacit approach decision makers took during the software selection process. The results were both revealing and confounding. At the same time, it became clearer that the process for software selection was not clear and that software selection decisions were largely ad hoc. The interviewees were marred with issues during the project life cycle that prevented the development of a software selection process and projects continue to follow the labs inertia on software selection. Decisions were largely delegated to the Mission Operations Manager or the Mission Operations System Engineer who consulted with the project personnel to gather the required feedback. Several people confessed that there were insurmountable constraints and requirements on the software selection process from budget [32, 37] to time [38] to capability [39]. In other words, one criterion was given so much weight that the software selectors had no choice. The decision was made for them.

So what were the trends and observations that I have made from those at JPL who went through the process of software selection? There were five:

1. There is an “ad hoc” process for software selection
2. Little documentation of the selection process was created
3. Only subset of proposed criteria in Appendix A were examined
4. Inadequate examination of all available COTS/heritage options
5. No follow-up to evaluation the process for software selection

As hypothesized, there were no comprehensive or detailed processes for the selection of software tools at JPL. Managers largely made decisions based on a restricted number of criteria and personal preference. However, it did appear that in several cases [30, 37, 38] that there was little to no choice on the criteria to examine. In Interviewee Number Eight's case, cost mandated not only that an existing system be low cost, but also that the system must be integrated at the lowest cost. In Number Nine's case, the software requirements for flexibility due to the evolving project requirements mandated that the project implement a custom system.

The second trend witnessed, supports the observation that there appears to be no such process. I could find no reference to the process for software selection in any of the documentation databases: neither the process for software selection, the results of the software selection process, nor follow-ups to determine if the "process" was successful. The only inference to the software selection process was in Preliminary Design Review presentations, which mandated that the choice of software be mentioned. This is a clear indication that the priority placed on software selection is not high and the decision is largely trusted to System Engineers. Documentation of this process, however, should not be viewed as only for the people making the decision. Rather, the document is a method of "Knowledge Transfer" within the organization. It is a mechanism by which others can learn from the efforts of people on the lab.

Based on the interviewees and the existing "process", only a subset of the suggested criteria was examined. Typically, the tool that could meet the needs of the project at the lowest cost was selected. [30, 31, 32, 34] However, this method fails to take into account those scores of issues that could ultimately lead to greater costs beyond those initially anticipated. Upon analysis of the data, there were several key factors that were over looked by the interviewees, such as software team experience, subsystem complexity, potential change for the project systems, historical perspective of COTS options, and type and quantity of experience of the software development company or team. Where possible, following up with the interviewees, I asked them about these criteria and some freely admitted that with hindsight, more could have been done to investigate the ideal software selection. This narrow scope approach of evaluation lead to at least one failure in the Mars Climate Orbiter (MCO) when the past experiences of the contractor (Lockheed Martin) was not taken into account thus resulting in a miscommunication over the software units. MCO entered the Martian atmosphere below the acceptable threshold during Orbit Insertion and the mission was lost.

Fourth, almost all of the interviewees admitted that all of the COTS options had not been pursued aggressively. In fact most of the interviewees did not look beyond the software available from JPL directly. [30, 33, 37, 38] Without investigation, well-suited and inexpensive software could have been missed which had the potential to increase robustness and decrease the costs from the missions reviewed.

And finally, there was no follow-up after the missions to evaluate the "process" for software selection. Essentially, once a decision was made, the alternate options were "taken off the table" and the process was never again discussed. None of the projects had a process to examine the software evaluation steps that were taken and capture a "lessons learned" for future missions. This sort of knowledge transfer is critical for the improvement of this process and was clearly not a priority for those missions who were finishing. As part of JPL's strategy for knowledge transfer, projects should be required to convert the final mission data into knowledge about what was learned and how the processes used during the mission could be improved. Without this step, only those on the project will come away with that experience and the true benefit of knowledge transfer is not realized.

It was clear from the interviews that the emphasis was limited exclusively to the projects with no visibility into the Lab wide strategy of process improvement on a long-term broad-based strategic sense. Had there been, decisions would take into account a wider array of issues surrounding software selection such as establishing strategic relationships with software providers and more importantly moving towards standardization. But at the core of the problem is the belief that all projects are *unique* and don't fit within the JPL strategy, what ever it might be.

Section D contains the final results of this paper. It is a collection of criteria and issues examined by decision makers inside JPL as well as suggestions from several industry papers. The issues can be broken down into 6 key categories:

1. State of the Project – These questions surround the needs and the state of the project that is making the software decision. Understanding the state of the project is one of the most often overlooked issues surrounding the software decision. Far too often, only technical considerations are made without insight into the impact on the entire project.
2. System Impacts – Issues such as cost, risk, and schedule impacts are critical to the compete evaluation of software options. People tend to examine these issues at the exclusion of other important issues.
3. Subsystem Interface – Specifics such as the interface requirements and software flexibility; issues that always need to be addressed.
4. Team Strength– These questions revolve around the strength of the members on the team that will be involved in the software adaptation or custom software development. It also includes the strength and experience of the COTS vendor if applicable. This sensitive area must be considered however, if a project hopes to be successful in software selection and integration or development.
5. Support – This is perhaps one of the most critical areas and is ironically often overlooked by many decision makers until the project is in a crisis situation. Essentially, these criteria bring into question the vendor and the support from the vendor both through documentation and physical support for the software being adapted. Issues such as software updates, technical support, vendor availability, and license issues can lead to severe cost overruns if a project is forced to react to problems that could have been avoided.
6. Lab Strategy – Finally, an issue that is infrequently looked at, whether the software selection strategy for the project is in alignment with the Laboratory's strategy. In

other words, there are issues beyond the project, which must be considered as well. For example, perhaps working with a certain vendor will allow greater leverage for future negotiations with other software vendors. These issues can be addressed during the software selection process with help from laboratory management who have insight into the key strategic goals for the lab.

Projects asked to use this template, will be able to raise issues that might not otherwise be considered. Simply by following using this tool and asking the tough and sometimes obscure questions, decision makers will be falling back on the scores of experiences of other decision makers that have been faced with the same issues. It is these issues that will point in the direction of the software tool that will give the fewest or least sever problems. And in so doing, NASA/JPL will be taking its first steps towards a more effective and efficient software selection process and will become better suited for exploring Earth, our Universe and beyond.

Results - Software Selection Criteria Table

Attribute	Question
Project	What existing software options (both COTS and heritage) are available?
Project	What are the engineering and scientific requirements of the Project?
Project	How likely are these requirements to change?
Project	Which of the requirements is the highest priority? Which of the software options can best meet that need?
Project	What is the current state of the project with regards to Schedule?
Project	What is the current state of the project with regards to Cost?
Project	What is the current state of the project with regards to Risk?
Project	For mission success, how critical is the subsystem into which the software will be implemented? (i.e. Is robustness a high priority software feature?) Is this emphasis reflected in the candidate software?
Impacts	What are the operational impacts of the software option? Must additional people be hired to manage the software or is additional operations activity required for the software to function?
Impacts	What are the costs associated with researching the available COTS options? Request for Information (RFIs)?
Impacts	What are the planned schedule savings with each COTS option?
Impacts	What are the planned cost savings with each COTS option?
Impacts	What are the planned risk savings with each COTS option?
Impacts	What are the planned schedule savings with each Heritage software option?
Impacts	What are the impacts to Science return through reduced functionality by utilizing the COTS/Heritage software? Can this impact be quantified?
Impacts	What are the operational complexities that will be introduced by this software? In other words, what will the operational impacts be on the users who use the software in question?
Impacts	Likewise, are there any operational complexities that will impact other subsystems, which share an interface with this software?
Impacts	What are the planned cost savings with each Heritage software option?
Impacts	For the subsystem receiving the software, what would the impact of a software failure in this area be to the system?
Impacts	Are software failures in this subsystem time critical? (i.e. Post-mission science

	analysis vs. Command and Control software)
Impacts	What is the relative robustness of the COTS/Heritage software option available?
Integration	Are there compatibility issues with the COTS/Heritage Software and the subsystem or system design?
Integration	What other projects has the software been used on?
Integration	How similar were the requirements of those projects to this project?
Integration	How flexible is each software option in its function? Can it be modified during operations to satisfy another function? Do you have access to the source code?
Integration	How similar were those COTS/Heritage applications to the present software options being considered?
Integration	Can the COTS/Heritage Software be integrated into the existing infrastructure during the design phase of the project life cycle?
Integration	How complex are the requirements for the specific location receiving the software?
Integration	How likely are the requirements on the subsystem to change?
Integration	How many interfaces are there with the software?
Integration	How complex are the interfaces to/from the subsystem that requires software?
Integration	How flexible are the interfaces to/from the subsystem that requires software?
Integration	What is the scope of custom integration needed to meet the System Interface Specifications (SIS) and Operational Interface Agreements (OIA)?
Team	If custom option is chosen, how much experience does the team have in developing Custom Software?
Team	How much experience does the team have with adapting and integrating other COTS/Heritage software packages?
Team	How familiar is the software team with the available COTS/Heritage software?
Team	What similar experiences have the software teams managers had on other projects? How similar were those experiences to the project under consideration?
Team	Is software development a core competence for your organization?
Support	What is the track record of the company or organization that designing the COTS/Heritage Software?
Support	What information is known about the people involved in the development? Is this information relevant?
Support	Is the support provided with the COTS/Heritage software acceptable? For example, will the company be able to provide customer support if needed? At what cost and frequency?
Support	What is the frequency that the company will be providing upgrades? At what cost? What is the impact of this upgrade to the license?
Support	What kind of documentation is provided with the COTS/Heritage option?
Support	What is the response time for modifications to the software? Does this meet the needs of the project?
Lab Strategy	By outsourcing software development, does the organization loose this core competence?
Lab Strategy	What are the implications for the organization if the Project fails to meet its science requirements, on time and within budget?
Lab Strategy	Does the contract with the software company serve the future needs of the organization? In other words, are there strategic advantages to establishing relationships with outside companies?

References

[1] Bennatan, E. M.: "On Time, Within Budget: Software Project Management Practices and Techniques", McGraw-Hill International (UK) Limited, Library of Congress Catalog Number 92-14357, 1992
[2] Bertoa, Manuel F., "Quality Attributes for COTS Components", Department de Lenguajes y Ciencias de la Computacion, 2002
[3] Brown, Alan W., et. al.: "A framework for Systematic Evaluation of Software Technologies", Software Engineering Institute, Carnegie Melon University, IEEE Software, September 1996
[4] Clarke, S.J.: "Selecting the correct tools and methods for software systems development"; The Institution of Electrical Engineers; London, UK: 1995.
[5] Collier, K., et. al.: "A methodology for Evaluating and Selecting Data Mining Software", 32nd Hawaii International Conference on System Sciences, 1999.
[6] Dean, John C., et. al: "COTS Software Evaluation Techniques", National Research Council Canada, Software Engineering Group, NRC Report #43625, 1998
[7] Hansen, W. J.: "A Generic Process and Terminology for Evaluating COTS Software", Software Engineering Institute, Carnegie Melon University, 1999
[8] Hariri, S., et. al.: "Software Tool Evaluation Methodology", Northeast Parallel Architecture Center (NPAC), IEEE, 1995.
[9] HRS/IS Office, "Build vs. Buy Considerations", University of Texas, September 2001 (http://www.utexas.edu/hr/is/pubs/buy_v_build.html)
[10] Kandt, R. K., et. al: "A Survey of Software Tools and Practices in Use at the Jet Propulsion Laboratory", SQI Report R-1, JPL Document D-24868, Oct. 2002.
[11] Kitchenham, B., et. al.: "A methodology for evaluating software engineering methods and tools", Computing and Controlling Engineering Journal, June 1997.
[12] Kohl, R. J., "COTS based systems: Benefits, Potential Risks, and Mitigation Techniques", Titan Systems Inc.
[13] Morgan, R., "Designing Deployed Systems with COTS", Acme Embedded Solutions, Rick Pacelle, Sky Computers Inc.
[14] Morisio, M. et. al.: "IusWare: a methodology for the evaluation and selection of software products", IEE Proc.-Software Engineering, Vol. 144, No. 3, June 1997.
[15] Paulet, M. C.: "CAD Tool Evaluation: Methodology, Criteria, Benchmarks, and Results", Institute National Polytechnique de Grenobla, France, 1990.
[16] Schneidewind, N.: "Methods from Assessing COTS Reliability, Maintainability, and Availability", U.S. Government, Naval Postgraduate School, 1999.
[17] Sciortino, Michael Anton, "COTS, Open Source, and Custom Software: Which Path to Follow", October 2001, University of Buffalo
[18] Seacord, Robert C., Kurt Wallnau, Scott Hissam. "Custom vs. Vendor Integrated COTS", Institute for Information Technology, National Research Council of Canada.
[19] Torchiano, M., et. al.: "COTS Products Characterization", Department of Computer and Information Science (IDI), SEKE '02 July 15-19, 2002
[20] Vigder, M. R., http://wwwsel.iit.nrc.ca/projects/cots/COTSpg.html
[21] JPL Software Development Process Description, D-15378
[22] Engineer Mission Operations Systems, Rev. 1, D-57172

Interviews Referenced

[30] Number One – Software Implementation Manager
[31] Number Two - Ground System Manager
[32] Number Three - President of Software Development Company
[33] Number Four – Multi-mission Software Developer and Researcher
[34] Number Five – Multi-mission Manager
[35] Number Six – Multi-mission Software Development Manager
[36] Number Seven – Multi-mission Software Engineer
[37] Number Eight – Project Scientist
[38] Number Nine - Mission Operations Manager
[39] Number Ten – Division Software Development Manager