COTS-Based Software Development: Processes and Open Issues

M. Morisio[4], C.B. Seaman[2,3], V.R. Basili[1,3], A.T. Parra[5], S.E. Kraft[6], S.E. Condon[5]

[1] University of Maryland
College Park, MD 20742, USA, basili@ cs.umd.edu
[2] University of Maryland Baltimore County
Baltimore, MD 21250, USA, cseaman@umbc.edu
[3]Fraunhofer Center Maryland
College Park, MD 20742, USA, basili@ fc-md.umd.edu
[4]Politecnico di Torino
Torino, Italy, morisio@polito.it
[5]Computer Sciences Corporation
Lanham, MD, aparra@csc.com
[6]NASA/Goddard Space Flight Center
Greenbelt, MD, skraft@gsfc.nasa.gov

ABSTRACT

The work described in this paper is an investigation of COTS-based software development within a particular NASA environment, with an emphasis on the processes used. Fifteen projects using a COTS-based approach were studied and their actual process was documented. This process is evaluated to identify essential differences in comparison to traditional software development. The main differences, and the activities for which projects require more guidance, are requirements definition and COTS selection, high level design, integration and testing.

Starting from these empirical observations, a new process and set of guidelines for COTS-based development are developed and briefly presented.

Keywords
Commercial off-the-shelf, COTS, component-based, empirical study, software development process.

1 INTRODUCTION

The world of software development has evolved rapidly in the last decade. In particular, the use of commercial off-the-shelf (COTS) products as elements of larger systems is becoming increasingly commonplace, due to shrinking budgets, accelerating rates of COTS enhancement, and expanding system requirements, according to the Software Engineering Institute (SEI) [13]. The growing trend toward systems configured of individual components has taken the original concept of reuse into a completely different arena. It has also presented many challenges to software developers attempting to enter this new arena. The SEI describes some of these:

> *[The marketplace is] characterized by a vast array of products and product claims, extreme quality and capability differences between products, and many product incompatibilities, even when they purport to adhere to the same standards.* [14]

As this change in practices is taking place, many questions have arisen, such as: how do we modify standard development practices to fit this new development paradigm; what methods are now effective; how do we quantify the cost savings expected of COTS use; and what metrics are worth collecting from COTS projects?

The term *commercial off-the-shelf (COTS)* is very generic; it can refer to many different types and levels of software, e.g., software that provides a specific functionality or a tool used to generate code. COTS may be one of the most diversely defined terms in current software development. For the purposes of this paper, the term *COTS* means a software product, supplied by a vendor, that has specific functionality as part of a system—a piece of prebuilt software that is *integrated* into the system and must be *delivered* with the system to provide operational functionality or to sustain maintenance efforts. For the purposes of this paper, *COTS-based* is the term used to indicate component- or package-based development—rapid configuration of software systems based on COTS packages, Government off-the-shelf (GOTS) packages, and some custom-built reusable packages. The term *COTS project* refers to a project that integrates COTS packages and other software to develop a system. This is not to be confused with the development of COTS packages that occurs at the "vendor" organization.

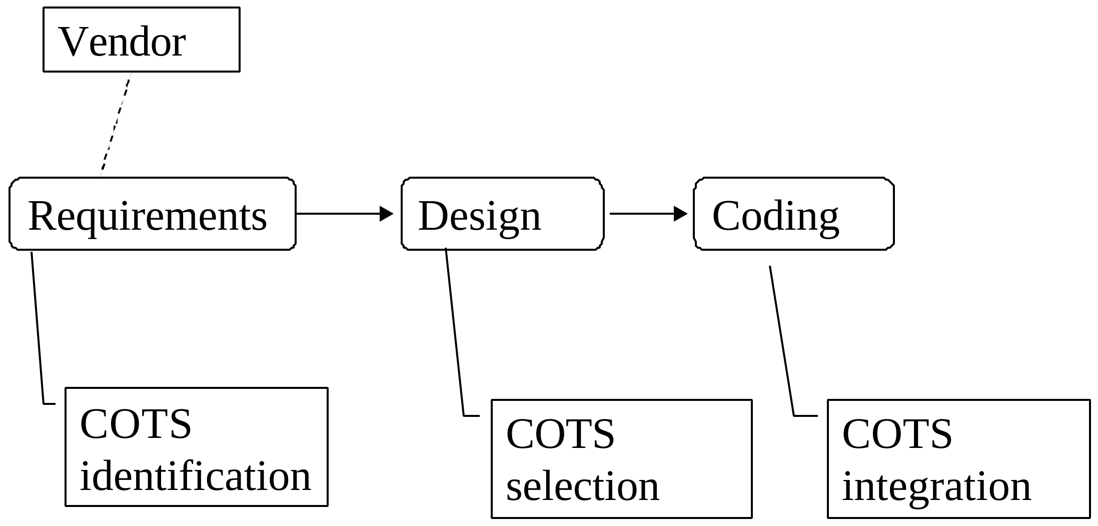
Section 2 describes the environment of the projects we analyzed and the motivations for this study, section 3 describes the methodological approach we used in the study, section 4 presents the results of interviews, specifically the COTS used and the COTS-based process in use, and section 5 presents a new suggested COTS process.

2 BACKGROUND

The Software Engineering Laboratory (SEL), a joint academic/industry/government institution known for empirical study of software development practice, provides local guidance to NASA/Goddard Space Flight Center (GSFC) projects, based on empirical findings, as well as serving as a model for process improvement for the global software development community.

The Flight Dynamics Division (FDD) at NASA/GSFC, which until recently has been responsible for the development of large amounts of ground support software for NASA satellites, has been studied by the SEL for more than 2 decades. In the 1980s and 90s, the FDD achieved effective software reuse levels as high as 90 percent. This trend has gradually evolved into COTS-based software development. Realizing that COTS-based development represents a significant shift in the way the FDD does business, the SEL decided to update its guidelines on recommended practice and processes [11]. Unlike previous SEL activities, this effort was in response to something that was already happening in the development community in the FDD. Projects began using COTS components in their systems, in part due to pressure from NASA management, without guidance from the SEL. Thus, the SEL was reacting to this new technology rather than proactively studying it and monitoring its introduction into the organization.

Figure 1. The proposed COTS process.



According to the Quality Improvement Paradigm [1], the study is an iteration of *understanding*, *experimenting* and *packaging* phases. During the last few months of 1995, the SEL designed an initial COTS-based process. Because the FDD had limited experience developing COTS-based systems at that time, the SEL looked at experiences of outside organizations in order to understand the challenges associated with this type of development and to gather best practices used on COTS projects. Using a solid understanding of the FDD project domain, history and environment, the SEL synthesized this information into a suggested process to be used to produce COTS-based systems in the FDD. This initial process was then reviewed for feasibility by key FDD software engineers who have had some experience with COTS. The resulting proposed process [12] is shown in Figure 1. The process is sequential, with the high level phases Requirements, Design, and Coding. The square-cornered boxes in the picture highlight COTS-related activities performed in a phase. COTS identification belongs to the requirements phase, along with interaction with the vendor. COTS selection belongs to the design phase, COTS integration and testing to the coding phase.

In parallel with the proposed process definition, several COTS projects started at FDD. The SEL tried to gain a greater understanding of these projects by looking at their effort data. It soon became clear, however, that this data was inadequate to tell the entire story of what was occurring on COTS projects. One shortcoming of the data collection at that time was that the traditional activities listed on the data collection form were too general to elicit useful data specific to COTS-based development activities. One indication of this was the disproportionately large amount of effort listed as "Other" on COTS projects. Further, these data were not sufficient to identify and understand the new issues arising from the use of COTS packages in FDD projects. As a consequence, it was decided to study these projects at a much deeper level, using new research tools to gather and interpret qualitative, and not only quantitative, data.

The reader will not be surprised to know that the actual COTS process used by the projects observed was radically different from the suggested process. This paper is dedicated to describing the actual COTS process used, issues raised by project members, and a new process defined to address and resolve these issues.

3 STUDY DESIGN

The goal of the observational study was to learn about how COTS-based development is carried out in the FDD, and to provide recommendations on how it could be conducted better.

The main investigation tool used in this study was the structured interview, because of the need to understand the existing de facto and possibly unstructured process, with its issues and problems.

From the high level goal of the study, a number of questions and measures were derived, using the GQM approach [2]. Questions were organized into structured interviews to be held with project managers and staff of COTS projects. The GQM decomposition and the interview guides can be found in [9,12]. In short, the interviews were aimed at characterizing the projects, the COTS used, and the process used. Overall, interviews were conducted with 25 representatives from 15 projects.

The interviews were conducted by teams composed of two study team members. One team member served as the interviewer and the second as the scribe. The interviewer's responsibilities included conducting the interview using the interview guide as an outline, posing open-ended questions to the interviewee, and following up as appropriate to gather as much information as was reasonable.

Table 1. A sample of COTS used by projects.

COTS name	Domain/ Functionalities
STK – Satellite Tool Kit	Orbit determination and mission planning
Labview	Data acquisition, analysis and visualization
Autocon	Orbit determination
Altair Mission Control System	Mission Control
Probe	Data analysis
GensAa (GOTS)	Spacecraft monitoring, commanding, fault detection and isolation
GTDS (GOTS)	Orbit determination
Builder Xcessory, X-Software, Shared-X, Visual Optimization package, X Runner	GUI, GUI builders
Matlab	Computing environment, data visualization, application development

The scribe's two main responsibilities were (1) to keep notes on the information covered in the interview to enable the interviewer to concentrate on facilitating the interview process and (2) to document the interview in a structured textual manner following the basic outline of the interview guide. The scribe usually spent some time after the interview to write detailed notes on all of the interviewee's responses, based on notes taken during the interview.

When collecting qualitative data, verification steps are crucial. Two quality checks were built into this procedure. First, at the end of the interview, the scribe asked the interviewee to clarify parts of the interview as needed and asked any questions from the interview guide that may have been inadvertently omitted by the interviewer. Second, prior to finalizing the interview notes, the interviewer reviewed them for concurrence.

The interview data were analyzed in several ways to extract a variety of different types of information. Different study team members reviewed the interview notes, each concentrating on gaining an understanding of a different aspect of COTS-based development. At the same time, the interview notes were also analyzed using a method loosely based on the constant comparison method [7, 5], a rigorous qualitative analysis method used to identify trends and consensus in textual data. This type of analysis led to findings about the process steps followed in COTS-based development and the main advantages and disadvantages of using COTS.

4 RESULTS OF INTERVIEWS

4.1 Projects and COTS

More than 30 COTS packages were used by one or more of the 15 projects contacted during this study. Several COTS packages, such as the Satellite ToolKit, were used by several projects. Some projects were using one or two COTS products, while others used more. One project used as many as 13 individual COTS. COTS were used to build ground support systems, as well as assist in a platform transition from mainframe to workstation that all flight dynamics systems underwent. COTS used ran the gamut from a COTS product that completes the entire function of a telemetry processor, to fourth-generation languages that must be delivered with the system to maintain it.

Table 2 lists a sample of COTS used by projects interviewed. The list is not exhaustive and is meant to give a better idea of the types of products used. COTS can be categorized as domain specific (orbit determination and mission control) or generic. Most of the COTS used could be classified as domain-specific. Among generic COTS, the subcategories are databases, data acquisition, analysis and visualization, GUI builders, networking and middleware.

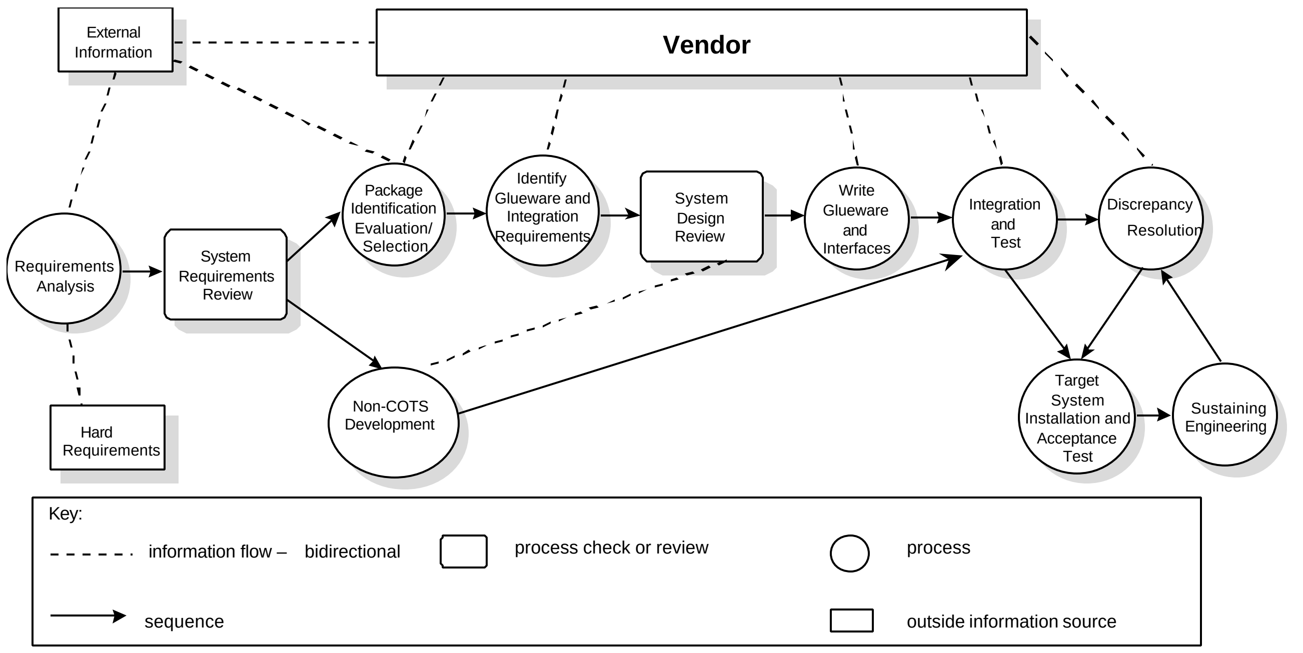
All of the projects were from the traditional domain of NASA Goddard's FDD, that is to say ground software dedicated to controlling satellite orbit, attitude and communication with earth. The fifteen projects ranged from 29 to 300 KSLOC, with the size figure including only code developed and excluding COTS code.

Carney [3] identifies three types of COTS-based systems based on the number of COTS used and their influence on the final system. **Turnkey** systems are built around a (suite of) commercial products, such as Microsoft Office or Netscape Navigator. Only one COTS is used, and customization does not change the nature of the initial COTS. **Intermediate** systems are built around one COTS (e.g. Oracle) but integrate other components, commercial or developed in-house. Finally **other** systems are built by integrating several COTS, all on the same level of importance. In our case projects fell into the second and third categories.

4.2 The actual COTS process

Although not every team followed all of the steps outlined below, a composite process flow emerged from the interview data.

Figure 2. The actual COTS process.


External Information
Vendor
Package Identification Evaluation/ Selection
Identify Glueware and Integration Requirements
System Design Review
Write Glueware and Interfaces
Integration and Test
Discrepancy Resolution
Requirements Analysis
System Requirements Review
Non-COTS Development
Target System Installation and Acceptance Test
Sustaining Engineering
Hard Requirements
Key:
information flow – bidirectional
process check or review
process
sequence
outside information source

Below we describe the steps and quality checkpoints in the overall process, as shown in Figure 2.

- **Requirements analysis**

The earliest steps in COTS-based development are similar to traditional development—requirements gathering and analysis. In the requirements phase a strong emphasis is on gathering external information. Much of this information comes from separate organizations. Some project requirements are predefined, with minimal requirements analysis needed.

Glueware and interfaces, as dictated by the system architecture, operating system and hardware, are identified. These can be seen as specific requirements imposed by the COTS used.

The requirements activity changes considerably as compared to traditional development. Projects reported that, with COTS, not all requirements are equal. Some are "free", or provided by the COTS. Some are not immediately available, but can be worked out with reasonable extensions and add-ons. Others are incompatible with the COTS. Satisfying these requirements can become time consuming. Sometimes projects complained about not being able to negotiate these requirements. The effort spent to satisfy these "hard" requirements can outweigh the effort saved with "free" or "easy" requirements.

- **System Requirements Review**

This review closes the requirements phase. This first verification step is aimed at checking the completeness and feasibility of system requirements. Early reviews of the requirements are crucial, even with a less formal process.

- **Package identification, evaluation/selection**.

COTS identification consists of Web searches, product literature surveys and reviews, identification of other reusable system components, and recommendations from external sources. Product information is kept in a central justification notebook, or an evaluation notebook. In most cases, this is a physical, paper-based, binder that contains spreadsheets comparing different alternative COTS products on various criteria. Different project members charged with evaluating COTS products from different perspectives enter the relevant information in these spreadsheets as they complete their evaluations. On many projects, one team member or a small task force of team members is charged with collecting information on COTS products and keeping the notebook. Not only are product evaluation notes kept, but subjective comments concerning vendor quality and responsiveness are also kept.

As COTS are identified, the evaluation and selection processes begin. COTS evaluation steps mentioned in the interviews included prototyping, vendor demonstrations, and in-depth review of literature such as manuals and user guides. Vendor training sites and availability are considered. Procurement issues surface such as development fees for added requirements, licensing and maintenance fees, and sustaining engineering support.

- **Identify glueware and integration requirements**

This step, although it is the activity that most closely resembles design in a traditional development process, is less a problem of decomposition (finding the right decomposition that best satisfies the requirements) and more a problem of composition and integration (finding the best way to integrate COTS that satisfies the requirements). That is, it is more of a bottom-up process, where developers attempt to put together COTS products, like building blocks, until all requirements are met. As a result, gaps are identified where the chosen configuration of COTS products leaves some requirements unfulfilled. At the same time, all interfaces are identified.

- **System Design Review**

This second verification step deals with System Design. Only some teams held it formally, but all teams mentioned some mechanism to apprise the customer of the design.

- **Non COTS development**.

Some projects reported that a significant part of the project effort was still spent in traditional development that did not depend on COTS. This development begins in parallel with the early COTS-related steps, as in a traditional development project. Non-COTS cost and schedule are monitored. There is a bi-directional information flow between the COTS-based process flow and the non-COTS development that comes into play in the design review.

After the design review, whether it is formal or informal, traditional non-COTS development continues in parallel with the coding of the glueware and the interfaces. This activity does not really require any new skills or tasks, but it is sometimes forgotten when effort and resources are allocated at the beginning of the project. In other words, it must be included in the project plan along with the COTS-related work.

- **Write glueware and interfaces**

Projects reported, in this phase, less coding and debugging than in the traditional process, and as a consequence less unit testing and inspections. Close contact with the vendor technical staff, or a competent Help Desk is essential during this development.

- **Integration and test**

Projects reported spending a lot of effort in integration, and encountering problems during it. Actually for many projects integration was the activity that consumed the most effort.

The integration step varies a great deal from project to project, depending on which and how many COTS products are being used. At system integration and testing the COTS packages are treated as black boxes. The teams commented that testing focused on the interface glueware and the format and correctness of the input files. Again, the importance of availability of the vendor technical staff or Help Desk was emphasized. Testing is usually conducted piece-by-piece, as each software component is integrated.

- **Target system installation and acceptance test**

Unlike the traditional life cycle, no formal acceptance testing or operational readiness reviews were mentioned by the teams. The development team installs the software on the target system. It seemed that the development teams felt that the systems being built relied enough on pre-tested components that the reliability of the system as a whole could be assured. Since none of these systems had yet reached the maintenance stage, it can’t be said whether or not this assumption held up.

- **Discrepancy resolution**

Once installed, navigational training to familiarize the customer with the system is conducted. During this phase, a member of the development team is the single point-of-contact or intermediary between the customer and the vendor. This person is responsible for reporting discrepancies, and handling software “patches” or corrections. Interviewees mentioned that software patches were placed on vendor Web sites that were downloaded to the target system.

- **Sustaining engineering**

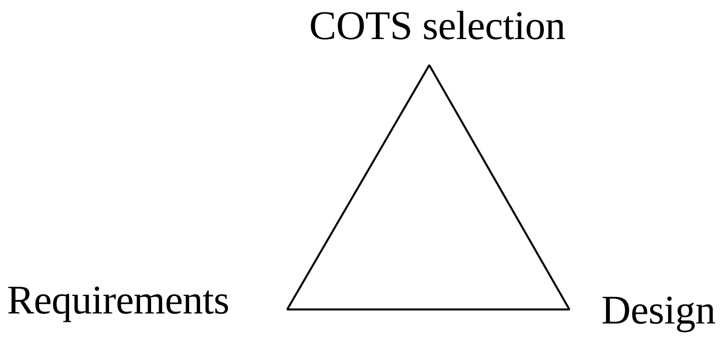
At the time of this study, none of the participating projects had reached this stage of the process, so we can only speculate about the nature of sustaining engineering (or maintenance).

4.3 Process Insights

The study team discovered more complexity in the current practice than expected in theory. This is clear when comparing Figure 1 (representing the process the study team expected to find) and Figure 2 (the process that emerged from the interview data).

For example, it had been expected that vendor interaction would be simple, and would end with the purchase of a product. In reality, the interaction continues throughout the life cycle and the flow of information is not merely one way. We found a strong dependence on *bi-directional* information flow. Projects reported that they needed to interact with the vendor during all phases. This activity consumes effort, and it is important to allocate time for it when planning. Further, there are two kinds of interaction: technical (to understand the functionality of the COTS, to understand how to use the packages, to solve integration issues, etc.) and non-technical (such as procurement, licensing, pricing, negotiations in general). Projects did not

Figure 3. Relationship among COTS selection, requirements and design



have the skills for non-technical interactions, and felt this task was a burden.

There is also constant involvement with separate organizations, such as other projects using COTS, independent evaluation teams, and other customers of the vendor. Another complication is that portions of the COTS-based systems include traditionally developed software.

Many projects underlined the strict relationship between requirements, design and COTS selection. Basically, requirements and design choices both depend on the COTS selected. Figure 3 summarizes this concept, showing that a decision on one vertex of the triangle implies consequences on the other two vertices. This was often the biggest challenge for projects. The wrong selection impacts all relevant project issues. And this choice has to be made early in the lifecycle, with imprecise information. Further, sometimes the COTS choice is dictated.

In summary, COTS projects were obliged to follow a process quite different from traditional projects, with more effort put into requirements, test and integration, and less in design and code. We can identify three types of differences.

- New activities: product evaluations, product familiarization, vendor interaction (both technical and non-technical). The related new roles are the COTS evaluation team, and a team member responsible for interactions with the vendor.
- Reduced activities: coding, debugging, unit testing, code inspections.
- Modified activities: design focused more on how to fit pieces together rather than the internal workings of different modules. Architecture issues (compatibility, configurability and integrability) must be considered.

These differences have several implications.

- New activities require new professional skills and guidance. This means, for example, that employees must either be trained in the administrative, commercial, and other non-technical issues that come up during vendor interaction, or be given the necessary support to handle these concerns with the vendor. Other such new skills include COTS evaluation and integration.
- Traditional project estimation and tracking methods are less effective on COTS projects, as during estimation new activities tend to be overlooked or underestimated, and in tracking they are not reported, or reported under the ‘other’ category. This is what happened in early FDD COTS projects, all reporting an ‘other’ effort well above average.
- Processes tend to be looser. Because much of the standard SEL recommended process did not apply to COTS-based development and schedules were very tight, project personnel felt freer to loosen the process requirements.

4.4 Major issues.

Several observations reported can be summarized into two major issues: dependence on the vendor and flexibility in requirements. These issues are seen as the price that a COTS project must pay, consciously or not, for expected gains in schedule, effort and cost. Managing these tradeoffs is crucial to the success of a COTS project.

COTS-based development introduces a ***dependence on the vendor***, who ultimately makes all decisions on the capabilities of the COTS product, the schedule for releases, the architecture, the level of reliability, the documentation, and the service level. The purchaser usually has little or no influence on these issues as they affect the application.

Some of the related problems identified by project leaders were:

- Slippage in schedule because of delays in vendor release of the COTS (typically the delay is between the beta version and the official release).
- Unavailable, incomplete, or unreliable documentation on the product.
- An unpredictable learning curve with the product.
- “Vaporware,” i.e. functions that are promised but never implemented.
- Modifications made by the vendor that alter the compatibility of one COTS with other COTS, or the rest of system, or introduce new bugs. For several reasons the purchaser could be obliged to upgrade to the new version even when such compatibility issues exist.
- One-way communication with the vendor, with many questions but no answers.

The most commonly reported risk mitigation strategies were: having a close relationship with the vendor; and having a backup plan if the COTS fails (e.g. a second choice COTS or internal development). In another study of lessons learned from

COTS usage, conducted by USC and Software Metrics Inc. [4], vendor-related problems were the most commonly cited issue.

COTS projects must also accept ***flexibility in requirements***. At the moment a COTS is selected, some requirements are immediately satisfied, some other requirements become easy to implement, and others become difficult if not impossible to obtain. Since the typical goal of a COTS project is to reduce (cost, effort or schedule) as compared with a traditional project, the project must be ready to give up the latter category of requirements. In other words, the COTS selected drives the requirements to at least some extent. Some projects interviewed reported they ended up writing (sometimes rewriting) requirements *after* the COTS was selected.

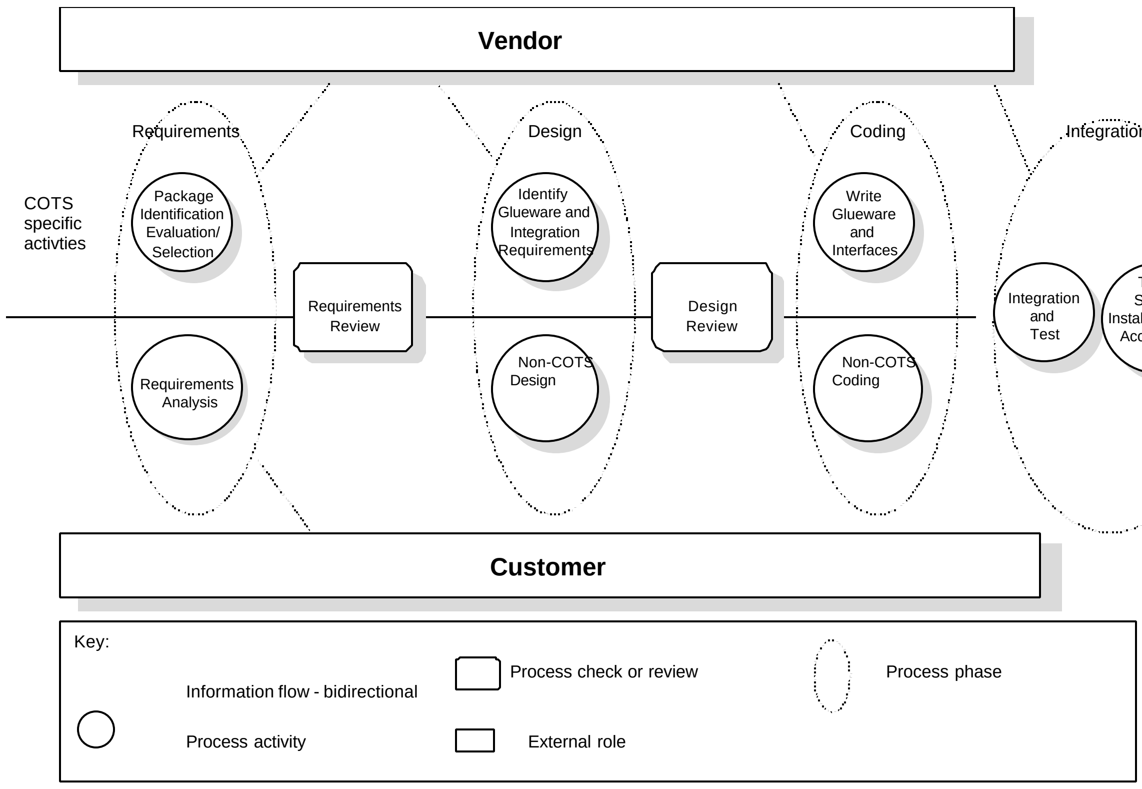
Others reported integration problems later in the project when requirements and COTS selection were not coordinated upfront. Still others reported major conflicts when they had little control over either system requirements or COTS selection (i.e. one or the other or both were mandated from some other organizational unit).

On the other hand, sometimes functionality was discovered in a COTS that was useful, even though the project had not originally planned to use it. Effectively managing the tradeoff between requirements and COTS selection seems to be a key to avoiding problems downstream, and to realizing the benefits of COTS. Interestingly enough, in the above-cited study at USC [4], flexibility in requirements was the second-most cited group of lessons learned from COTS usage.

5 A NEW PROPOSED COTS PROCESS

We present here (see Figure 4) the new proposed process for COTS-based projects developed by the SEL as a result of this study.

Figure 4. The new proposed COTS process.



The process is targeted to COTS-based projects using several peer COTS or one COTS integrated with a considerable amount of new developed software. In other words we exclude the case of adaptation of a single COTS (a turnkey system).

The process covers only development. The SEL will extend it as soon as enough observations of maintenance processes will be available.

We discuss phases and activities in section 5.1 below. We will concentrate on the differences and enhancements as compared with the actual process (Figure 2). In section 5.2, we discuss some of the broader process issues, such as project management and roles/responsibilities.

5.1 Process phases and activities

The main phases of the proposed process (represented as ovals in figure 4) are: requirements, design, coding and integration. Activities specific to COTS-based development are depicted by circles in the upper half of the diagram, while those not specific to COTS-based development are in the lower half.

5.1.1 Requirements

Previously, COTS selection was performed at the beginning of the design phase. Now, based on findings from the interviews and on actual processes, requirements analysis and COTS selection are performed together. Further, new activities are added and key decisions stressed. We list them in a logical order, while in practice many of them will be concurrent or iterative.

Make versus buy decision I. Using or not COTS in a project is a key decision that impacts all subsequent phases, and the success of the project. The decision should consider technical and non-technical issues.

COTS selection has thus far been treated more as a managerial than a technical decision. Instead, technical staff should be empowered to make decisions regarding COTS, as opposed to management dictates that a specific COTS shall be used. For many projects, COTS choices were made outside the project team, thus ignoring the team's expertise and experience. Requirements also often came from outside the team, and conflicts between requirements and COTS functionality often occurred later in the project.

For these reasons the make vs. buy decision should be recognized, formalized and justified. At this point, a first make vs. buy decision can be made, considering only non-technical issues, the flexibility in requirements and the willingness to depend on the vendor. If any of these prerequisites is not satisfied, the project should not use COTS.

Requirements definition. Requirements for the project are sketched out in brief detail. The goal is to guide the identification of COTS. When the domain of an application is stable and well known, this activity could be skipped, as the requirements and the COTS available are pretty familiar.

COTS identification and selection. COTS are identified and evaluated using vendor documentation, reviews, peer experiences and suggestions. The goal of this activity is to reduce the number of candidates to two or three to be deeply evaluated. Clearly, the number of deep evaluations must be kept low for cost and schedule reasons.

COTS familiarization. The COTS selected above are actually used. The projects interviewed considered this activity essential to better understand the functionalities available (not just the ones claimed), their quality, and architectural assumptions.

Feasibility study. In this activity a product is described at a level of detail sufficient to make the second make vs. buy decision (see below). The description should consist of a complete requirements definition, a high level architecture, an effort estimation, and a risk assessment model. The high level architecture allows the team to sketch dependencies among COTS, incompatibilities and integration effort [15,16]. Integration effort is an input for the effort estimation model, while incompatibilities, vendor dependability, COTS dependability and other factors are an input for the risk model.

The feasibility study should be repeated for a product without COTS (the make solution), and one (or more) products with a COTS. As an example, let's assume that three variants of the product are studied, without COTS, with COTS A, and with COTS B. In real cases more or fewer variants could be studied. Using different combinations of COTS could be analyzed in the same way.

Make vs. buy decision II. At this point the make vs. buy decision can be reviewed at a much deeper level of detail. The attributes considered for the decision are the requirements satisfied by a product variant, the estimated cost, and the estimated risks. The algorithms used to guide the decision process come from [6,8]. The result of this decision is the product variant that will be developed, including requirements and COTS selected. Each variant represents a different tradeoff among requirements satisfied, risks accepted and cost. This make vs. buy decision analyzes in detail these tradeoffs.

The process sketched above has several variants, which are currently undergoing experimental validation dedicated to understanding under what conditions each variant is more suitable. In general, though, the process outlined above formalizes and stresses the importance of key activities and decisions in a COTS project.

The phase ends with a requirements review. Reviewing requirements with the customer is a fundamental step in traditional software engineering. In COTS projects several decisions (COTS selection, requirements satisfied) are made early, with limited information available. Therefore the requirements review becomes, if possible, even more important. The review is guided by a checklist covering the main decisions made in this phase, and the assumptions they are based on (risk, cost and requirements).

5.1.2 Design

Some parts of design (definition of high level architecture, analysis of integration issues) were anticipated in the requirements phase. However, these activities are repeated here at a much lower level of detail, as all effort is concentrated on fully designing the product variant selected.

Design includes a high-level design activity where one of the main concerns is defining the integration of COTS and newly developed software. This is particularly demanding when several COTS are involved, each one with, possibly, different architectural styles and constraints.

The phase ends with the COTS design review. This is a typical design review for the traditional part, but covers other aspects too, essentially the decisions about architecture, COTS integration and glueware. At this point the make vs. buy decision is reviewed too. More information is available now, essentially about the COTS selected and about integration issues. Risks, the integration effort and overall cost are re-estimated and the decision re-assessed. It is possible that at this stage it becomes clear that integrating the selected COTS is impossible, requiring a loop back to the requirements phase.

5.1.3 Coding, Integration

Coding deals with parameterizing and possibly modifying COTS, and writing glueware. Integration gradually puts the pieces together - COTS, glueware and traditionally developed software - to assemble the final application. The main decisions have been made in the Requirements and Design phases, so Coding and Integration are dedicated to implementing these decisions. However, problems may be discovered. In this case the decisions could be reconsidered, with backtracking to Requirements and Design.

5.2 Cross-phase issues

There are a number of issues that are relevant through all phases of the process. Two areas that are of particular interest when talking about COTS projects are project management and the changes in roles and responsibilities.

5.2.1 Project management

Project management is probably the most impacted of horizontal activities (i.e. activities performed throughout the development process). Project estimation and tracking both have to consider new activities. Estimating their duration is currently a complex task, due to the limited amount of existing experience and the few estimation models available. Project estimation is performed using guidance from [15] to build and calibrate local estimation models, using the estimated amount of glueware to be produced as a predictor of integration effort. Project tracking is easier to accomplish, as it only requires modifications to the effort accounting procedures.

5.2.2 Roles

We describe here new roles and responsibilities peculiar to COTS projects. One role (the COTS team) is at the organizational level, while the other is at the project level.

COTS team

A group or a person, depending on the size of the organization, should concentrate on the following COTS-related skills and activities. Single projects cannot afford to build these skills individually. The team acts as a repository of history, knowledge and skills about COTS, and offers them to projects as a consulting activity.

- **Evaluation and selection of COTS**. Evaluations done by individual projects tend to be narrow in scope, concentrating only on those packages with which the project team members are familiar. Furthermore, unbiased evaluations require techniques and skills that projects cannot have.
- **History of COTS evaluations**. These are organized in an easily accessible catalogue of COTS known to the organization, describing concisely the function provided, vendor, cost, location, and projects using it. The real difficulty of this task is the rapidity of changes in the market place that makes the catalogue rapidly obsolete. The COTS team should rely on the COTS evaluation notebooks from individual projects as the input to this catalogue. However, the information from the notebooks must be merged, summarized, and generalized by the COTS team in order to provide a catalogue that is useful to the entire organization.

- **COTS usage**. A project becomes more readily proficient with a new COTS if it can access the experience of other projects that used it in the past. The COTS team normally does not have this experience, but can act as a contact point between projects.
- **Procurement**. Procurement of COTS requires administrative, managerial and commercial support that is missing in technical teams. The COTS team defines a repeatable process for vendor interaction. Part of the process is documentation of interactions with the vendor, an important record for the project manager.

Interface with vendor

A project should design a single point of contact with the vendor. The role is supported by the COTS team as far as non-technical, procurement skills are needed. The role records all interactions with the vendor and follows a defined and documented process. The role is also essential to building a partnership with the vendor, a key factor for success.

6 CONCLUSION

We have analyzed 15 COTS projects, performed at the Flight Dynamics Division at the Goddard Space Flight Center of NASA. These projects represent a variety of software domains and use more than 30 different COTS products. We think that the following observations can be generalized and could be useful to improve COTS-based software development in any context.

COTS-based processes differ considerably from traditional software development. Failure to recognize this by attempting to use traditional processes in a COTS project could be a factor in overall project success (or failure). It also could give developers leeway to loosen their processes, as the only processes they have are clearly inappropriate for the project at hand. On the other hand, new processes and activities have to be defined and given guidance.

New activities identified in COTS-based processes are: product evaluations, product familiarization, and vendor interaction (of technical, administrative and commercial kinds). New roles are the COTS team and the interface with the vendor. Requirements elicitation is deeply impacted. Selecting a COTS means selecting a (set of) requirement(s) and assuming it to be available cheaper and faster. The decision to use COTS, as opposed to making a product from scratch, is often made implicitly, is not formalized nor rigorously defined and analyzed under the points of view of cost and risks.

Effort in coding, debugging, unit testing, and code inspections is reduced, while design and testing activities are modified. Design focuses more on how to fit pieces together rather than the internal workings of different modules. Architectural issues, especially the difficulties of integrating components, must be considered. Testing has to deal with large modules, where no source code or documentation is available, and complex glueware.

Given a large number of new or modified activities, project tracking has to be adapted, and project estimation is especially difficult because of limited or missing historical data, and limited or uncalibrated estimation models. The projects interviewed *felt* that effort and cycle time could be reduced in COTS projects, but compelling evidence is still missing, especially because no data from the maintenance phase of projects is available yet.

Starting from the experience gained through the study, the SEL has defined a new proposed COTS process that tries to tackle the points listed above.

The new process defines in detail the phases where COTS-based development is most challenging: requirements, design and project management. In particular, the make vs. buy decision is formalized and guided, considering the overall tradeoff among cost, risks and provided functionality. The high-level research goal is to define under which conditions COTS-based development is superior to traditional development, including the maintenance lifespan.

As far as the study itself is concerned, it should be noted that quantitative data traditionally collected by the SEL was not sufficient to understand changes in the process. Using qualitative data was key, although this requires specific, more difficult tools for analysis.

7 ACKNOWLEDGEMENTS

This work was funded in part by NASA grant NCC5170.The authors wish to thank all of the developers and managers in the FDD and at Computer Sciences Corporation who generously allowed us to interview them. This study would not have been possible without them.

8 REFERENCES

1. Basili V.R., Caldiera C., Rombach H.D., Experience Factory, *Encyclopedia of Software Engineering*, Marciniak J.J. editor, Volume 1, John Wiley, 1994, pp 469-476.
2. Basili, V.R., D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, *IEEE Transactions on Software Engineering*, vol. SE-10, no.6, November 1984, pp. 728-738.
3. Carney, D. Assembling Large Systems from COTS Components: Opportunities, Cautions, and Complexities. *SEI Monographs on Use of Commercial Software in Government Systems*, Software Engineering Institute, Pittsburgh, USA, June 1997.
4. Clark, Betsy, CBS Development: Guidelines Based on Lessons Learned, presented at the *Focused Workshop on COTS-Based Systems*, University of Southern California, February, 2001.
5. Glaser, B.G., A.L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Company, 1967.
6. Kontio, J., A Case Study in Applying a Systematic Method for COTS Selection, *Proc. of the 18th Int. Conf. on Software Engineering*, IEEE CS Press, March 1996.
7. Miles, M. B., A. M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook* , 2nd Edition, Sage Publications, 1994.
8. Morisio M., Tsoukiàs A., IusWare: A methodology for the evaluation and selection of software products, *IEE Proceedings Software Engineering,* June 1997, pp. 162-174.
9. NASA/SEL, SEL COTS Study, Phase 1, Initial Characterization Study report, SEL-98-001, August 1998.
10. NASA/SEL, SEL COTS Study, Phase 2, New proposed COTS process, SEL-99-002, November 1999.
11. NASA/SEL, SEL Recommended Approach to Software Development, Revision 3, SEL-81-305, June 1992.
12. Parra, A., C. Seaman, V. Basili, S. Kraft, S. Condon, S. Burke, and D. Yakimovich, The Package-Based Development Process in the Flight Dynamics Division. in *Proc. of the Twenty-second Software Engineering Workshop*, NASA/Goddard Space Flight Center, December 1997, pp. 21-56.
13. Software Engineering Institute, COTS-Based Initiative Description, available at <http:// www.sei.cmu.edu/cbs/cbs_description.html>.
14. Software Engineering Institute, COTS-Based Initiative Overview, available at <http:// www.sei.cmu.edu/cbs/cbs_description.html>.
15. Yakimovich D., J.M. Bieman, V. R. Basili, Software Architecture Classification for Estimating the Cost of COTS Integration, *Proc. of the 21st International Conference on Software Engineering*, Los Angeles, California, May 1999, pp. 296-302.
16. Yakimovich D., Travassos G.H., Basili V.R., A classification of software components incompatibilities for COTS integration, *Proc. of the 24th Software Engineering Workshop,* NASA/Goddard Space Flight Center, December 1999.