Simplified Use Case Driven Approach (SUCADA) for Conversion of Legacy System to COTS Package

Jose D. Montero
Utah State University
josedmontero@yahoo.com

YongSeog Kim
Utah State University
yong.kim@usu.edu

Jeff Johnson
Utah State University
jeffrey.johnson@usu.edu

Abstract

The conversion of a legacy system to a system based on a commercial-off-the-shelf (COTS) package demands a dedicated guidance. The assumption that it is just a matter of adopting a selected package may prove disastrous and even more expensive than building the system in house and from scratch.

Building a software solution based on a COTS package not only has its risks, but it is also different from a custom development effort, and it needs to follow a rigorous methodology for a successful implementation. Therefore, it is necessary to define how to solve some of the challenges that this type of project presents, and how to balance customer requirements with the features offered by the COTS package.

To successfully and efficiently convert a legacy system into a new system based on COTS package, we developed and present a methodology that utilizes a general process flow chart, simplified Use Cases, and a mapping to the COTS package functionality. We also present the findings of a case study on the applicability and effectiveness of the proposed methodology for the conversion of a legacy Laboratory Information Management System (LIMS).

1. Introduction

The adoption of commercial-off-the-shelf (COTS) packages has become a common practice in today's enterprise. There is also a "vibrant market today that delivers COTS software components that range from software development environments to operating systems, database management systems, and increasingly, business and mission applications" [1] (p.13). It is common to find among the guiding principles of today's IT organizations the policy of "*prefer buy to build"*. In other terms, if there is a viable COTS solution, buy it and implement it instead of designing a new in-house system. It is anticipated, as expressed by McKinney, that "in the future, most software development efforts won't be able to afford not to use off-the-shelf software" [12](p.17). COTS packages may be implemented as a stand-alone application that may upgrade or replace an existing legacy system, or as components on new larger system. We focus in this paper on the first scenario.

The conversion of a legacy system by using a COTS package differs considerably from the most traditional software development process. Further, developing software based on the inappropriate use of off-the-shelf software can cost more than developing the needed software functionality from scratch [12] (p.17). The many and abundant methodologies for traditional software development may prove not useful, at least if not adapted properly, for this type of software conversion. This is mainly because the COTS-based process includes new activities such as product selection, high level design, product adaptation, and integration. On the other side, there are other activities from a traditional system development that may not to be employed or that may need to be modified.

The adoption of a COTS package has not only its benefits, but also challenges, risks and pitfalls [3], [20], [12]. Some of the benefits for adopting COTS products include robust and improved software quality through continued testing by real-world users [3], reduced efforts in coding, debugging, and code inspections [14], and lower costs than custom development through cost sharing with many users [12]. However, the conversion of a legacy system to a COTS package involves much more than the selection of the package and the decision to implement it in the company. Therefore the application of a traditional system development methodology to the conversion of a legacy using a COTS package can be risky [17]. As expressed by Morisio, et al., "not modifying the traditional process can be a *failure* factor" [14](p. 9, italics added). In

1530-1605/08 $25.00 © 2008 IEEE

fact, COTS requires new processes and also new skills, roles, and responsibilities [21].

There are mainly three challenges and risks that system development teams should consider in the process of developing a system based on COTS. First of all, development teams should consider how to adapt the business processes to the conceptual framework of the COTS product. Note that COTS products come with their own architectural concepts that may not match those of legacy system. Therefore, using the COTS package requires adapting the business processes in the organization to the conceptual framework of the COTS product [12], [20]. Second, development teams should also be aware of that COTS packages are not designed to meet a specific project's specification, but to satisfy the needs of a market segment [17]. Therefore, they should try to keep a balance between the features offered by the COTS product and system specific functionality requirements by establishing a baseline architecture with specific COTS components [13]. Finally, development teams should consider how to reduce the risk of COTS' vendor dependency in the long term. It is well known that often good and objective benchmark data about COTS products is not available [13], [20], and hence organizations have very limited access to product's internal design [3]. Therefore, the life expectancy and success of the new system and its long-term support may be heavily dependent on COTS' vendor survival [12].

In addition, project managers, systems analysts, developers, and users involved in a conversion project often suffer from great pressures due to unreasonable or false expectations that the adoption of a COTS package may bring. Note that the conversion team should not be accused of "*paralysis by analysis*", not being able to move faster in the conversion process because of unnecessary analysis. Therefore, there is a real need for specific guidelines on how to execute the conversion process and how to eliminate unnecessary activities that do not add value to the conversion project.

This paper proposes a new methodology to convert a legacy system into a new system based on COTS package while minimizing afore-mentioned risks and maximizing the benefits of adopting COTS package in the conversion process. In particular, this paper shows how to solve some of the challenges and how to balance customer requirements with the features offered by the COTS package. It also presents the findings of a case study on the applicability and effectiveness of the proposed methodology.

The remainder of the paper is organized as follows: Section 2 presents a brief summary of some proposed methodologies. Then, we describe the proposed methodology in Sections 3 and 4, and detail briefly the specific case study where the methodology was used and its results in Section 5. We finally provide some lessons learned from the project and conclude the paper in Section 6.

2. Literature review

Several methods have been proposed for COTS-based software development. Most of these methods deal with the evaluation and selection process of COTS products. Some of the representative methods include OTSO (Off-The-Shelf-Option) [9], STACE (Social-Technical Approach to COTS Evaluation) [11] and PORE (Procurement-Oriented Requirement Engineering) [15]. Alves and Finkelstein [3] provided a quick review of these methods and stressed the need to keep the rationale of the decisions made over both the development life cycle and the evaluation process. This is precisely one of the premises taken by the Evolutionary Process for Integrating COTS-based Systems (EPIC) framework developed by the Carnegie Mellon University-Software Engineering Institute (CMU-SEI) for building, fielding, and supporting COTS based systems [1], [2].

EPIC is a framework developed to help organizations build, field, and support solutions based on COTS and other pre-existing hardware and software components. EPIC is based on the research of the Software Engineering Institute (SEI) COTS-Based Systems Initiative, CBS initiative [20]. The CBS Initiative has studied systems engineering techniques to design COTS-based systems and assess the risks of COTS-based program, suitability of COTS products, and appropriateness of COTS-based system designs.

A COTS based solution, according to EPIC, integrates the following elements:

- one or more pre-existing hardware and software components from COTS components, the legacy system, reusable libraries and other sources (e.g., freeware, shareware)
- any required custom code (including wrappers and "glue")
- appropriate linkage/interface to the broader organization's architecture and external systems.
- end user's business processes including any changes necessary.

The software development framework in EPIC is based on an adaptation of the Rational Unified Process (RUP) [2], [17]. RUP is an iterative software development process *framework* rather than a process. It was created by the Rational Software

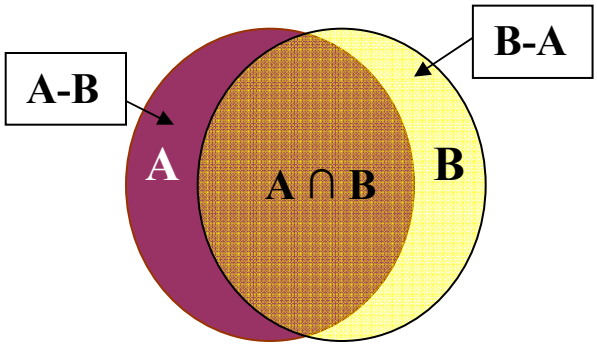
Corporation, since 2002 a division of IBM. The original idea was that like the unified modeling languages (UML), the unified processes (UP) could unify software development processes. Therefore, RUP is not a single concrete prescriptive process, but rather an adaptable process framework intended to be tailored by organizations and software project teams. Rather than giving a single process for software development, RUP provides a common set of practices for teams to choose from for an individual project. As a result a team's first step using RUP should be to define a *development case*, their individual processes. Each of four phases in a RUP project-Inception, Elaboration, Construction, and Transition involves a number of iterations. How many iterations in each phase is defined by each project. For each phase there is a number of activities to do, roles and their responsibilities, and a number of deliverables, what RUP calls artifacts to produce.

For the purpose of developing new software based on COTS packages, EPIC defines the four *spheres of influence* and proposes to simultaneously define and tradeoff among them throughout the life of a project. The four spheres is summarized as follows [17] (p. 3-4):

- **Sphere 1 - Stakeholder Needs and Business Processes** denotes requirements, end-user business processes, business drivers and operational environment.
- **Sphere 2 - Architecture and Design** denotes the essential elements of the system, the relationships between them, and how they fit with the enterprise system.
- **Sphere 3 – Marketplace** denotes available and emerging COTS technology and products, non-development items and relevant standards.
- **Sphere 4 - Programmatics and Risk** denotes the management aspects of the project. Key to these management aspects are the cost, schedule and risk of changing the necessary business processes

3. Simplified Use Case Driven Approach (SUCADA)

3.1. Objectives of SUCADA



A: Current legacy system B: COTS package

Figure 1: Legacy system versus COTS package solution

We first illustrate in Figure 1 the relationship between the legacy system (**A**) and the proposed COTS solution (**B**). The goal in replacing a legacy system (**A**) with a COTS package (**B**) is to maximize **A ∩ B.** It is the goal of the selection process to find the COTS package which guarantees the best match with the legacy system to be converted, in order words the largest **A ∩ B**. In practice we have found that usually **A–B** is not empty neither **B-A.** It is necessary to define strategies on how to deal with both sections of the converted system. Dealing with **B-A** is the easiest part. It represents functionality provided by the COTS package that is not used in the legacy system. As users become more familiar with the COTS package, they either decide to use this new available functionality or decide to ignore it. Therefore, we focus on only two subsets: **A ∩ B** and **A–B.**

In SUCADA, we do not cover the selection process of the COTS package either mainly because, in many instances, the selection of COTS package is completed even before the conversion project starts. For example, when a company decides to standardize certain type of systems and prefer to use a specific COTS package, the conversion project of a legacy system starts with the constraint to use the chosen COTS package. The major objectives of SUCADA include:

- Providing a rapid mapping of the functional requirements of the legacy system to COTS package: **A ∩ B**
- Providing a rapid identification of legacy system's functionality not supported by the COTS package (**A–B)** and defining an strategy on how to deal with them.
- Identifying interfaces to other company systems and defining how to implement them.

- Identifying and defining business processes for the new environment.

3.2. Comparison with EPIC

From our experience with RUP in other projects, we recognized the positive value of incorporating into software development projects best practices recommended by RUP such as: iterative development, risk analysis during the whole life of the project, use of component architectures, and continuous verification of quality. However, RUP's infinite variability can be cumbersome. RUP as a generic framework provides a long list of activities, artifacts (deliverables), roles, and responsibilities. It could be challenging to simply define the minimum absolutely necessary set of activities, artifacts, roles for a given project. RUP implementation may present challenges if a clear definition is not provided in the beginning of the project on which roles to use, who will be playing those roles, and what are the accepted subset of responsibilities for the selected roles.

The RUP-base EPIC solution could be very useful as a process framework. Those who adopt this RUP variation will need to spend a considerable amount of time deciding which activities to do, which artifacts to create, and which roles to use. Using all the suggested elements in the framework a project may make take a longer time to complete. The EPIC framework may prove too costly and too time consuming for small projects or in those conversion projects where the selection of the COTS package is already done.

The SUCADA integrates the elements of a COTS-software-based solution as indicated by EPIC. However, it focuses on the minimum necessary activities and tasks required to convert the legacy system with the approved COTS package. It provides a practical guideline for documenting the current legacy system's business processes and for mapping them to the functionality offered by the COTS package. Note that there is always a risk and a conflict when adopting an off-the-shelf solution for converting a legacy system. On one side, it would be ideal to leverage the use of the COTS package and adopt it with none or very minimum changes to it. On the other side, having an off-the-shelf solution in mind may lead to a simplified list of functional requirements that just mimic what the new system offers. It is recommended that the business processes should always dictate the functional requirements.

Note also that the requirements analysis is not exclusive for a custom-built system. However, by fully or partially utilizing a commercial-off-the-shelf package, project teams do not need to go into a great level of detail on functional requirements. In this regard the description of functional requirements to understand the legacy system and map that functionality to the COTS package can be more flexible and less specific [3], [5]. In our proposed approach, we take an iterative approach, starting at a high level description and expanding the granularity to the point that is sufficient to map it to the COTS package. The acceptance of the COTS package to serve as the basis of the new converted system conveys with it the acceptance of adapting the business processes to the conceptual framework of the COTS package. This may lead to important procedural changes.

4. Conversion process with SUCADA

The proposed methodology uses a simplified Use Case approach. As explained by Alistair Cockburn, "use cases are popular largely because they tell coherent stories about how the system will behave in use" [7](p.15). The core of this process is the application of Use Cases only at the business level. Using RUP's terminology, create only Business Use Cases. A Business Use Cases describes the interaction between the people, departments, other systems, and organizations within the user community of the legacy system. It defines "what" is needed. It does not go to the level of details required in a Systems Use Case. The creation of System Use Cases derived from the Business Use Cases is replaced by a mapping process. It aims for a quick definition of functional requirements directly from the legacy system business processes and mapping to the functionality of the adopted COTS package.

The steps of this methodology are as follows:

- Functional requirement analysis
- Requirements mapping to COTS
- Identification and definition of needed extensions or exclusions from new system.
- Design interfaces to other systems.
- Design new procedures, policies and standards for new systems.
- Design a transition plan
- Design and implement a training plan
- Implementation of new system.

4.1. Functional requirement analysis

It consists of the following two sub-steps:

- *Create a very general process flow chart*
- *Create Business Use Cases*

The goal of the general process flow chart is to create a high-level summary of the major processes and interfaces of the legacy system. It focuses on what the current system does. The ideal scenario is to define each process at a high level sufficient to list the specific tasks that are executed for each process. It does not need to go to the level of activity diagrams used for modeling the logic of the system.

For each major process or interface a Business Use Case is created. In the general process flow chart a hyperlink may be added to the corresponding Business Use Case for easy navigation between both documents. The Business Use Cases are designed to identify the specific task performed currently for the associated process. Use Cases should help in distinguishing later between true requirements and carry-over processes from the legacy system. This distinction should help in step 3: *Identification and definition of needed extensions or exclusions from new system.*

Figure 2 provides an example of a partial general flow chart. The model used can be changed but the aim should be the same: get a rapid high-level description of the legacy system. Here the process, the data, and the interfaces of the current system are documented.

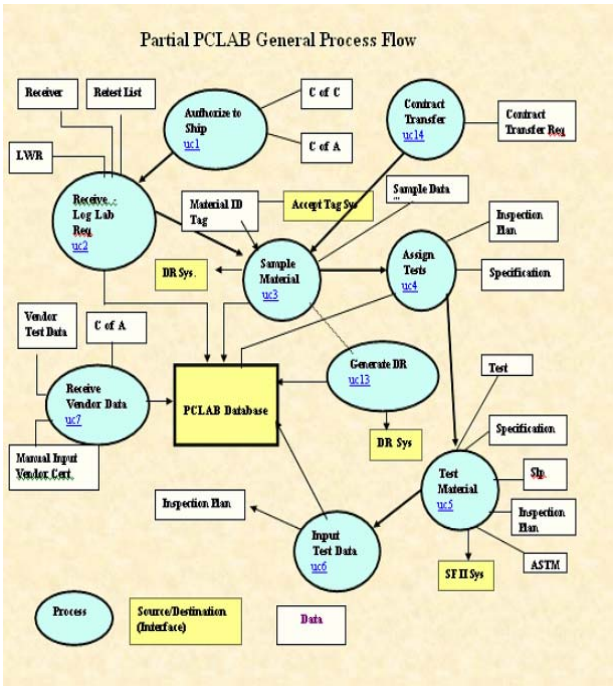


Figure 2 Legacy system general process flow (partial)

Since the introduction of Use Cases in the Unified Modeling Language [8] a lot has been written about what sections to include and about the format of the Use Case [4],[7]. As part of the Rational Unified Process (RUP) the user gets a template for Use Cases [10]. Cockburn [7] provides several examples and a chapter dedicated to Use Case formats. We propose a simplified and modified Use Case format that may be used for the capturing of the business requirements and later can be used to map the tasks to the COTS package functionality. Figure 3 is an example of such Use Case template. This example presents a partial version of an actual Business Use Case. Some information has been changed to protect company confidentiality.

4.2. Requirements mapping

The detailed System Use Cases that would be used for an in-house built system is replaced by a mapping to the COTS package and definition of needed extensions or exclusions. The mapping process will map the business needs as documented in the Use Cases to specific functions in the COTS package.

The modified template for Use Case can be used to map each function of the given current legacy system to the COTS system. The modified template for Use Case also list all specific tasks required to accomplish each process. The level of granularity for each task varies from system to system and even from process to process.

4.3. Identification and definition of needed extensions or exclusions from new system

This task can be done concurrently with the mapping of requirements tasks. Using the modified template for Use Case can be listed the action items for what to do with those functionalities in the current system that are not covered by the COTS package. The subset **A-B** in Figure 1 represents the functionality (logic and data) that the current system has but that is not provided in the COTS system. This subset has two components:

- *Functionality with affinity to the generic scope of the COTS package*.
  This functionality could be logical or data requirements not covered by the COTS package

but that could have a close affinity to the general purpose of the COTS package.

- *Functionality beyond the generic scope of the accepted COTS package and with not affinity to its general purpose.*
  This could be carry-over processes or appendices that over time have been added to the current legacy system but that are not part of the standard functionality of the given COTS package. For example, in the conversion of a legacy laboratory information management system (LIMS) to an off-the-shelf LIMS, it may be found in the current system functionality that is not associated typically with a LIMS. The tasks associated with this category should be excluded from the conversion project and moved to another project.

While the goal is to minimize the development of software outside the COTS package, the specific user requirements may demand extensions of the COTS package. COTS packages should provide the capability for adding extension. As expressed by Potts, "by having to deal with a large and possible heterogeneous customer population, off-the-shelf products have to be user customizable or extensible" [19].

Two forms of extensions to the features offered by the COTS products may be needed: functional or logic extensions and data base extensions. Logic extensions are functional components not provided by the COTS system. The data base extensions may be new fields to tables provided by the COTS products data base or addition of entire new tables.

A set of strategies should be defined on how to implement both types of extensions. It is generally not recommended to modify the COTS product even if that option is available. Modifying the base code of the COTS

Process: Receive – Log Process	
Number	2
Author	JDM/ DR
Last Update	March 18, 2005
Actors	Chemist, Lab Technicians, ERP-SYS1, SF-II, MSDS
Brief description Receive Planning and log required work. Receive laboratory work request and log required work. For new materials the user gets printout with Receiver from the ERP-SYS1 system. The automated interface with previous MRP systems has been severed. For re-testable materials the user pull list from ERP-SYS1. No automated interface exists now. Other LWR's (laboratory work requests) are received from several Work Centers. Using the information found in the Inspection Plan, the material is logged into the system. The MSDS system is accessed to obtain safety and health information. A request for Planning is added to a list for most types of material. A batch application on VAX2 reads the list and triggers the MIPS system to print the Planning. Files are sent by way of FTP. For some in-process tests the user gets the testing instructions from the Shop Floor system (SF-II), print them out. For those in-process tests that use SF-II, the user types back to SF-II the results of the tests. No automated interface exists. Most tasks (or tests) are assigned at login. Possible tasks for a material are listed and the user selects each task. Material could be assigned to a hot list when the material is logged in. The estimated time to complete and the date the material is needed is input. When the task (test type) indicated in the MIPS or SF-II I.P. is not in the PCLAB DB the lab technician then input that test to the PCLAB DB.	
Constraints None.	
Initialization Receiving of laboratory work request or after user gets list of re-testable materials.	
Termination Once requested work is logged into system.	

Post Conditions
Future Explore possible automated interfaces to ERP-SYS1, MIPS and SF-II in both directions. An accurate list of material coming up for retest is important. The laboratory needs to know the status of material, whether they may proceed with testing or not. Obtain a standard I.P. from MIPS and from SF-II Add Lab test requestor Id and other to be notified electronically when work is done.
Notes

System Functionality Mapping

Task	New System Mappings	Notes
Get Receiver from ERP-SYS1 for new material	Use Nautilus File Log-in API	Research if this is received from SLIMS??
Process New Stock Number	Create Nautilus workflow	Currently done by QDEs. Recommend PQEs do this in the future (only new stuff)
Get material to retest from ERP-SYS1	Use Nautilus File Log-in API	
Log Lab Request per test type: In Proc, Std, MRI, Eng Spec	Use Nautilus' Sample log in. Some may be defined as SDG.	List of materials and samples that must be treated as a group is given.
Print MIPS's I.P.	Outside scope In future use electronic IP and buy-off.	Create new business process for the future
Add task (test type) to DB if test in IP is not in DB Assign	What to do depends on result of the business process review underway.	Create new business process. Define who is authorized to create a new test.
Flag request for Accept tag	Not necessary for Nautilus to flag anything.	

Action Items

1. Define how MIPS's I.P. will be printed and create new business process or modify existing one.
2. Create new business process to define how to add tasks to DB and define who is authorized to create a new test.

Figure 3. Example of simplified Use Case (partial)

package will make difficult the implementation of future upgrades to the COTS package. These upgrades, which may arrive in many cases unsolicited, are often dictated by the market demands and not necessarily from demands of the legacy system to convert. The extensions have to be created in such a way that new upgrades to the COTS product can be implemented without affecting the added extensions. Consideration also must be given to options on how to eliminate the added extensions when the upgraded COTS product includes those missing functionalities or data fields.

On the logic additions, it may be sufficient to create a new library with the new logical components. Sometime the COTS package may present constraints such as which language to use for those logical extensions.

On the data base extensions, it is necessary to define a strategy for adding new tables or for adding fields to existing tables in the COTS data base. For adding new tables, good strategies include using distinctive names for tables, and keeping them in a different data base schema separate from the COTS data base schema(s). For adding fields to existing COTS data base tables, it is best not to add them to the actual table on the COTS data base. A better solution is to create a new table in a new data base schema with same COTS original table name plus a differentiating suffix, e.g., TableA and TableA_Ext. Here TableA_Ext holds the new fields that are not provided by the COTS' original TableA table.

4.4. Interfaces design

The analysis of how to implement the interfaces of the new system to other company systems may begin during the *Mapping of requirements to COTS* step. However, it is better to give special attention on how to define them. Both the input to and the output from the COTS package may dictate important changes on how to communicate with other systems. Also, legacy systems have usually been in existence for a while. Now, since the time the legacy system was written to the present day there may have been important changes, for example:

- *Some interfaces may be not needed any more.*

This is particularly true if during the *Identification and definition of needed extensions or exclusions from new system step*, that functionality was excluded from the new system.

- *New application integration methods may be now available*

It is possible that the company may have adopted more modern technologies for application integration, e.g. messaging systems or web services that may provide new ways to create the system interfaces.

- *New ways provided by the COTS package*

The adopted COTS system may provide new ways to connect with it.

4.5. Policies and standards design

The conversion of a legacy system to a COTS package may involve the modification or changes to existing procedures and it may require new ones to be written. As it has been mentioned before, using the COTS package requires adapting the business processes to the conceptual framework of the COTS product which may imply changing business processes in the organization [12]. These changes may well be extended to policies and system controls. This step will involve identifying those changes. It may include:

- Define procedures and policies that may be required for using new system.
- Define new naming conventions
- Determine groups and access privileges to the new system.

4.6. Transition plan

This step is to create a plan to transition to the new system and may involve:

- Defining what to do with existing Data Base and other information of the current legacy system. Among the questions to answer are: are the historical data in current data base going to be migrated to the new data base? Is it necessary to archive the data in the existing data base for auditing or other purposes?
- Defining transition method: a parallel run versus a cut-off and start with new system.
- Creating a couple of processes with the new system in order to get a feeling for the new system and to identify complexities and plan for time to implement them.
- Defining plan for creation of other processes.

4.7. Training plan

This step is to define the training plan for the new system and may be done concurrently with the step to define a transition plan. It may involve tasks such as:

- Organizing workshops to introduce the new system and to allow users to further explore the system and ask questions in a more informal atmosphere.
- Defining a training schedule and groups composition

4.8. Implementation

This step is to implement the strategy specified in the "*define a transition plan step"*

5. Application of proposed methodology

The proposed methodology was used at a major aerospace company for converting a legacy Laboratory Information System to an off-the-shelf Laboratory Information Management System (LIMS). The legacy system was developed in 1998 in Application by Form (ABF) from Ingres. The system was used in a material testing laboratory. The DBMS in use was Ingres and it was running in a non-supported VAX system.

The company had decided to adopt Nautilus, a product of Thermo Scientific (http://www.thermo.com) for conversion of several in-house built laboratory management systems. Then the selection process of the COTS package was not needed for this specific conversion. It had also been decided to use Oracle as the DBMS.

Nautilus is designed to be a general purpose, configurable and extensible Laboratory Information Management System (LIMS). It is intended to manage the creation or login of samples or specimens, assign tests to them, and acquire and store the testing results. Nautilus uses a three-tier architecture with a database server (Oracle or SQLServer) backend, a Microsoft Windows Powerbuilder client, and an Microsoft Windows "background" processor machine. Nautilus uses a dynamic data lifecycle and hierarchy, workflows, templates, and syntaxes; security configuration of groups and roles; static data, and reports to configure and manage these testing results.

The documentation of the existing legacy system was very poor and was not available in several areas. Further, most available documentation existed in terms of comments in program code. Two initial challenges for the project were first, how to document the current system tasks, and second, how to map them to the functionality offered by Nautilus. After working with key users, well experienced in the use of the legacy system, we were able to create the general processes flow. This tool proved to be very effective to get quickly a high level view of the system, its processes and its interfaces. For each process in the general process flow chart we created its Business Use Case. Here we were interested in listing and describing the tasks associated with the business process. We used an iterative and refinement approach. We started with a general list of tasks. In successive iterations we increased the granularity of the description of each task

and in some occasions tasks were further split. We stopped at a point where we believed was sufficient for both describing the functionality of the system, and for mapping later the tasks to Nautilus' specific functionality.

In a similar way we identified the interfaces of the current system and the data used in such interfaces. Some interfaces were later eliminated when they were part of functionality on the legacy system that was removed from the conversion project because it did not have any affinity with the nature of a LIMS. Others were changed because a new applications integration messaging system was now available. It was decided that the interface with other systems was to follow a loose coupling model, preferably using the available messaging services.

And finally, other interfaces changed due to the new functionality offered by the COTS package. Nautilus for example, offers easy login of samples, and also the facility to write scripts for automatic parsing of test results files.

The modified template for Use Cases was a valuable tool for describing in detail each process; for detailing the tasks involved in each process; for mapping the tasks to the new LIMS; for identifying which tasks or entire processes to exclude from the conversion project; and for identifying the required logical and data extensions.

Several processes and some tasks on some processes were identified as no pertaining to a LIMS. A couple of examples are: "Receive vendor data" and "Authorize to ship materials" processes. Decisions were made to move those processes to new or existing systems. Extensions to the functionality provided by Nautilus were identified. The logical extensions had to be written in a .Net language due to the constraints imposed by Nautilus. Data Base extensions, both in new tables and new data fields to Nautilus tables were implemented as described in the proposed methodology.

Once the mapping of tasks was completed it was easy to start creating workflows and other deliverables offered by Nautilus. For this, we started with models for creating *Sample Work Flows* and *SDG Workflows*.

There were important changes on the way to perform the tasks of the legacy LIMS system. It was not always possible to find a one-to-one mapping between the tasks on the legacy system and the functionality offered by the COTS package. This was due, as expected, to differences in the architecture of the legacy systems and Nautilus. This implied the creation and documentation of new processes. The same modified Use Case form was used to write notes on action items that were needed a follow up.

A small group of key users was selected for the creation of the processes flow and Use Cases with a team of system analysts and developers, and also for implementing new processes with Nautilus.

6. Conclusions

Existing methodologies for adoption of a COTS package are very useful particularly on their proposed methods used during the selection process of the COTS package. When such selection has been already made for whatever reasons, or when the legacy system is small or medium size, some of the existing methodologies such as EPIC may prove too costly and too time consuming. The number of activities and deliverables to create proposed by EPIC is too large and could be cumbersome.

The proposed methodology provided a way to go directly to what was needed: a rapid mapping of the processes and their tasks of the legacy system to the adopted COTS package. It provided a guideline on where to start and it was easy to follow. The simplified Use Case form was a key document and its multiple purpose usage was very effective to hold in a common place the list of tasks for each process, their mapping to the new system or removal from the target system, and notes for action items.

SUCADA helped on the creation of a minimum set of activities and deliverables. Its simple design may be adapted to the specific needs of another conversion project. For example, the level of granularity for detailing tasks may be adjusted according to the project needs. Still, the recommendation is to keep it at the level that is just sufficient to do the mapping to the functionality of the COTS package. For this, it is necessary to have in the team working in the mapping an expert on the adopted COTS package. In our case, we enjoyed having team members who were very familiar with Nautilus.

References

[1] C. Albert, C. and L. Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview. Key Elements in Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions", *Technical Report, CMU/SEI-2002-TR-009, ESC-TR-2002-009*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[2] C. Albert, and L. Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC). Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions", *Technical Report*, CMU/SEI-2002-TR-005, ESC-TR-2002-005, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November 2002.

[3] C. Alves, and A., Finkelstein, "Challenges in COTS decision-making: a goal-driven requirements engineering perspective", *Proceedings of the 14th international conference on Software engineering and knowledge engineering SEKE 2002*, ACM Press, New York, NY, USA, 2002.

[4] K. Bittner, and I. Spence, *Use Case Modeling*. Addison-Wesley: New York, NY, USA, 2002.

[5] D. Carney, "COTS Evaluation in the Real World", *SEI Interactive*, Carnegie Mellon University, December 1998.

[6] D. Carney, D., "Requirements and COTS-Based Systems: A Thorny Question Indeed", *SEI Interactive*, Carnegie Mellon University, June 1999.

[7] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley: New York, NY, USA, 2001.

[8] M. Fowler,*UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Third Edition. Addison-Wesley: New York, NY, USA, 2003.

[9] J. Kontio, "A COTS Selection Method and Experiences of Its Use", *Proceedings of the 20th Annual Software Engineering Workshop*, Maryland, November 1995.

[10] P. Kroll, and P. Kruchten, *Rational Unified Process Made Easy, The: A Practitioner's Guide to the RUP*. NY, NY: Addison Wesley, 2003.

[11] D. Kunda, and L. Brooks, "Applying Social-Technical Approach for COTS Selection", *Proceedings of the 4th UKAIS Conference*. University of York, April 1999.

[12] D. McKinney, "Impact of Commercial Off-The-Shelf (COTS) Software and Technology on Systems. Engineering", *North Star Chapter of The International Council on Systems Engineering (INCOSE),* August 22, 2001.

[13] M. Milligan, M., "Implementing COTS Open Systems Technology on AWACS", *CrossTalk, The Journal of Defense Software Engineering*, September, 2000.
As retrieved on April 20, 2007 from
http://www.stsc.hill.af.mil/crosstalk/2000/09/milligan.html

[14] M. Morisio, C. Seaman, A. Parra, V. Basili, S. Condon, and S. Kraft, "Investigating and Improving a COTS-Based Software Development Process", *Proceedings of the 22nd International Conference on Software Engineering (ICSE) 2000* , Limerick, Ireland, June 2000.

[15] C. Ncube, N. Maiden, "PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm", *International Workshop on Component-Based Software Engineering*, May 1999.

[16] P. Oberndorf, L. Brownsword, and C. Sledge, "An Activity Framework for COTS-Based Systems", *Technical Report CMU/SEI-2000-TR-010 ADA383836*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2000.

[17] C. Péraire, and R. Pannone, "RUP for COTS Package Delivery Roadmap". IBM December 6, 2005.
As retrieved on April 21st, 2007 from http://www-128.ibm.com/developerworks/rational/library/05/1206_perairе-pannone/ .

[18] K. Phillippe, *The Rational Unified Process: An Introduction*, 2nd ed. New York, NY: Addison-Wesley Object Technology Series, March 2000.

[19] C. Potts, "Invented requirements and imagined customers: requirements engineering for off-the-shelf software," *Second IEEE International Symposium on Requirements Engineering (RE'95),* 1995, p. 128

[20] SEI (2007). "COTS-Based Systems (CBS) Initiative.Software". Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. 2007.
As retrieved on April 23rd, 2007 from from: http://www.sei.cmu.edu/cbs/index.html

[21] SEI (2007b). "COTS-Based Systems (CBS) Initiative.Software". Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. Evolutionary Process for Integrating COTS-Based Systems, 2007
As retrieved on April 23rd, 2007 from http://www.sei.cmu.edu/cbs/epic/overview.html