

COTS Software Acquisition Meta-Model

Marc Mosko, Hong Jiang, Arindam Samanta, Linda Werner

Jack Baskin School of Engineering

University of California at Santa Cruz

Baskin Engineering Building

Santa Cruz, CA 95064

(831) 459-4822

{mmosko, hjiang, arindam, linda}@cse.ucsc.edu

ABSTRACT

We present a software acquisition meta-model, called SAMM, for *Commercial-Off-The-Shelf* (COTS) software acquisition. Our model is a meta-model in that it includes sections that use other, detailed, models for specific tasks. SAMM is a complete life-cycle model that begins with an end-user need and ends with software maintenance. We have adapted several other models in to SAMM and added several novel features appropriate for the acquisition of commercial software.

Keywords

Acquisition, evaluation, requirements definition, life-cycle

1 INTRODUCTION

The SAMM model addresses the need for a practical software acquisition process that incorporates best practices of the software industry. We have developed an iterative model that focuses on user requirements and choosing software to meet those requirements. The present work derived from a joint project of UC Santa Cruz and Seagate Corporation to develop a COTS acquisition process [19].

There is a clear need for an acquisition model. [4, 18] note many risks associated with software acquisition. An organizational model can help a company successfully purchase and implement software. Such a model may help management conduct acquisitions and formalize user involvement. [27] found a positive correlation between management involvement in the acquisition process and the perceived quality of the software. This emphasizes the need for management to have a cohesive plan for software acquisitions that involves user groups. [8] found a large correlation between user involvement and system success. They also found a mod-

erate correlation between user participation and system success. [12] found that management involvement is important for successful information technology (IT) projects. Managers in successful companies took an active role in fostering innovation through technology. [12] found three other components of a successful IT project: speed of diffusion of technology, managing quality, and sustainability. In terms of quality, successful organizations paid particular attention to user support and had a feedback mechanism such that users could review the project. Finally, the study found that successful organizations funded their IT group such that they could sustain innovation and rapid diffusion.

From these case studies, we would deduce that a successful IT project should have the following characteristics. Management must take an active, supportive role. They should demonstrate how new technology and innovation may benefit end-users. They should also support the IT group. Through user contact and efficient project execution, the IT group may bolster end-user enthusiasm for IT projects. Management should provide the IT group the resources to engage users on a personal level. Finally, the project methodology should foster user buy-in and cultivate a positive user attitude towards the project. This should include user feedback channels.

Section 2 introduces our COTS acquisition model. Section 3 describes documentation and traceability standards used in our model. We conclude the paper in Section 4.

2 MODEL

We divide the software acquisition process as two distinct phases. We call the first phase the “choice phase”. The second phase is the “implementation phase”. The purpose of the choice phase is to choose COTS software that most closely meets user requirements. This phase also makes estimates for a fully custom application if no such COTS software may be found. The implementation phase concerns process control. It oversees the implementation of the chosen package to make sure it is implemented as anticipated and performs as expected.

The choice phase is iterative. The acquirer performs as many rounds as the customer needs to pick a product (or to decide no such product exists). Each round would refine requirements and perform any needed analysis. These results are presented to the customer who makes the final decision to buy or iterate. Based on a previous iteration’s analysis and testing, the following requirements stage may modify or remove existing requirements, in a *capabilities-to-requirements* style [3].

The implementation phase is sequential, possibly with some parallel activities. The SAMM model does not specify a deployment methodology, but it does state certain feedbacks that should flow from the deployment teams back to the acquisition group. The customer made a decision to buy a product based on certain representations, either by the vendor or by the IT group, that the product could be implemented for a certain cost and within a certain schedule. In the implementation phase, we observe cost, schedule, and satisfaction factors ¹. If any of these exceed a customer threshold, the acquirer should review the purchase decision and possibly implement remediations.

Following Fig. 1, there are four stages to the choice phase. We include two reviews. The stages are preparation, requirements, analysis, and presentation. The two reviews are after requirements and analysis.

The implementation phase has five stages. It is sequential, barring any major problems with the chosen software. The stages are: purchase, modify/customize, pilot deployment, general deployment, and maintenance. There are also two formal reviews in this phase, one after modify/customize and one after pilot deployment. We would note that sometimes the modification/customization activities may be pipelined with pilot group deployments and general deployments. By describing this phase as sequential, we do not mean to inhibit efficient deployment schemes.

The implementation phase also has two “remediation” terminations in cases where significant problems arise with an acquired software package. The specific activities of these steps would come from a risk management plan. Such a plan should be considered and defined by a risk management group.

Preparation The preparation stage defines the general parameters of the software acquisition process. This stage begins upon an original user request. This stage states the main constraints of time and budget. The project may also be defined as “simple” indicating that one may streamline some stages of the model. The

¹[22] notes several other indicators for quality besides our common understanding of “satisfaction.”

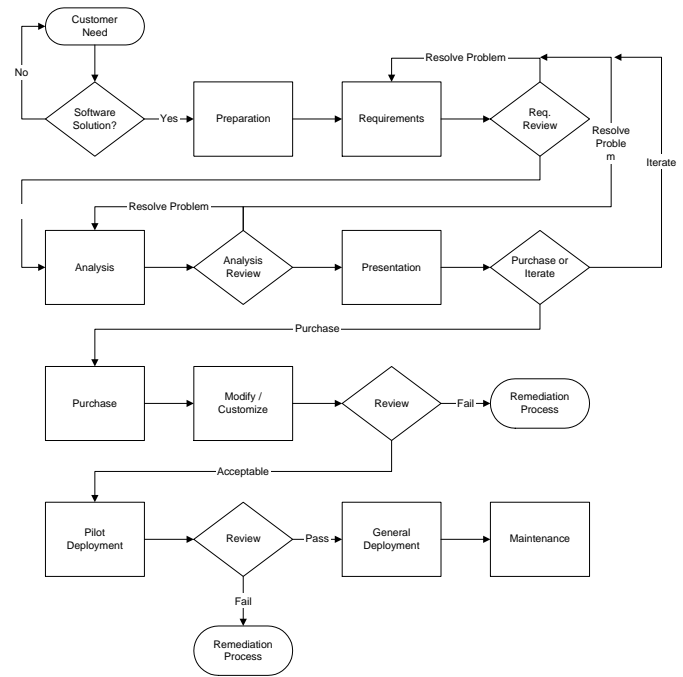


Figure 1: SAMM Process Flow

exact simplifications would be at the discretion of the project manager. This stage defines the review boards and stake-holders in the project. As the project proceeds, new stakeholders may also be identified.

Requirements The requirements section takes the original problem description and produces a testable, measurable plan that one may use in analysis. The main activity in this stage is to determine the software requirements. This section will produce the following documents: weighted user requirements, use cases, test plan, requirements quality measure, and weighted technical requirements.

Our preferred elicitation method is use cases. We believe use cases are appropriate for our model because they (a) are user-centered, (b) proceed from general requirements to specific, (c) the system is a “black box”, and (d) use cases are well received by methodologists [17, 23].

One benefit of use cases is that they may lead to *in situ* test plan cases. Such test cases may yield higher quality evaluations. In the reciprocal process of stating requirements and evaluating capabilities, use case testing might lead to specific process improvements. As users and the IT group test products, their understanding of software capabilities may change. These new views, expanded or contracted, may modify how users think about their

business process. They may also affect the stated requirements or use cases.

User requirements should be weighted. If requirements are based on use cases, then use cases should be weighted. [25] has a weighting system specific to use cases to estimate work in a project plan. Several ideas from [25] might be useful for requirement weights. One may weight actors such that activities which involve specific actor(s) have proportional weights. One may also weight use cases based on the number of business transactions they contain.

The goal of the test plan is to provide the analysis stage with an implementable test suite. It should be a maximal coverage of user requirements within time and budget constraints. The test plan should specify: (a) what is being tested (requirement), (b) who will do the testing, (c) how testers will perform the test (e.g. by cases), and (d) criteria for grading

We wish to call attention to the field of product testing [26]. COTS evaluation is essentially a black-box environment [14]. The COTS acquisition process is closer to buying consumer goods than to specifying and developing a custom software package. Based on [26, 22], we would recommend the following in devising a test plan. First, designate “must have” requirements, which lead to “must pass” tests. If a product fails these tests, there is no need to test further. Second, scale each “must have” requirement’s numeric weight to the sum of all other non-must have requirements. Third, revisit early tests and check for skew in how graders evaluated products. Lastly, compare features between multiple products in head-to-head tests (see [7] for an opposing view).

Reviews Formal reviews are evaluations of project status to ascertain discrepancies from planned performance and to recommend improvement. Such activities help find and solve problems as early as possible. In our model, we use formal reviews after requirements, analysis, customization, and pilot stages.

A review process can be divided into management and technical reviews [9]. The objective of a management review is to ensure that activities progress according to plan, to identify the need for alternative planning or to change project direction when necessary. Technical reviews verify the correctness and standardization of activities. If errors are found in the output or the examined process, the process might need to be repeated.

This stage produces a review report identifying: the project being reviewed, the review team, inputs to the review, review objectives, action item ownership and status, a list of issues and recommendations identified

by the review team that must be addressed for the project to meet its milestone, recommendations regarding any further reviews and audits, and a list of additional information and data that must be obtained before they can be executed [9].

Analysis The Analysis stage achieves three goals: estimating an equivalent custom development effort, acquiring vendor proposals, and testing vendor products for conformance with user requirements. This section depends on the weighted user requirements, use cases, test plan, and weighted technical requirements documents produced in the requirements stage. It produces a qualified vendor list, RFP, proposals and test results obtained from the test plan. The custom development estimate may use a method such as Function Point Analysis [29], which does not require a detailed knowledge of the software project. [20] also has some useful high-level methods when considering how to acquire an application.

Vendors play an important role in the acquisition process. [10, 11] also define the acquirer/vendor relationship and prescribe actions. The acquirer should prepare a detailed Request for Proposal (RFP). There may be significant informal exchanges between the qualified vendors and the acquirer before a formal vendor response [6]. Qualified vendor responses to the RFP should provide: executive summary, detailed costs, detailed implementation schedule, training plan, and detailed responses to RFP questions.

The analysis stage also evaluates qualified vendor products against user requirements. Our complete description of SAMM [19] proposes a method to weight requirements and assign letter grades to tests. The letter grades correspond to real values between 0 and 4. Through a method of aggregation, we arrive at weighted overall grades for each product. The exact process is not important to the SAMM model as long as there is a method to evaluate products in regards to requirements [13, 5, 32]. This overall grade is then used by the Presentation stage, in which the customer makes a “buy” or “iterate” decision.

Presentation The main purpose at this stage is to help the customers understand the results of analysis, and to solicit their opinions when choosing the most suitable product. The IT group needs to generate an overall evaluation on each candidate product based on the analysis results. They should present to the customer an evaluation of data as cost, schedule estimation, and risk assessment of vendors. Multiple metric graphs [21] may provide a good presentation tool.

After presenting enough information and collecting cus-

customer's feedback, the buy or iterate decision can be made. If the customer prefers one of the candidate products, the purchase can start. If the customer is not satisfied with any of the products, then the acquisition process must go back to refine either the requirements or the analysis. If the customer can not make the final choice, one might include non-functional requirements [2, 24, 30].

The presentation stage produces the decision either to purchase or to iterate and documents explaining the decision.

Purchase At this stage, the acquirer and vendor will conduct negotiations followed by signing a purchase contract. Negotiations should be based upon the existence of: adequate written specifications, clear definition of the obligations and responsibilities, time frames for work to be accomplished, and a balance of the responsibilities, risks, and benefits to both parties. One approach for negotiation is to break the negotiations into "technical" and "commercial" parts [16]. One should consider future upgrades, technical support, and contingencies such as product substitution or code escrow [4].

This stage produces: a legal contract, the COTS software, and supporting documents to the software.

Modification/Customization Since COTS software is designed originally for an overall marketplace, its functionality and performance may not completely fit the requirements of a specific user group [15]. The acquirer may customize the software through APIs or scripting languages. Some modifications require the vendor to change source code. Tests for the final software product should be given based on the test plan produced at the requirements stage.

For the acquirer, the focus of this stage is control of schedule and software quality. The IT group should work with the vendor's development group, plan and monitor the customization or modification process, and then test the modified product. The process should adopt dynamic, risk-driven process models, assessing risks via prototyping, benchmarking, reference checking, and related techniques, and using top people to resolve top risk items in advance [4].

For vendor modifications, the acquirer's IT group needs to monitor the vendor's progress to ensure that all milestones are met and approve work segments [9]. For in-house customization, the IT group needs to develop an implementation plan and a test plan, using any accepted development methodology. This stage produces: a user manual reflecting the current characteristics of the soft-

ware product and a technical document reflecting the current technical detail, which is useful for future maintenance.

Pilot Deployment Pilot deployment stage of the software acquisition process verifies whether the software operates as needed in a real environment. The acquired software is deployed in a subset of the systems and tested by a subset of the users. This stage has three sub-stages: definition, evaluation, and analysis. The definition sub-stage specifies the pilot group size, pilot group members, work load, and test suites. In the evaluation sub-stage, the software is deployed to the pilot group and evaluated. A risk mitigating model [31] with faster review process and low maintenance cost is used in this sub-stage. In the analysis sub-stage, the IT group analyzes the results obtained from the pilot group. All the results are summarized according to the requirements met or missed. Thereafter the software receives a final evaluation.

General Deployment The general deployment stage performs the delivery of the acquired software to users' systems. A deployment team installs the acquired software. The acquisition group tracks their progress through status reports to ensure the process meets customer expectations. They review the reports for compliance with cost, schedule, and user satisfaction goals.

Maintenance In this stage, the IT group supports the deployed software. Maintenance has three sub-stages: routine maintenance, troubleshooting, and end-user review. Routine maintenance includes periodic upgrades or vendor patches. Troubleshooting solves user problems with a feedback channel to the vendor. End-user review periodically evaluates customer satisfaction with the product. It also looks at any changes in end-user requirements, the maintenance history of the software, and longterm costs. This may result in a new iteration of the acquisition model to find replacement software.

3 DOCUMENTATION AND TRACEABILITY

It is important that the IT group maintain a written history of software products. Over time, the main benefit will be for impact analysis. When the acquirer wishes to change systems at a later date, it becomes important to know how that will affect installed software. The topics in this section are based on [1]. The main objective is to create a paper trail (or e-trail) such that one has forward and backward references.

Documentation should be maintained in a revision con-

trol system. This may be an automated system or a manual process. We would recommend the following minimum practices: (1) control the releases of a document, and number each release; (2) maintain a history of released documents; (3) appoint a person responsible for each document; (4) maintain a version change history that indicates what changed from version to version; (5) use change request forms.

Traceability means that at each stage of a project, one may determine from where one came and to where one will proceed [28]. When requirements change, one may find the affected pieces. If the customer drops a requirement, for instance, one should no longer test for that requirement and make a “buy” decision based on such tests. We make specific traceability recommendations in [19].

4 CONCLUSION

In this paper we presented a complete acquisition meta-model. It is user-oriented and iterative. We adopt use cases for requirements elicitation, test case design, and product evaluation. Our model unifies all aspects of COTS acquisition with specific feedback to the acquisition process. The SAMM model fills a business need to codify the COTS process. Through proper user involvement, we should achieve higher user acceptance of both the process and end result.

REFERENCES

- [1] ANSI/IEEE Std 1042. *IEEE Guide to Software Configuration Management*. IEEE, New York, 1987.
- [2] D. Avison, W. Eardley, and P. Powell. Suggestions for Capturing Corporate Vision in Strategic Information Systems. *Omega*, 26(4):443–459, Jul 1998.
- [3] B. Boehm. *Tutorial on Software Risk Management*. IEEE Computer Society Press, Los Alamitos, CA, 1989.
- [4] B. Boehm and C. Abts. COTS Integration: Plug and Pray. *Computer*, pages 135–138, January 1999.
- [5] L. Briand. COTS Evaluation and Selection. In *Proc. International Conference on Software Maintenance*, pages 222–223, Nov 1998.
- [6] J. Chudge and D. Fulton. Trust and Co-operation in System Development: Applying Responsibility Modelling to the Problem of Changing Requirements. *Software Engineering Journal*, 11(3):193–204, May 1996.
- [7] D. Gensch and S. Ghose. Differences in Independence of Irrelevant Alternatives at Individual vs Aggregate Levels, and at Single Pair vs. Full Choice Set. *Omega*, 25(2):201–214, Mar 1997.
- [8] M. Hwang and R. Thorn. The Effect of User Engagement on System Success: A Meta-Analytical Integration of Research Findings. *Information and Management*, 35(4):229–236, Apr 1999.
- [9] IEEE Std 1028. *IEEE Standard for Software Reviews and Audits*. IEEE, New York, 1989 corrected edition, 1989.
- [10] IEEE Std 1062. *IEEE Recommended Practices for Software Acquisition*. IEEE, New York, 1998.
- [11] IEEE/EIA Std J-STD-016-1995. *Software Development Acquirer-Supplier Agreement*. IEEE, New York, 1995.
- [12] R. Jain. Key Constructs in Successful IS Implementation: South-East Asian Experience. *Omega*, 25(3):267–284, Jun 1996.
- [13] J. Kontio. A Case Study in Applying A Systematic Method for COTS Selection. In *Proc. IEEE 18th International Conference on Software Engineering*, pages 201–209, March 1998.
- [14] B. Korel. Black-box Understanding of COTS Components. In *Proc. 7th Inter. Workshop on Program Comprehension*, pages 92–99. IEEE Computer Society, May 1999.
- [15] S. Lacy. Experience/Position Paper on COTS Issues. In *ICSE COTS Workshop*, Jan 1999.
- [16] J. P. Lewis. *Mastering Project Management*. McGraw-Hill, New York, 1998.
- [17] N. Maiden, L. James, and C. Ncube. Evaluating Large COTS Software Packages: Why Requirements and Use Cases are Important. In *ICSE COTS Workshop*, Jan 1999.
- [18] N. Maiden and C. Ncube. Acquiring COTS software selection requirements. In *Proc. 3rd Inter. Conf. Requirement Engineering*, page 241. IEEE Computer Society, Apr 1998.
- [19] M. Mosko, H. Jiang, A. Samanta, and L. Werner. Software Acquisition Meta-Model. UCSC-CRL-00-02, <ftp://ftp.cse.ucsc.edu/pub/tr/ucsc-crl-00-02.ps.Z>, Dec 1999.
- [20] P. Nelson, W. Richmond, and A. Seidmann. Two Dimensions of Software Acquisition. *Communications of the ACM*, 39(7):29 – 35, 1996.
- [21] S. Pfleeger, J. Fitzgerald Jr., and D. Rippy. Using Multiple Metrics for Analysis of Improvement. *Software Quality Journal*, 1:27–36, 1992.

- [22] S. Rivard, P. Lebrun, and J. Talbot. Measuring the Quality of User-Developed Applications. *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, 4:471–478, Jan 1991.
- [23] J. Rumbaugh. Getting Started Using Use Cases to Capture Requirements. In R. H. Thayer and M. Dorfman, editors, *Software Requirements Engineering*, pages 123–127. IEEE Computer Society Press, second edition, 1997.
- [24] P. Sawyer, I. Sommerville, and S. Viller. Capturing the Benefits of Requirements Engineering. *IEEE Software*, 16(2):78–85, Mar-Apr 1999.
- [25] G. Schneider and J. P. Winters. *Applying Use Cases A Practical Guide*. Object Technology Series. Addison-Wesley, Reading, Massachusetts, 1998.
- [26] M. Scriven. Product Evaluation. In N. L. Smith, editor, *New Techniques for Evaluation*, volume 2, pages 121–166. Sage Publications, Beverly Hills, 1981.
- [27] H. Shin and J. Lee. A Process Model of Application Software Package Acquisition and Implementation. *Journal of Systems and Software*, 32(1):57–64, Jan 1996.
- [28] I. Sommerville and P. Sawyer. *Requirements Engineering – A Good Practice Guide*. John Wiley & Sons, New York, 1997.
- [29] C. Symons. Function Point Analysis: Difficulties and Improvements. *IEEE Transactions on Software Engineering*, 14(1), Jan 1988.
- [30] J. Teng, K. Fiedler, and V. Grover. An Exploratory Study of the Influence of the IS Function and Organizational Context on Business Process Reengineering Project Initiatives. *Omega*, 26(6):679–698, Nov 1998.
- [31] V. Tran and D.-B. Liu. A risk-mitigating model for the development of reliable and maintainable large-scale commercial-off-the-shelf integrated software systems. In *Proceedings of Annual Reliability and Maintainability Symposium*, pages 361–367. Annual Reliability and Maintainability Symposium, Philadelphia, PA, USA, 13-16 Jan. 1997, 1997.
- [32] Y. Sumi, K. Hori, and S. Ohsuga. Supporting the Acquisition and Modeling of Requirements in Software Design. In *Knowledge-Based Systems*, volume 11, pages 449–456. International Workshop on Strategic Knowledge and Concept Formation, Loughborough, UK, 17-19 Nov. 1997, 1998.