Commercial-Off-The-Shelf (COTS): A Survey
A DACS State-of-the-Art Report

Contract Number SP0700-98-D-4000
(Data & Analysis Center for Software)

Prepared for:
Air Force Research Laboratory -
Information Directorate (AFRL/IFED)
32 Brooks Road
Rome, NY 13441-4505

Prepared by:
Maurizio Morisio
University of Maryland
College Park, MD 20742

Nancy Sunderhaft
Data & Analysis Center for Software
775 Daedalian Drive
Rome, NY 13441-4909

December 2000

Unclassified and Unlimited Distribution

Data & Analysis Center for Software	Data & Analysis Center for Software (DACS) P.O. Box 1400 Rome, NY 13442-1400 (800) 214-7921, (315) 334-4964 - Fax cust-laisn@dacs.dtic.mil http://iac.dtic.mil/dacs	DEPARTMENT OF DEFENSE UNITED STATES OF AMERICA

Abstract

The goal of this report is to survey the state of the practice in COTS-based development. The report discusses the definition of COTS and COTS-based system. Then it lists and discusses pros, cons and issues in COTS-based development. The central part is dedicated to survey methods and techniques that can be useful in COTS-based development. Finally a process to support COTS-based development is proposed, and emerging standards and techniques for component integration are discussed.

List of Figures

List of Tables

List of Acronyms

ACC	Architecture Coordination Council
API	Application Program Interface
BDK	JavaBeans Development Kit
C4IFTW	C4I for the Warrior
CASE	Computer Aided Software Engineering
CMM	Capability Maturity Model
COCOMO	COnstructive Cost MOdel
COCOTS	COnstructive COTS
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off -The -Shelf
DII COE	Defense Information Infrastructure Common Operating Environment
DCOM	Distributed Component Object Model
DoD	Department of Defense
DOT	Distributed Object Technology
EJB	Enterprise JavaBeans
ERP	Enterprise Resource Planning
FAQ	Frequently Asked Questions
FAR	Federal Acquisition Regulation
GCCS	Global Command and Control Systems
GCSS	Global Combat Support System
GOTS	Government Off-The-Shelf
GIS	Geographic Information Systems
GUI	Input/Output
I/O	Global Command and Control Systems
IPESO	Information Processing Engineering Support
Java/RMI	Java Remote Model Invocation
JCP	Jini Technology Core Program
JDK	Java Development Kit
JMS	Java Message Service
JSTK	JavaSpaces Technology Kit
LAN	Local Area Network
MS	Microsoft
OMG	Object Management Group
OMT	Object Modeling Technique
OOT	Object Oriented Technology
ORB	Object Request Broker
OS	Operating System
OT	Object Technology
RMA	Reliability, Maintainability, and Availability
SEI	Software Engineering Institute
SEL	Software Engineering Laboratoratory
STSC	Software Technology Support Center
TAFIM	Technical Architecture for Information Management

TCK	Jini Technology Core Platform Compatibility Kit
UML	Unified Modelling Language
WAN	Wide Area Network
WWW	World Wide Web
XML	Extensible Markup Language

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. 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. [Oberndorf 1997]

COTS software has the potential to save both time and money in the software development process. Each COTS software component used is less code that needs to be designed and implemented by the developers. However, the developer is faced with the problem of ensuring that the COTS product does perform the functionality that it claims to perform, that it does not intentionally perform functionality to be harmful to the system, that it will not adversely affect the system and that it can robustly respond to failures and anomalous inputs to prevent errors from propagating through the entire system.

Further, the use of COTS products in software development can require a considerable integration effort. Early estimation of this effort will help developers to choose the right COTS products and to decide whether to develop their own software instead of using COTS.

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? This report tries to give an answer to these questions.

2 COTS and COTS Systems

2.1 COTS Definition and Classification

The acronym "COTS" stands for Commercial-Off-The-Shelf, so firstly we must define what is 'commercial', and what is 'off-the-shelf'. The official definition of the term "commercial" is given in the Federal Acquisition Regulations (FARs); the summary here is (from [Oberndorf 1997]).

A commercial item is:

1. Property customarily used for non-governmental purposes and has been sold, leased, or licensed (or offered for sale, lease or license) to the general public;
2. Any item evolved from an item in (1) through advances in technology and is not yet available commercially but will be available in time to satisfy the requirement;
3. Any item that would satisfy (1) or (2) but for modifications customarily available in the commercial marketplace or minor modifications made to meet Federal Government requirements;
4. Any combination of items meeting (1) - (3) above;
5. Services for installation, maintenance, repair, training, etc. if such services are procured for support of an item in (1), (2), or (3) above, as offered to the public or provided by the same work force as supports the general public; or other services sold competitively in the marketplace;
6. A non-developmental item developed exclusively at private expense and sold competitively to multiple state and local governments.

As for the term "off-the-shelf," it can mean that the item is not to be developed by the user but is already existing. Such software can be used as

- development tools (e.g., compilers);
- integral parts of the new system (e.g., libraries);
- both development tools and parts of the new system (e.g., DBMS, compilers with run-time libraries, OS with APIs).

Even with these more detailed definitions, the term remains very broad. 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. Not surprisingly, different organizations and individuals mean different things for COTS.

We discuss now various characteristics and issues raised by the term COTS: origin, or who develops it, modifiability, or whether it can be modified or not, cost and property, its form of packaging, whether it is integrated in the final deliverable or not, and the type of delivered functionality.

Table 1. Origin and Modifiability of COTS.

Independent Commercial Item	Commercial Product with Escrowed Source Code		Oracle Financial		Microsoft Office
Special Version of Commercial Item					Standard compiler with specialized pragmas
Component Produced by Contract				Standard industry with custom systems	
Existing Components from External Sources		Standard gov't practice with NDI			Legacy component whose source code is lost
Component Produced In-house	Most existing custom systems				
	Extensive Reworking of Code	**Internal Code Revision**	**Necessary Tailoring and Customization**	**Simple Parameterization**	**Very Little or no Modification**

Table 1, proposed by [Carney and Long 2000], considers origin and modifiability of COTS and reports some examples.

Origin ranges from in-house to external vendor (independent commercial item). 'Special Version of Commercial Item' refers to a product developed by a commercial vendor and slightly modified for a client, where the modification may or may not be included in the next commercial release of the product. 'Component Produced by Contract' refers to subcontracting. 'Existing Components from External Sources' refers to components that are not developed internally, and usually not paid for either.
The authors do not discuss the issue of payment / cost/property, that seems to be sometimes mixed with the origin axis, while in our opinion it should be discussed separately.

Modifiability is split into five categories. Two of them assume access to code (extensive reworking, internal code revision), two (necessary tailoring, parameterization) imply some mechanism built into the COTS to modify its functionality. The authors report Microsoft Office in the 'Very Little or no Modification' category. However, this tool suite has a very rich set of mechanisms to modify it, from simple parameters (where to save files), to initialization files (templates for documents), to macros, to Visual Basic scripts (that involves programming). So Microsoft Office can be used as is, or with extensive modifications. It is an interesting example of a product that can fall into different categories of modifiability in function of the use one makes of it. In other words, the classification of a product depends both on the product and on its use.

Cost and Property. The COTS can be obtained for a price or free. Obtaining the COTS could mean acquiring property of it, or leasing it. 'It' means source code, executable code, or else, in function of how the COTS are packaged (see packaging). Related legal / commercial issues are liability for defects contained in the COTS, and responsibility for maintenance.

Packaging. The COTS can be packaged as
- source code library or program;
- linkable binary library;
- shared library (dynamic linkage library in MS Windows);
- stand-alone executable program;
- combination of the items above

The packaging form usually has an impact on modifiability. Clearly, the source code can be modified only if it is available. However, sometimes even if source code is available it is not modified, but used only for documentation and understanding.

A number of technologies, such as COM, CORBA, DCOM, ActiveX, JavaBeans, have been defined to standardize packaging and make it less dependent on programming languages, development environments, operating systems and networking protocols.

Integration in the Final Deliverable. Considering the product delivered to a customer or user, a COTS can be integrated in it or not. In the Microsoft Office example, Office is integrated. If we consider a project developing in C, the C compiler, CASE tools possibly used for analysis, design, testing, configuration management, quality assurance, project management, are not. However, some tools usually associated to the C compiler (i.e., the library of I/O functions) are probably integrated in the final product.

Type of Functionality. COTS offer a variety of functions, however they can be classified in two broad categories.

Horizontal. The functionality is not specific to a domain, but can be reused across many different application domains. Examples are DBMSs, GUIs, networking protocols, web browsers, etc.

Vertical. The functionality is specific to a domain, and can be reused only in the domain. Examples are financial applications, accounting, ERP or Enterprise Resource Planning, manufacturing, health care management, and satellite control.

Table 2. COTS Characteristics and Process Phases Impacted.

Characteristics	Effect	Phase Impacted	Risk
No source code available	No white box testing No source code inspection	Unit Test, Integration Test, Acceptance Test.	Reliability cannot be controlled
Functionality ready, not developed for the current project	Low cost	Cost Analysis	
[redacted]	Immediate/fast availability	Planning, Requirements	
	Training/ Familiarization needed	Planning	
[redacted]	Architecture, interaction mechanism are given, user must comply	Design, Integration	Incompatibility with COTS is discovered late, or is not solvable
	Functionality available is limited, and only partially customizable	Requirements	Requirements have to be dropped
Several partially overlapping products can be available	Evaluation and selection is needed	Requirements, Cost Analysis	
Functionality is developed by external vendor	The user has no influence on functionality removed (extreme case: all functionality is removed as vendor goes out of business)	Maintenance	(part of) Functionality offered by COTS must be developed
[redacted]	The user has no influence and no control on quality of functionality	Unit Test, Integration Test, Acceptance Test.	Reliability cannot be controlled
	The user has no influence on the pace of new releases of COTS (functionality added and removed, reliability increased or decreased)	Maintenance	
	The user has no influence on functionality added	Enhancements	COTS selected is suitable today, could not be suitable in the future if new functionality has to be added.

Horizontal COTS have been available on the market for a long time, experience and know how about them is usually widely available. As a result, using horizontal COTS is usually less risky and more common than using vertical COTS.

It should now be clearer that COTS is a broad term, with a variety of more detailed characteristics. Each of these characteristics can have an impact on a project, and implies a different set of risks. Table 2 reports COTS characteristics, effects, development phase impacted, and related risks.

In summary, the characteristics that imply more risks are the unavailability of source code and the fact that functionality is developed and maintained by an uncontrollable source. Note that the latter is also what makes the advantage of using COTS.

Given the immature state of the art, in the literature the term COTS can mean several different things. For the purpose of this document we restrict the scope of analysis to:

> Origin = independent commercial item or special version of commercial item
> Modifiability = no, or simple parameterization, or customization and tailoring (in other words, no source code available)
> Integration in final deliverable = yes.

With such a restricted scope we can perform a better, more focused analysis. Further, in industrial cases [Morisio 2000] this category of COTS is the one mostly used and that generates the most challenging problems. Finally, some issues are already treated under other headings in the literature. Subcontracting management is the topic to be considered when a third party develops a product or part of it, for a customer who then receives the source code and performs the final integration and maintains the system. Outsourcing could also be a topic to consider, when the third party develops and maintains the product and the surrounding services. CASE tools, software development environments are the keywords for tools that help developing but are not actually part of the final system. Code generation (compilers and products like Matlab being its more common implementation) should also be considered.

2.1.1 Examples of COTS

We report here (see also Table 3) some examples of COTS. The SEL COTS study [Morisio 2000] found that these categories of COTS were used by projects.

- Internet tools and browsers (Netscape Navigator, Internet Explorer, Eudora)
- Orbit determination and control, mission management packages.

Other third party products were used to build applications, but were not integrated in them. These products are part of the traditional development process since years and do not raise any peculiar problem to developers.

- CASE tools.

- GUI generators.
- Code generators – C compiler, development environments with libraries, editors, debuggers and generators (Visual C++, Visual Café, Mathlab, Mathematica).

Table 3. Example of COTS.

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

Outside the scope of the COTS study, studies in literature report usage of these other categories of COTS.
- Database management systems
- Geographic Information Systems (GIS)
- Office automation software, such as calendars, word processors, spreadsheets, etc.
- Operating systems, including low-level software such as device drivers, windowing systems, etc.
- Middleware (or services for communication and distribution)

2.2 COTS and Components

Component is a term now widely used, and probably as ambiguous as COTS. The relationship with COTS is strict, but COTS and components should be intended as two different concepts.

Components have evolved in the object-oriented community to mean relatively large pieces of software (typically comprising many classes and functions). In one meaning components are (similarly to subsystems) a tool for decomposing a system. They are also units of composition to assemble a system from parts.
In this sense a more formal definition is

> unit of composition with contractually specified interfaces and explicit context dependencies only. Components can be deployed independently and are subject to composition by third parties.

A characteristic often associated with components is 'independence from the context', where context means programming language, operating system, user interface, networking services, database and transaction services. A *container* is charged with hiding the implementation of these services while offering them to components in a standardized way. Ideally, a component can be plugged and work in any container. While this is more a target than the current situation, the concept of clear interfaces and defined dependencies from the context is shaping both industrial and research proposals. Hence the burgeoning of packaging and interface specifications for components, like CORBA, JavaBeans, EJB (Enterprise Java Beans), COM/DCOM, etc.

On the other hand COTS are not usually expected to have the independence from the context that components should have. In summary, not all COTS are components, while some components are COTS (i.e., they are produced by third parties).

2.3 COTS-System Definition and Classification

We have discussed above what constitutes a COTS. COTS are usually parts used to build larger systems. Here we change point of view, and we consider the system instead of the part.

A COTS-system is a computer based application that integrates one or more COTS. Carney [Carney 1997] identifies three types of COTS systems in function of 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 (ex., Oracle) but integrate other components, commercial or developed in house. The central COTS is the main part of the system, but integration of other components is key.
- *Integrated systems* are built by integrating several COTS, all on the same level of importance. The final system is not dominated by any single COTS, integration is the key to building the system.

3 Issues in COTS-Based Development

The **pros** in using COTS are easy to list and understand.

- *Functionality is ready and available*. The cost to buy the functionality is usually a fraction of the cost to develop it in-house. The delay to have the functionality is much lower too. And far less resources (personnel, know how and machines) are needed. The un-measurable sense of frustration implicit in redeveloping what is already available is avoided too.
- *Functionality is tested and working*. The functionality has already been used, possibly by a large community of operational users, and debugged extensively.
- *Maintenance is made by the vendor*. Again, cost, schedule and resources paid to the vendor for maintenance are usually a fraction of the cost for doing it in-house.

However, a number of **cons,** or at least **issues**, are increasingly apparent to the community of COTS users.

- *No access to source code*. No modifications are possible, no quality assurance techniques that require code can be applied by the user.
- *No control over the requirements for the COTS*. A single user has little or no control on the requirements implemented by the COTS. Actually, the vendor does not master completely the requirements either, as its goal is probably to satisfy the largest or richest possible market. Requirements are just subject to market forces.
- *No control on the quality of the COTS*. The COTS can contain faults, but the vendor decides when and whether to fix them. In general, also the availability and quality of documentation, training, consultancy, and support for the COTS are in the hands of the vendor. Documentation can be inconsistent with the COTS, promised functionalities can be missing (‘vaporware’).
- *No control over COTS updates*. This is a problem of periodic updates of COTS products, which can be costly or even inconsistent. Different versions of COTS products may not be compatible causing additional problems for developers [Swanson 1997].
- *Inflexibility of the COTS*. The COTS can be hard to adapt to the need, or not adaptable at all [Fox 1998].
- *Flexibility in requirements*. No control over the requirements for the COTS and inflexibility of the COTS means that the user must be flexible in his requirements, otherwise the COTS is not usable. Some functionality in the COTS could be an approximation of what is needed. In other words, the user must trade off schedule or cost for requirements [Morisio 2000].
- *Learning curve*. Although the COTS offers ready functionality, developers must become familiar with it to be able to use it. The more complex the COTS, the longer the familiarization period.

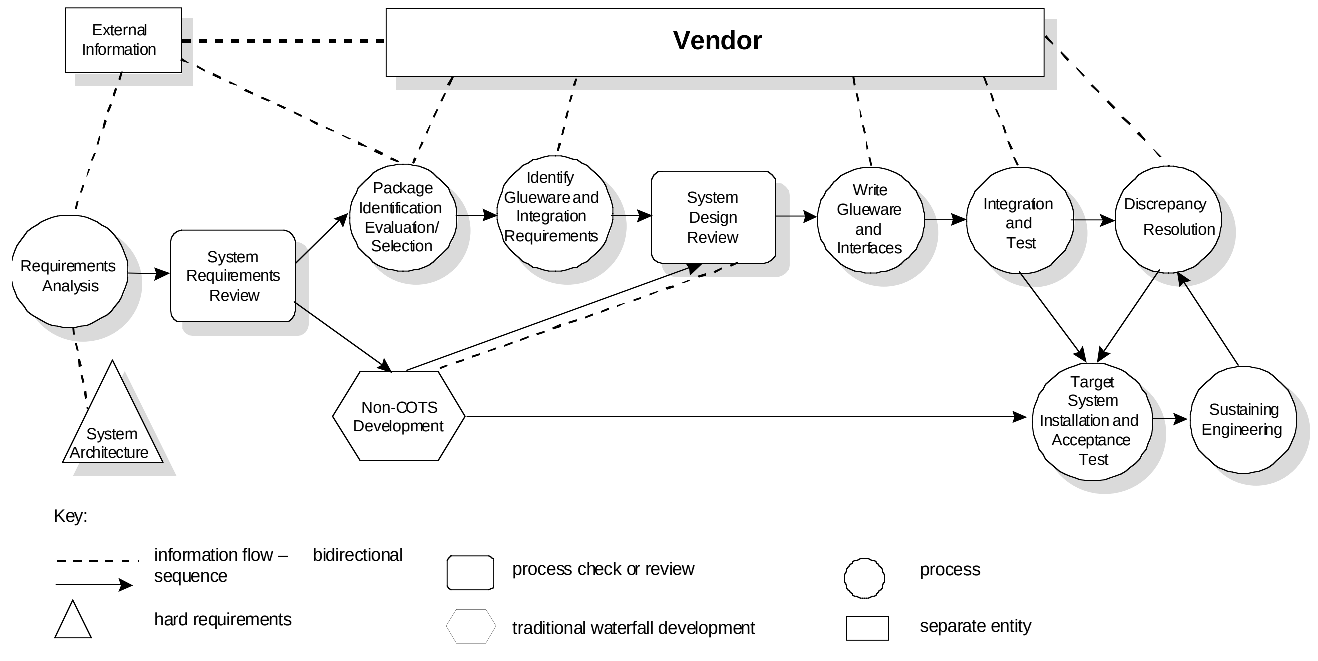
It should be noted that several of the issues listed above are not peculiar to COTS only, but are relevant for in-house development too. Learning is necessary whenever a new person joins the staff, the source code could be available but so complex to be virtually untouchable, resources and know-how could be unavailable to develop new requirements, documentation unavailable or inconsistent or not in sync, key staff capable of offering consulting and training gone. However, some of these issues can be worsened by the interaction with a third party like the COTS vendor.

To give more insight on these issues we report now an actual process followed by COTS projects and the issues they faced.

3.1 An Actual COTS Process

In 1996 and 1997, 15 projects using COTS were studied through questionnaires and interviews by the NASA Software Engineering Laboratory (SEL) [Parra 1997, SEL 1998] with the purpose of understanding the actual process used and the issues encountered. The process phases are the following (see also Figure 1).

Figure 1. An Actual COTS Process.


External Information
Vendor
Requirements Analysis
System Architecture
System Requirements Review
Package Identification Evaluation/ Selection
Identify Glueware and Integration Requirements
System Design Review
Write Glueware and Interfaces
Integration and Test
Discrepancy Resolution
Non-COTS Development
Target System Installation and Acceptance Test
Sustaining Engineering
Key:
information flow – bidirectional
sequence
hard requirements
process check or review
traditional waterfall development
process
separate entity

- Requirements Analysis

The earliest steps in COTS-based development are similar to traditional development—requirements gathering. 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. Early reviews of the requirements are crucial, even with a less formal process.

- System Requirement Review

This is the first verification step, aimed at checking the completeness and feasibility of system requirements.

- COTS Identification, Evaluation, and 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. 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 include 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

Glueware and interfaces as dictated by the system architecture, operating system and hardware 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 — Write Glueware and Interfaces

Most projects studied have an element of traditional development that does not depend on COTS or other packages. 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. Close contact with the vendor technical staff, or a competent Help Desk is essential during this development.

- Integration and Test

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.

- 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 and were then downloaded to the target system.

- Sustaining Engineering

The end of the configuration process is marked by the sustaining engineering, or maintenance, effort. No team that the study team interviewed had reached this stage.

The study revealed some interesting patterns. 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 (see the dashed lines in Figure 1). Also, there is a strong dependence on *bi-directional* information flow.
Also shown is a more 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.

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 (of technical, administrative and commercial kinds). 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.

- Project estimation and tracking are less effective, 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 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.

3.2 Major Issues.

Several observations reported in the SEL study can be summarized in two major issues: dependence on the vendor and flexibility in requirements. These issues are seen as conditions that a COTS project must accept, consciously or not, as a trade-off with 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 is the ultimate decision maker on the functionalities available, the schedule for releases, the architecture, the reliability level, the documentation, and the service level. Consequently, the purchaser has little or no influence on the above issues as they affect the application.

Some of the related problems identified by project leaders were:

- Slippage in schedule because of delays in releasing the COTS by the vendor (typically the delay is between the beta version and the official release).
- Documentation on the product is not available, or incomplete, or not reliable.
- The learning curve with the product is hard to estimate.
- "Vaporware," i.e., some functions are promised but never implemented.
- Modifications made by the vendor can alter the compatibility of one COTS with other COTS, or the rest of the system, or introduce new bugs. However, for several reasons the purchaser could be obliged to upgrade to the new version.
- Communication with the vendor can be one way, with many questions but no answers.

The most common risk mitigation strategies reported 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).

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., 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.

4 Techniques and Methods for COTS

This chapter is dedicated to presenting and discussing methods and techniques useful to tackle the problems listed in the previous chapter. Methods and techniques are listed under the process phase that more likely uses them.

4.1 Requirements

4.1.1 COTS Selection

Selecting a COTS is a decision that impacts requirements and design. It encompasses several sub-activities:

- COTS Identification. COTS should be identified with any possible practical means. Web research, specialized literature and exhibitions, peers suggestions.
- COTS Evaluation. The goal of this activity is to reduce the number of COTS among which the selection will be made to two or three, assuming to start from a set of half a dozen to a dozen. Information on these COTS is acquired through reading documentation, assisting to demonstrations, asking questions to the vendor. The evaluation should be formalized by using specialized techniques such as weighted average sums [Vincke 1992], feature analysis [Kitchenham 1997], analytic hierarchy process [Saaty 1980, Kontio 1996], and outranking techniques [Morisio 1997, Stamelos 2000].

While weighted average sums are the most popular technique used, it should be made very clear that each technique has constraints of application, so that no one technique can be used meaningfully in all cases. Conversely each evaluation case has specific goals and context that should drive the selection of a suitable evaluation technique. The common point to all techniques is that COTS are scored on a number of characteristics (or features, or attributes), then the scores are aggregated to rank COTS. Typically scores are given to the degree to which a COTS satisfies a given requirement (functionality or non-functional requirement).

Weighted average sums require ratio scales for the attributes. This excludes them from the most popular case of use, judgments like ‘satisfies fully, satisfies fairly, does not satisfy’ that are on an ordinal scale. It should be recalled that even if these judgments are expressed in a numeric scale like 3,2,1, the scale remains ordinal.

The analytic hierarchy process requires that attributes are independent, another condition that is not easy to satisfy in COTS evaluations.

Outranking techniques are less popular, but can be very powerful as they allow the evaluator to state "product A is better than / is equivalent to product B on attribute X". Usually an evaluator finds it more convenient to express a relative judgment than an absolute one. Further, a profile for an ideal product can be generated, then real COTS can be compared with it. Another advantage of outranking techniques is the capability of considering the evaluation process as a team activity by accepting scores from several stakeholders and aggregating them in various ways.

In many cases cost happens to be an important parameter in the selection. Nearly always in a project cost is traded for something else (functionality, quality, schedule). Therefore we suggest to keep cost separated from the evaluation, and to consider explicitly and formally any trade-off. Cost related issues are taken into consideration in the Project cost estimation activity.

- COTS Familiarization. Since the previous activity has reduced the number of candidates to two or three, it is now possible to use them for a while, until a good degree of familiarity is gained. The means to achieve this familiarity is using a demo version of the product.

- COTS Selection. The two or three topmost COTS are reevaluated with the same techniques used in COTS evaluation, but with much more insight as gained in COTS familiarization. A winner is defined. Project cost estimation is updated in function of the COTS selected. Reasonably what-if simulations are carried on with each of the topmost COTS, meaning that this is the time to merge project cost and COTS selection.

- COTS Purchase, Licensing. Finally, the COTS is purchased. In case of multiple licenses, the purchase and licensing process should be supported by specialized entities capable of obtaining substantial discounts [Reifer 1999].

4.1.2 Requirements and COTS Selection.

It is very hard to clearly separate these two activities, actually they go on in parallel. An initial requirements analysis clarifies the type of COTS that could be more suitable for the project. More in-depth analysis of COTS excludes some of them and proposes others. Selecting a COTS impacts the set of requirements gathered, some of them become easily satisfied, some more are included ‘for free’ in the COTS, some more could be identified as a consequence of the COTS selected.

An issue is the level of detail where to stop the analysis of requirements provided by a COTS. Ideally, the level of detail is the same as if the requirements would be satisfied by newly written code. In practice, when a COTS is selected, the tendency is to stop further analysis of requirements offered by the COTS and use it as the de facto description of such requirements. However, this approach implies that the COTS acts as driver of requirements, not the customer, and should be clear to both the customer and the project manager.

4.2 Design and Integration

The more a project is COTS intensive, the more design deals with how to integrate COTS together, solving possible inconsistencies among them. We discuss here design representations, integration incompatibilities, and integration techniques.

4.2.1 Design Representation

The Unified Modeling Language (UML) [Fowler 1997, Rumbaugh 1998] is not only an OMG (Object Management Group) standard, it is also becoming the de facto standard notation to document designs. Using it to document designs of COTS-based systems is therefore natural. The Catalysis method [D'Souza 1998] takes this point of view and marries UML with component based development (remark however that Catalysis considers components and not COTS, see section 2.2).

Component based development for Catalysis means building a family of software products from a kit of components. Some of the components may be adapted from existing systems. Some can be bought (or COTS).
The idea is that components should be pluggable together in different ways, rather like hardware components. This requires a small number of standard interfaces compared to the number of components: each component has one or more interfaces that can be connected to any other.

Catalysis provides:
- interface specification — allowing a chief architect to specify an interface standard, and third parties (who don't know each other) to make components that will interoperate with each other
- techniques to define component 'connectors' abstracting above the level of OO messages (sort of like multi-pin plugs);
- 'retrieval' techniques for relating the differing models that different components (especially bought-in or legacy components) usually have.

4.2.2 Integration Incompatibilities

A serious challenge for COTS integration can be a difference in architectural assumptions among different COTS products, and between the COTS products and the target system [Davis, Williams 1997], [Garlan et al. 1995], [Shaw 1995]. It was determined that using just four COTS products in one project with different architectural styles can increase the effort considerably [Garlan et al. 1995].

The first step in solving this problem is understanding and classifying the incompatibilities. Yakimovic [Yakimovic 1999] splits the problem into two parts: understanding and classifying interactions, and understanding and classifying inconsistencies.

The COTS interact with other system components, and with the system environment. The system components can be either software or hardware (excluding everything related to the environment, such as CPU and memory, but including devices directly controlled by the system, such as on-board devices) that are used by the system. The environment can be of the development phase, which includes compilers, debuggers, and other development tools; or it can be the environment of the target system, which includes Operating Systems, virtual machines (Java), interpreters (Basic), and other applications and utilities used by the target system. The parts of both environments can also be considered components. Figure 2 shows the different perspectives that can be used to classify these software component interactions.

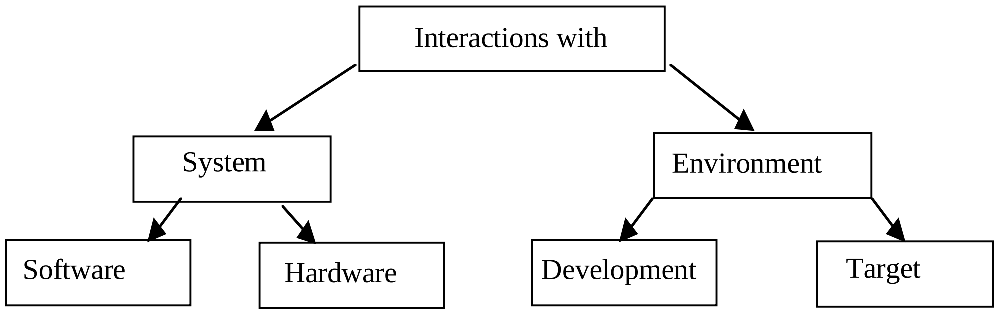


Figure 2. COTS Interaction Types.

Two main layers can be distinguished in the inter-component interactions:

Syntax, defines the representation of the syntax rules of the interaction, e.g., the name of invoked function; the names, types, and the order of the parameters or data fields in the message, etc.;

Semantic-pragmatic, defines the semantic and pragmatic specification of the interaction, i.e., what functionality and how is performed by the component, e.g., invoking the function "SQRT" calculates the square root of its only argument and returns it to the caller. However, in this work we do not consider semantic and pragmatic issues separately.

An incompatibility can occur either because of syntactic difference between two components, or because the interaction does not perform correct functionality due to some semantic or pragmatic issues. Further, incompatibilities on the semantic-pragmatic layer can be classified according to the exact number of components that caused the interaction to fail. Therefore, the following types of semantic-pragmatic incompatibilities can be considered:

- 1-order semantic-pragmatic incompatibility, or an internal problem, if a component alone has an incompatibility. For example, a component does not work properly, because of not matching the requirements or an internal error.

- 2-order semantic-pragmatic incompatibility, or a mismatch, if an incompatibility is caused by interaction of two components. For instance, a procedure that calculates the square root of a real number receives a negative argument from a caller; when it supposes that this is a proper argument (it is important that the caller does not have a 1-order incompatibility).

- N-order semantic-pragmatic incompatibility, or a conflict, if an incompatibility is caused by interactions of several components. For instance, several processes that together require more memory than the available amount, although each of them can be satisfied independently, they together cause a n-order incompatibility on the semantic-pragmatic layer in hardware interactions.

Table 4. Layers of Interaction Incompatibilities.

Type of component	*System*		*Environment*	
Type of Incompatibility	Software	Hardware	Developme nt	Target
Syntax	S.1	H.1	D.1	T.1
Sematic-pragmatic1-order	S.2a	H.2a	D.2a	T.2a
Semantic-pragmatic 2-order	S.2b	H.2b	D.2b	T.2b
Semantic-pragmatic n-order	S.2c	H.2c	D.2c	T.2c

Thus, according to the assumptions above, syntactical and semantic-pragmatic incompatibilities can occur in the system and environment dimensions. We can use Table 4 to capture this classification, where the cells are described below.

S. Interactions with Software

S.1. Syntax:

S.1.a. different types of information flow, e.g., control instead of data.

S.1.b. different types of binding: static, dynamic compile-time, dynamic run-time, topological, etc. As the result a component can not find another one.

S.1.c. different interface protocol: different number of parameters or data fields, or different types of parameters or data fields.

S.2. Semantic-pragmatic:

S.2.a. 1-order: internal problem. These incompatibilities appear when the COTS product does not match the required functionality (e.g., a function performs addition instead of multiplication), or due to its poor quality it still does not work properly (an internal error). On the other hand, it can be other software that is solely responsible for the failure of interaction with the COTS product.

S.2.b. 2-order: different assumptions between two components, including the synchronization issue. These incompatibilities are products of mismatch between the COTS product and other components surrounding it. Even when two components have correct functionality they can fail to work together due to some differences. (e.g., one object expects to receive the size of an angle in radians, but another sends the size in degrees, hardly can the result be correct; another example is a mismatch between an asynchronous and a synchronous component).

S.2.c. N-order: a conflict between several software components. Even when the COTS product works correctly itself and correctly interacts with other components, some incompatibilities can appear as the result of a combined interaction with several other software components. (e.g., an object that controls rotation of a spacecraft receives the command for rotating on n degrees from an commanding object, but occasionally there is another commanding object, which sends the same command at the same time, in the system. Every single interaction is correct, but the spacecraft rotates twice as fast as it should.).

H. Interactions with Hardware

H.1. Syntax:

H.1.a. different type of protocol. A software component can not work with a piece of hardware, because they assume different protocols (e.g., TCP/IP and Decnet).

H.2. Semantic-pragmatic:

H.2.a. 1-order: wrong functionality of hardware or the COTS component. A hardware component does not work correctly (e.g., a printer does not support the Cyrillic alphabet), or the COTS component causes a failure.

H.2.b. 2-order: different assumptions between software and hardware. An interaction between software and hardware components does not work correctly (e.g., a program tries to print a Cyrillic text, but the printer has a different coding for the Cyrillic alphabet, therefore the output will be unintelligible).

H.2.c. N-order: a conflict between several software components over hardware. An interaction among several software components and a hardware component does not work correctly (e.g., several applications simultaneously accessing a single printer).

E. Interactions with the Development Environment

E.1. Syntax:

E.1.a. Different component’s representation. The environment does not understand the packaging of a software component (e.g., a C program can not be compiled by a Fortran compiler).

E.2. Semantic-pragmatic:

E.2.a. 1-order: wrong functionality of the environment or the COTS component. The environment does not work properly (e.g., a defect in the compiler version), or the component has an error (e.g., a program can not be compiled because of a syntax error in it).

E.2.b. 2-order: different assumptions between the software component and the environment. A software component cannot interact with the environment (e.g., a program is written in an old dialect of a language and can not be compiled by a newer compiler).

E.2.c. N-order: a conflict between several software components over the environment. An interaction among several software components and the development environment causes an incompatibility (e.g. two or more C modules can not be compiled or linked together because of a name collision).

T. Interactions with the Target Environment

T.1. Syntax:

T.1.a. Platform type. The environment does not understand the packaging of a software component (e.g., a program uses another OS, or an interpreter can not run an program written in another language).

T.2. Semantic-pragmatic:

T.2.a. 1-order: wrong functionality of the environment or the COTS component. The environment does not work properly (e.g., the OS crashes), or the component has an error (e.g., a memory violation in a program).

T.2.b. 2-order: different assumptions between the software component and the environment. A software component does not interact with the environment correctly (e.g., the OS version performs some functions used by the component in a way other than expected by the component's developers).

T.2.c. N-order: a conflict between software components over the environment, including the control issue. An interaction among several software components and the environment causes an incompatibility (e.g., a conflict between two object-oriented frameworks in a one-process program for the control flow [Sparks et al. 1996]).

4.2.2.1 Solution Strategies

Different incompatibilities have different solutions, but generally we can find five groups of related problems with the proper solution strategies. We assume that one type of incompatibilities can cause problems in different groups. For example, a syntax software incompatibility can cause different types of binding, which can require a special architectural solution for the whole system, or it can be just a different order of parameters, which can be overcome by a simple wrapper. Thus, we can differentiate the following groups of integration problems:

- **Functional**. All the 1-order semantic-pragmatic incompatibilities that are caused by missing or wrong functionality. Re-implementation or modification of faulty components can solve these problems.

  Example. A graphics library does not provide a function for drawing circles. So it can be re-implemented using a function for drawing pixels.

- **Non-functional**. Some 1-order semantic-pragmatic incompatibilities can be caused by not matching to non-functional requirements, such as reliability, maintainability, efficiency, usability, etc. These problems are difficult to solve without reworking the component.

  Example. A sorting procedure is implemented using a relatively slow bubble-sort algorithm ($o(n^2)$). To make it faster this function can be re-implemented using the quick-sort algorithm ($o(n*\ln(n))$).

- **Architectural**. These issues constitute another class of problems and can cause changing the overall system's architecture, but the incompatibilities causing them are different. We consider the following architectural assumptions of software components with their respective incompatibilities: packaging (syntax development and target environments), control (n-order semantic-pragmatic target environment), information flow (syntax software), binding (syntax software), synchronization (2-order semantic-pragmatic software).

  Example. Java and C programs are to be used together in one system. Since they can not be compiled or linked together (different packaging of these programs), the resulting system will consist of two independent programs: Java and C that can interact through means of operating system, e.g., the C program can read requests from a file written by the Java program.

- **Conflicts**. Problems of this type are conflicts between components in the system (e.g., deadlocks). The related incompatibilities are n-order semantic-pragmatic software and hardware. The possible solutions can include changing the system's configuration without changing the overall architectural type (minor architectural changes, including monitoring components) and using glueware.

Example. Two message-based programs are to be used in one system. To prevent their conflicts over the incoming messages, a special monitoring component can be added to the system. This component will accept all messages coming to the system and send them to the proper program.

- **Interface**. These problems are incompatible interfaces between the components caused by some syntax and 2-order semantic-pragmatic software and hardware incompatibilities (other than major architectural). The possible solution is glueware.

Example. One program calls function "sqrt", meanwhile the mathematics library provides function "Sqrt" (just another name). The following piece of glueware (wrapper) can be used to solve this problem: float sqrt(float x) { return Sqrt(x); }. This function will provide a compatible interface for the program to call the library function.

4.2.3 Integration Techniques

Integrating COTS components into a functional system presents new challenges. The problems that arise with COTS integration are very similar to those seen in general with software reuse [Shaw 1995]. Since different components are written by different vendors with different needs in mind, they may need to be adapted to work properly together [Haines].

As discussed previously, COTS can suffer from general inter-component mismatches, for example, of representation, communication, packaging, synchronization, semantics, control, and other properties [Shaw 1995]. To overcome these mismatches between the components A and B the following techniques can be used [Shaw 1995]. However, not all of them can work for the black-box reuse (A and B in these examples are symmetrical, and it is usually possible to consider that A is a COTS software component and B is an in-house component):

1. Change A's form to B's form: completely rewriting one of the components to work with the other. Rewriting an in-house component in order to match the COTS component is feasible, but it can be very expensive depending on the in-house component.

2. Publish an abstraction of A's form: APIs publish the procedure calls used to control a component. Open interfaces usually provide some abstractions too. Feasibility of this approach depends on whether an abstraction of a COTS component is available.

3. Transform from A's form to B's form on the fly: some distributed systems do on-the-fly conversions from big-endian (the most significant byte of a word is stored first) to little-endian (the most significant byte of a word is stored last) representations. This approach

may require very intellectual and expensive software for on-the-fly conversion.

4. Negotiate to find a common form for A and B: modems commonly negotiate to find their fastest common protocol. This seems to be achieved for hardware only at this time.

5. Make B multilingual: portable Unix code will run on many processors. Implementing multilingual in-house software is feasible, but can be expensive.

6. Provide B with import/export converters: some applications provide representation conversion services. This can work if there are representation mismatches (data format, etc.).

7. Introduce an intermediate form: it can be used as a neutral base for components with different representations. Feasibility of this technique will depend on whether the COTS component supports an intermediate form.

8. Attach an adapter or wrapper to component A: some code can be written to leverage the difference between the interacting components. Writing a wrapper is a flexible solution; the wrapper can be tailored for any particular mismatch.

9. Maintain parallel consistent versions of A and B: A and B can maintain their own different forms. This depends on the availability of parallel forms of COTS software. Maintaining parallel forms for in-house software can be a problem as well.

Three techniques to adapt the COTS component are to wrap the component in a software container, use glueware to mediate component interactions and using bridges or adaptors to smooth over incompatibilities in the component interfaces [Brownsword et al. 1998]. These are all black-box techniques that can be applied without access to the COTS component source code.

Wrappers are software containers used to mediate access to the COTS component. They can be used to force compliance to programming standards, provide a standard interface to the COTS component can be swapped out to and upgrade or different vendor or restrict the available functionality of the COTS component [Vigder Dean 1997].

Glueware is used as middleware to bind components together. It can be used for control flow, to invoke the component’s functionality and do exception handling [Vigder Dean 1997]. This can also include code to resolve incompatibilities between two components. By acting as an adaptor or bridge, the glueware can allow the interaction of two components [Vigder Dean 1997].

Table 5. COTS Integration Techniques.

Technique	***Goal***
Wrapper	Mediate access to the COTS component, provide standard interface, restrict access
Glueware	Bind components together, exception handling, control flow
Adaptor/Bridge	Resolve incompatibilities in component interfaces

4.3 Testing

Testing is the process of reviewing a software process artifact with the intent of finding errors.

COTS software has the potential to save both time and money in the software development process. Each COTS software component used is less code that needs to be designed and implemented by the users (in this paper the user is defined as the buyer of the COTS product). However, the user is faced with the problem of ensuring that the COTS product does perform the functionality that it claims to perform, that it does not intentionally perform functionality to be harmful to the system (i.e., Trojan Horse), that it will not adversely affect the system and that it can robustly respond to failures and anomalous inputs to prevent errors from propagating through the entire system.

These needs have lead to research in ***unit testing COTS software***, ensuring the COTS product, standing alone, will perform as expected and ***COTS integration testing***, ensuring that the combination of COTS and custom software works together to produce a quality system.

4.3.1 Unit Testing

The goals of COTS unit testing are to ensure the component is functionally correct (i.e., it does what is specified), externally secure (i.e., the COTS unit will not carry out malicious commands from input data), internally secure (i.e., there is no malicious code hiding in the component) and robustly responds to anomalous inputs (so an error with the input to the COTS product does not necessarily propagate to the rest of the system). Unit testing of non-COTS software faces very similar issues and a number of techniques have been created to deal with them. Briefly we will look at some of these techniques in unit testing and then discuss techniques specific to COTS testing.

4.3.2 Unit Testing Techniques

Unit testing techniques can be classified into two different categories: white box and black box testing. Black box techniques approach testing from an external perspective and focus on testing with respect to how the specifications say the component should behave. White box techniques are drawn from an internal perspective and deal with testing the logic/structure used to achieve the behaviors [Hetzel 1984]. White box testing will require access to source code and design documents, something COTS component users do not have. Black box testing may only require the requirements the component needs to satisfy, something the COTS component user could supply.

4.3.3 White Box

Testing the structure of the object is the foundation of white box testing [Beizer 1995]. Reviewing code, requirements or design documents are all examples of testing, looking at a software process artifact to detect errors. Some strategies for white box testing include source code statement coverage, branch coverage and condition coverage. These techniques are effective at ensuring that all the code has been exercised by the test cases, no malicious code is hiding in the component and that the structure of the code is well designed. Statement coverage ensures that every source statement is executed at least once. It does not, however, require that each statement be executed in each possible state the code can be in. Branch coverage requires that every possible outcome of a decision statement must be exercised. Condition coverage requires that every condition within a decision statement must achieve all possible values at least once [DeMillo et al 1987].

These techniques, while effective at enumerating errors, are not feasible with COTS software since the user cannot expect to have any of the necessary artifacts. At best the user has a set of requirements that the system needs to be satisfied by the COTS software.

4.3.3.1 Program Slicing

Program slicing is a white box technique that uses data flow and control flow to produce a trace (slice) through the code that produces a specific behavior [Weiser 1984]. This slice can be used to reduce the amount of code that needs to be reviewed to track down a bug found with a particular behavior of the software. Since this is a white box technique the source code is assumed to be available. The source code is necessary to do the control flow and data flow analysis. In practice, the slice is produced before program execution and during execution these statements are watched for bugs [Weiser 1984].

Some work has been done on slicing a program that uses separate compilation [Weiser 1984]. In this work it has been assumed that when a call is made to an external module (i.e., one for which you do not have the source available) it behaves as in the worst case. That is the external module is assumed to reference and change any external variable available to it [Weiser 1984].

Program slicing has also been applied as a software maintenance technique [Gallagher 1991]. Before a change is made it is applied to relevant slices to try to determine what effect the change will have on the behavior of the software. The authors claim this process can do away with regression testing as long as the changed slices produce the required behavior.

4.3.3.2 Fault Injection

Fault injection serves to determine how the software system will respond to faults arising during operation. These faults can include corrupted memory, failure of operating system subroutines or invalid input from a sensor, among others. The idea is to inject the faults that are possible and probable to encounter in the real world operation of the system. A fault tolerant system should be able to recover from some degree of these faults, with emphasis on some safety critical component not failing, i.e., medical devices, in a drastic way, i.e., harming the patient.

There are various ways to implement fault injection in software. Some common practices include wrapping the software in a container that muddles the output to other components, using debuggers to corrupt memory by overwriting portions of data or inserting extra instructions at compile time to corrupt the operation of the code.

With respect to COTS software, there is no chance to recompile the code, but certainly it can be wrapped and forced to produce bad outputs or given erroneous inputs. A good debugging tool could also be used to corrupt needed data, specifically argument lists after they have been passed to the software component. It would be difficult with COTS software and lacking the source code to corrupt specific data items.

4.3.4 Black Box

Black box testing techniques are characterized by focusing tests on the expected behavior of a software component and ensuring that the resulting functionality is correct [Beizer 1995]. In many cases this behavioral testing is done without the aid of source code or design documents. It is possible to provide input test cases and only test the outputs of the object for correctness. This type of black box testing does not require knowledge of the source code and so is possible to do with COTS components.

Strategies for black box testing that may be effective for COTS components includes test case generation by equivalence classes, error guessing and random tests. Equivalence classes minimize the number of tests necessary by representing the input space of the component by a few test cases that cover the entire set of possible inputs. Error guessing involves listing possible errors that could arise and generating test cases to ensure that these errors do not in practice happen [Beizer 1995]. Random testing involves randomly selecting input values from the entire input space of the component. It has been suggested that these random values should be chosen to reflect the expected operational input distribution [DeMillo, et al 1987]. They also should reflect previous experience with similar components and systems and the bugs found in them [Beizer 1995].

These three techniques rely only on some notion of the input space and expected functionality of the component being tested. With COTS components, the user should know these facts and be able to create test cases around them.

Black box testing is not strictly required to operate without access to the source code. Techniques such as control flow analysis and data flow analysis are black box techniques that require the access to requirements documents and source code [Beizer 1995]. They require that the internal behavior of the object be studied for correctness, making these techniques all but impossible to apply to COTS components.

To guide test case selection, the operational profile of the COTS component can be used. The operational profile identifies the criticality of components and their duration and frequency of use [Schneidewind 1998]. Both the operational profile of the COTS component and the whole system must be taken into account.

Table 6. Black Box Testing Techniques.

Technique	Description
Equivalence classes	Mark the input space into similar classes to allow on representative input to test for the entire class
Error guessing	Based on experience, test possible or common errors that are possible.
Random tests	Pick randomly from the input space for the test cases.
Operational profile testing	Choose test cases based on the expected operational environment for the product.

4.3.5 COTS Unit Testing Techniques

Unit testing techniques to gauge the functionality of a COTS component is not very different than judging the functionality of a non-COTS component. The user can, in both cases, define the input and output space and the correct mapping the component should provide. The COTS component can then be tested using the above mentioned techniques to ensure that the necessary functionality is present.

Testing a component for internal and external security and the handling of anomalous inputs is quite different between COTS and non-COTS components. The testing would be easier with access to the source code. Various inspection techniques are available to test for these items. Without the source code, as in the COTS case, the only recourse is to exercise the COTS product to the user's satisfaction that it is safe.

A proactive method to ensure the COTS product is to test the COTS product in isolation by feeding it inputs and watching the outputs produced. Of particular importance here is whether the correct outputs are produced, what happens when anomalous input is provided or malicious input is provided. [McGraw] describes a method to randomly generate intelligent (i.e., possible) data to feed to the COTS product during testing. They applied this to several Windows NT applications to determine their robustness. Their goal is to ensure that faulty input does not cause the COTS product to fail and to ensure that the output of the COTS product (if it does not fail) does not cause the rest of the system to fail.

Ballista is an automated tool that is used to test object interfaces for robustness [Kropp 1998]. It focuses on the "invalid inputs" referred to in the robustness definition. Ballista is designed to produce invalid inputs and watch the component for crashes or hangs as a result. A weakness here is that only crashes or hangs are considered failures. This methodology does not ensure that the outputs produced by the component are correct or that the functionality specified by the COTS vendor actually exits.

[Ghosh 1998] describes another method to stress test the COTS product with intelligent (syntactically correct) but unexpected input. In this case the input is guaranteed to be syntactically correct, i.e. of the correct data types, correct format of input strings, etc. This may not always be the case; corrupt data with unimaginable errors may be passed to the COTS product. They suggest that a wrapper is placed around the COTS product to filter the input to disallow explicitly incorrect inputs. Their method is also more robust in what types of errors it looks for from the COTS product. They check to see if any incorrect exit codes are raised, unhandled exceptions are thrown, the process hangs, insecure behavior (i.e., suspicious memory/disk access) or system crashes. Many other methods only focus on a hung process or system crashes, assuming the rest of the system will handle the error codes or exceptions. Their paper provides a case study of this method and note that currently work is being continued to extend the tool to handle COM/OLE/DCOM objects.

[Ghosh 1999] describes a method to test the robustness of a COTS product in the face of the failure of an Operating System call. They describe a tool that simulates the failure of various OS calls in a number of different ways. Some may count these call failures as an input failure, but no other paper explicitly mentions failures to OS calls or other function calls. One could conceive of extending this tool to handle simulating a failure in an arbitrary function call.
Unit testing needs to be able to ensure that no malicious code is hiding in the COTS product. Without the source code it is difficult to ensure test coverage of a component. However, [Kohl] suggests identifying dormant code in a COTS product to begin to determine if it include extra functionality or malicious code.

In a process oriented approach, Voas proposes a method to allow the vendor to attach a quality rating to a product, "Test Quality Rating." [Voas 1999] He proposes that an independent certification organization be allowed to do white-box testing on the component to determine how the product functioned and also how good the tests are expected to be at finding defects. Voas proposes using the Squeeze Play method to determine what types of test cases need to be run to ensure that the number of defects that has not been tested for is small [Voas 1999]. This method has at least one weakness; the vendor has to be willing to submit the product to independent testing. If the vendor will not allow the independent body to look at the source code of the product this is no better than any other black-box testing scheme. This is also not a proactive scheme from the point of view of the consumer of the COTS product. They could request or require the COTS vendor to provide this "Test Quality Rating" but if the industry refuses to comply, there is nothing the consumer can do.

Voas also stresses the need for testing the COTS product, not just looking at the software process. His metaphor is that a clean pipe (a quality software development/test process) can still produce dirty water (poor quality COTS software). [Voas 1999] He stresses that independent certification is necessary, many of the same arguments for IV&V of software products apply here. He notes that ISO or CMM (Capability Maturity Model) certification is not enough to ensure the COTS product is of high quality and meets the needs of the user.

Hissam details an interesting case study of defect finding in a COTS based system. The COTS components used in this project were loosely coupled and their functionality well specified. This may not always be the case with COTS components, but it does serve as good anecdotal evidence that is its possible to use black box testing to identify bugs within a COTS component. As the source code was not available for the COTS components the defect was defined in terms of high level functionality but in specific enough terms that the vendor was able to provide a patch.

Table 7. COTS Unit Testing Techniques.

Technique	***Goal***
Randomly generated intelligent input, valid and invalid input	Ensure robustness of COTS product to handle invalid input
Randomly generated intelligent input, syntactically correct, valid and invalid data values	Ensure robustness of COTS product to handle invalid input that is syntactically correct
Simulate Operating System call failures	Ensure robustness of COTS product to handle Operating System failures
Test Quality Rating	Allow the vendor to have an independent certification organization brand its product with a rating of quality
Identify dormant code	Ensure that no malicious code is packaged in the COTS product by ensuring that all the code has been executed/tested.

4.3.6 Integration Testing

Integration testing is done to ensure that component interfaces work together properly when the system is assembled [Hetzel 1984]. Functionality is not as important to test at this point as is interoperability of the components. Ideally unit testing has already assured the user that the functionality of the COTS component is consistent with the user's need. Several techniques can be used to achieve integration testing. The major difference in the techniques has to do with which components are integrated at what point in time. The common point is that components (or modules) are usually integrated one at a time, allowing the assumption that the last integrated component is responsible for any new failure. Bottom up integration testing takes components that have completed unit testing and integrates them together with other small modules. The main weakness of this technique is no preliminary version of the software product is available until the final module (the highest level module) is added to the system [DeMillo 1987]. Top down testing starts with the high level driver or top level modules and integrates increasingly lower level models until the smallest modules have been integrated. This method requires a good deal of stub or scaffolding code to be created to allow higher level modules to be run without the lower level modules being in place [DeMillo 1987]. Other combinations such as the most critical modules or a group of modules that perform some higher level functionality can be used to structure the integration testing [Hetzel 1984].

4.3.7 COTS Integration Testing

The goal of integration testing with COTS products is the same as with any software development process, the subsystems need to be tested to ensure that they work together correctly. In this case, the COTS products make this a bit more difficult for all the same reasons that unit testing of COTS products is difficult. Lack of source code and lack of clear understanding of its development/quality process make system testing a challenge. The unit testing should have found errors caused by the COTS product receiving erroneous input. The system testing needs to focus on how the system reacts to the output from the COTS product and general behavior of the COTS product. It is important to realize that integration testing does not ensure that the COTS product performs correctly but rather that the entire system works correctly with the addition of the COTS product. The COTS product may still fail or produce bad output, integration testing is done to ensure this does not cause the entire system to fail.

Voas describes a method, Interface Propagation Analysis, that predicts what impact a COTS product will have on system stability/performance [Voas 1998]. The most important object of study here is what happens if the COTS product fails. Voas suggests simulating the failure of a COTS product by isolating the COTS output and replacing it with erroneous output (random data or known bad outputs). If bad outputs from COTS are shown to cause system failures, the user needs to be able to show that these outputs are not possible from the COTS product. Voas mentions Static Fault tree analysis and Backward Static Slicing to do this. Unfortunately, both these methods require access to the COTS component source code, which is assumed to be unavailable. His solution to this is to wrap the COTS product in a software wrapper that filters its inputs and outputs to ensure that only good input goes to the COTS product and only good outputs are passed on to the rest of the system.

Bad output is not the only object of study. The user needs to ensure that correct/good output from the COTS product does not cause problems in the system. Does the COTS product have side effects? Does it corrupt other pieces of memory as it operates? These issues should be resolved during integration testing. This is also a bit involved with ensuring the custom software behaves as expected, i.e., does it really accept the inputs the COTS component was required to produce?

[Voas Charron] provides an assessment technique to determine the failure tolerance of interfaces between objects. This is a similar idea to [Voas] but focuses on how far the erroneous output propagates and whether it causes a system failure. This method as well as [Voas] does not guarantee that the COTS component will not fail or that it will produce correct output. It merely ensures that the entire system will not fail on the failure of a COTS product.

Assessing COTS products' reliability, maintainability, and availability (RMA) is considered in [Schneidewind 98].

The idea is that the decision to employ COTS on mission critical systems should not be based on development cost alone. Rather, costs should be evaluated on a total life cycle basis and RMA should be evaluated in a system context (i.e., COTS components embedded in a larger system).

Table 8. COTS Integration Testing Techniques.

Technique	***Goal***
Interface Propagation Analysis	Predict what impact a COTS product will have on a system's stability
Error Propagation	Determine how far bad input data will propagate through the component interfaces
Side Effects Check	Check that correct performance of the COTS component does not have side effect detrimental to the system

4.4 COTS and Highly Reliable Systems

The issues related to assuring the reliability of COTS-based systems were treated in section 4.3. Here we deal with techniques proposed to build highly reliable systems using COTS, and specifically with the Simplex approach [Sha 1998].

Some characteristics of COTS, such as the unavailability of source code and the possibility of inspecting and testing them white box, and the lack of control on processes used to build them, make COTS a sub-optimal choice for highly reliable systems.

It often is possible to obtain the source code of a COTS software component by paying a large sum of money to the vendor. With the source code, the customer can then subject the COTS to a high-assurance inspection and testing process and make any modifications that are needed. But once a COTS has been modified, it is no longer COTS software, the vendor is no longer in charge of maintenance, future releases could be not compatible, most if not all of the benefits of the COTS approach are lost.

The Simplex approach [Sha 1998] proposes a software fault-tolerant architecture to use COTS in highly reliable systems.
[Sha 1988] discusses reliability techniques such as replication, majority voting, and N-version programming.

Replication and majority voting were developed to deal with random hardware faults. However, they are ineffective against software faults.

N-version programming is an approach that is intended to randomize software errors and thus make majority voting work for software faults. Different programmers build different versions of the same software (or critical parts of a software system) with the idea that different designers and implementers will produce different errors. Therefore, when one system fails under a given set of circumstances, the other probably will not fail.

An adaptation of N-version programming for COTS is to use different vendors' COTS components with the same interface. For example, in the Boeing 777, three different vendors' Ada run-times and compilers are used [Yeh 1996]. However, some studies have indicated that some errors will still be shared among the independently developed systems [Knight 1986].
[Sha 1998] proposes to use the principle of analytic redundancy. Using analytic redundancy, a system is partitioned into a high-assurance portion and a high-performance portion. The high-assurance application kernel is designed to ensure simplicity and reliability, uses high assurance processes and excludes the use of COTS. On the other hand, COTS components can be used extensively in the high-performance subsystem. This model is applied in the Boeing 777: a high-assurance backup software controller implements the highly assured and simpler 747 control laws, whereas a high-performance 777 software controller serves as the normal digital controller [Yeh 1996].

The kernel monitors the system state. If unsafe, the kernel dynamically takes over. When a safe condition is in place again, the kernel switches back control to the high-performance subsystem.

This approach works well for systems that have states that can be monitored, such as feedback control and command-and-control applications.
Under the Simplex architecture, each major system function is implemented as an analytically redundant module consisting of a high-assurance application kernel and a high-performance subsystem, the components of which can be swapped in real time.

4.5 Project Management and Cost Estimation

Project management is probably the most impacted of horizontal activities (i.e. activities performed throughout the process). Project estimation and tracking both have to consider new activities. Estimating their duration is currently a complex task, due to the limited experience and estimation models existing. Project tracking is easier to accomplish, as it only requires modifications to the effort accounting procedures.

COCOTS (COnstructive COTS) [USC2000, Abst 2000], focuses on estimating the cost, effort, and schedule associated with using COTS in a software development project. Though still experimental, COCOTS is a model complementary to COCOMO II [USC 1997], capturing costs that traditionally have been outside the scope of COCOMO. Ideally, once fully formulated and validated, COCOTS will be used in concert with COCOMO to provide a complete software development cost estimation solution.

COCOTS considers four activities: assessment of candidate COTS, tailoring of COTS, development and testing of glueware, and increased system programming and testing. It is assumed that COCOMO models cover the traditional development, based on development and reuse of source code. Currently [Abst 2000] the COCOTS database contains data from 20 projects, too limited a data set to draw final conclusions.

5 A Suggested COTS Process

We present here (see Figure 3) the new proposed process for COTS-based projects developed by the SEL. 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. We discuss phases, activities and roles/responsibilities. We will concentrate on the differences and enhancements as compared with the current processes.
The main phases (dashed ovals) are: Requirements, Design, Coding and Integration. Most phases encompass activities specific to COTS-based development. These activities are drawn above the horizontal line in Figure 3. This line graphically separates the two tracks existing in COTS-based projects, traditional activities and COTS-specific activities.

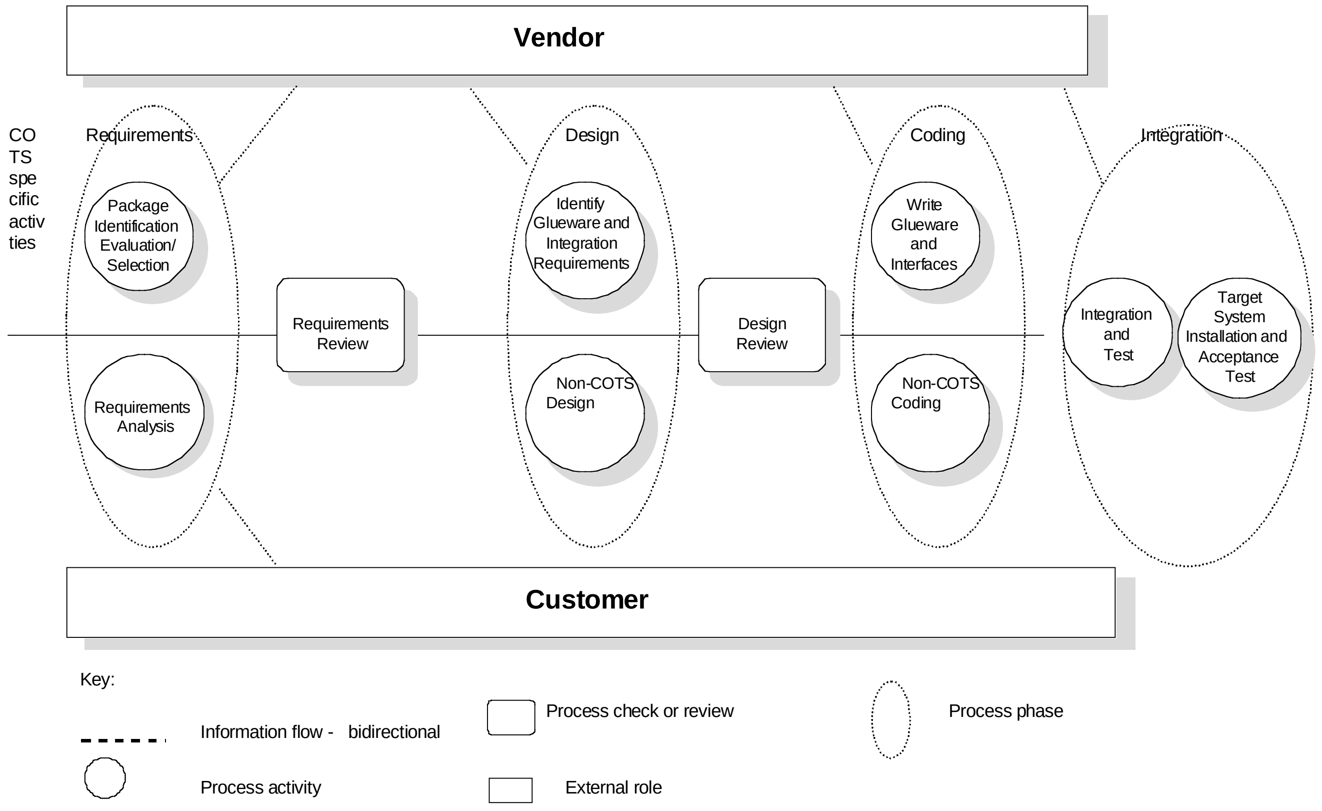


Figure 3. The New Proposed COTS Process.

5.1 Requirements

Previously, COTS selection was performed at the beginning of the Design phase. Now, based on findings from interviews with projects 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 iterated several times.

Make Vs. Buy Decision I. Using or not using COTS in a project is a key decision, which impacts all subsequent phases, and the success of the product. 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 are made outside the 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 occur 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 taken, considering only non-technical issues, the flexibility in requirements and the willingness to depend on the vendor. If either of these prerequisites is not satisfied, the project should not use COTS.

Requirements Definition. Requirements for the project are sketched out in little 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 take the second make vs. buy decision (see below). The description should consist of a complete requirement 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 and incompatibilities [Yakimovic 1999]. 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 could be [Kontio 1996, Morisio 1997]. The result of this decision is the product variant that will be developed, including requirements and COTS selected. Each variant represents a different trade-off among requirements satisfied, risks accepted and cost. This make vs. buy decision analyzes in detail these trade-offs.

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).

5.2 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, requesting a loop back to the Requirements phase.

5.3 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.

5.3.1 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.
- 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.

5.3.2 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 and 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 Technologies and Standards for COTS Integration

This section contains collections of online resources for the following COTS related technologies: CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model), and Java/RMI (Java/Remote Method Invocation); ActiveX and JavaBeans; Jini; JavaSpaces; Java Message Service (JMS); Enterprise JavaBeans (EJB); Extensible Markup Language (XML); DII COE (Defense Information Infrastructure Common Operating Environment); Unified Modeling Language (UML); and COnstructive COTS (COCOTS).

6.1 CORBA, DCOM, and Java/RMI

One architectural style for developing systems from COTS components is distributed object paradigms/component integration technologies/middleware remoting technologies. Three of the most popular technologies are CORBA, DCOM, and RMI.[1]

• CORBA is the Common Object Request Broker Architecture. It is a standardized specification developed by the Object Management Group (OMG). CORBA relies on a protocol called the Internet Inter-ORB Protocol (IIOP). CORBA is object-oriented middleware for heterogeneous distributed systems. It is language independent and supports multiple platforms. CORBA has language mappings to C, C++, Smalltalk, Cobol, Ada, and Java. CORBA provides a set of common interfaces through which object-oriented software can communicate, regardless of computer platform.

• DCOM is the Distributed Component Object Model developed by Microsoft. DCOM supports remoting objects by running on a protocol called the Object Remote Procedure Call (ORPC). DCOM is language independent and is not cross-platform compatible.

• Java/RMI is JavaSoft's Java/Remote Method Invocation. Java/RMI relies on a protocol called the Java Remote Method Protocol (JRMP). It supports Java implementations and is therefore language dependent. RMI enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts.[2] RMI is JavaSoft's implementation of a distributed object design. RMI provides a way for client and

[1] Gopalan Suresh Raj's Web Cornucopia — A Detailed Comparison of CORBA, DCOM, and Java/RMI. http://www.execpc.com/~gopalan/misc/compare.html
[2] The Source for Java Technology — java.sun.com — Java Remote Method Invocation (RMI). http://java.sun.com/products/jdk/rmi/

server applications to invoke methods across a distributed network of clients/servers running the Java Virtual Machine.[3]

6.1.1 Resources –– CORBA

http://www.execpc.com/~gopalan/misc/compare.html
A Detailed Comparison of CORBA, DCOM and Java/RMI — In this article, let us examine the differences between these three models from a programmer's standpoint and an architectural standpoint. At the end of this article, the reader will be able to better appreciate the merits and innards of each of the distributed object paradigms.

http://www.sei.cmu.edu/publications/documents/97.reports/97tr011/title.htm
A Study in the Use of CORBA in Real-Time Settings: Model Problems for the Manufacturing Domain —Technical report from the SEI (CMU/SEI-97-TR-011), by Andreas Polze, Daniel Plakosh, and Kurt Wallnau. This report describes the application of an off-the-shelf ORB to two real-time model problems. Based on the authors' experiences, they believe that today's ORBs can be used in real-time settings, with certain caveats as outlined in this report. They also outline the concept of composite objects, an approach for extending the range of non-real-time ORBs into a greater variety of real-time settings.

http://stsc.hill.af.mil/crosstalk/1997/feb/corba.asp
An Introduction to CORBA (CrossTalk, February 1997) — The Common Object Request Broker Architecture (CORBA) is an object-oriented infrastructure for distributed computing. CORBA enables software interoperability across multiple programming languages and platforms. CORBA is applicable to legacy integration, commercial-off-the-shelf integration, and new software development. This article describes the role and status of CORBA for Department of Defense (DoD) software development. CrossTalk is a publication of the Software Technology Support Center (STSC).

http://www.cetus-links.org:80/oo_corba.html
Cetus Links: Distributed Objects & Components (CORBA) — This site offers a variety of resources including demos and examples, central sites, link collections, tutorials, FAQs, references, standards, general newsgroups, mailing lists, general articles, special articles, interoperability bibliographies, books, organizations, projects, conferences/workshops, and products/companies.

[3] Earthweb — developer.com.
http://developer.earthweb.com/news/techfocus/022398_dist1.html

http://www.serve.com/mowbray/CDPflyer.html
CORBA Design Patterns — This book is written by Dr. Tom Mowbray (author of *The Essential CORBA*) and Raphael Malveau. This site includes CORBA Design Patterns for C++ using Orbix 2.0. ISBN 0-471-15882-8, An Object Management Group (OMG) book from John Wiley & Sons.

http://www.infosys.tuwien.ac.at/Research/Corba/OMG/arch2.htm#446864
CORBA Overview — This document provides an overview of COBRA taken from the 2.0 specification.

http://www.cs.wustl.edu/~schmidt/tutorials-corba.html
CORBA Tutorial — This page has links to electronic versions of tutorials on CORBA by Douglas C. Schmidt, schmidt@cs.wustl.edu, an Assistant Professor in the Department of Computer Science and the School of Engineering and Applied Science at Washington University. Introduction to Distributed Object Programming with CORBA: This tutorial provides an introduction to distributed object programming using the Common Object Request Broker Architecture. Implementing Concurrent CORBA Applications with Multi-Threaded Orbix and ACE: This tutorial illustrates how to use multi-threaded (MT) Orbix and ACE to implement various types of concurrent CORBA servers including thread-per request, thread-pool, and thread-per object. Measuring the Performance of Object-Oriented Components for High-speed Network Programming: This tutorial describes performance results from benchmarking several network programming mechanisms (C, ACE C++ wrappers, and two CORBA implementations — Orbix and ORBeline) over 155 Mbit/sec ATM and 10 Mbps Ethernet networks. An Overview of CORBA COSS Event Services: This tutorial explains the architecture of the CORBA Event Service and provides several complete examples.

http://cgi.omg.org/library/c2indx.html
CORBA/IIOP 2.1 Specification — The Object Management Group (OMG) provides electronic versions of the full and complete CORBA/IIOP 2.1 specifications in their formal, edited versions. This corresponds to OMG Technical Document formal/97-09-01.

http://www.objenv.com/cetus/oo_object_request_brokers.html
Cetus Links: CORBA Object Request Brokers (ORB) — Very comprehensive collection of links to CORBA-related tools in the following categories: ORBs / Vendors / Supported languages and features, CORBA Service Implementations, COM/CORBA Bridges, CORBA-aware CASE Tools, CORBA for Messaging and Fault Tolerance, Miscellaneous CORBA Products and Tools, and other lists of CORBA software.

http://www.mitre.org/research/domis/omg/orb.html

Common Object Request Broker Architecture (CORBA) Specifications — This site contains the CORBA specifications.

Comp-Object-CORBA — Send mail to majordomo@omg.org with the following command in the body of the message: subscribe comp-object-corba.

http://corbaweb.lifl.fr/
CorbaWeb — CorbaWeb is a generic gateway between the Common Object Request Broker Architecture (CORBA) and the World Wide Web (WWW). This site provides information on the GOODE Project, CorbaScript language, a list of publications, demonstrations, and links to related resources.

http://dii-sw.ncr.disa.mil/coe/topics/atd/
Defense Information Systems Agency's (DISA) Advanced Technologies — Links to briefings on topics in OOT, CORBA, and Java, including applications to Ada and reengineering.

http://www.itsi.disa.mil/
Defense Information Systems Agency (DISA)/Joint Interoperability and Engineering Organization (JIEO) Center for Standards — This site supports DISA/JIEO's mission to orchestrate, as DoD's Executive Agent for centralized management of information technology standards, the development, adoption, specification, certification and enforcement of information processing, transfer and content standards within DoD. This includes influencing the development and adoption by industry of standards supporting DoD requirements. A document library is provided which serves as a central on-line source for the policies, guidance, standards, and other references related to building affordable, maintainable, and interoperable standards-based information systems for the DOD.

http://www.sei.cmu.edu/publications/documents/97.reports/97tr004/97tr004chap01.htm
Distributed Object Technology with CORBA and Java: Key Concepts and Implications —The purpose of this Software Engineering Institute (SEI) report is to analyze the potential impact of distributed object technology (DOT) on software engineering practice. The analysis culminates with the conclusion that the technology will have a significant influence on both the design and reengineering of information systems and the processes used to build them. The authors see a profound impact and fundamental change in both technical thinking and practice as a result of the related technologies they group together as DOT.

http://www.flashline.com/

Flashline.com -- The Software Component Marketplace This marketplace is available to research, buy, and sell Java, COM, and CORBA software components. JavaBeans, Enterprise JavaBeans (EJB), and custom components are also available.

JavaCORBA Mailing List — This is a place to discuss various issues of using Java and CORBA together. To subscribe, send a mail to listserv@luke.org with SUBSCRIBE or SUBSCRIBE DIGEST in the Subject field.

http://www.javacoffeebreak.com/articles/rmi_corba/
Java RMI & CORBA: A Comparison of Two Competing Technologies — With the introduction of CORBA support to Java (as of version 1.2), developers now face the question of whether to continue to use remote method invocation (RMI), or make a move to CORBA. This article discusses the pros and cons, and evaluates the potential of these two technologies.

http://cgi.omg.org/corba/beginners.html
OMG's CORBA for Beginners — The Object Management Group (OMG) provides the resources at this site. Here you will find a variety of information about CORBA including: overviews and tutorials; books and magazine articles; CORBA mailing lists; CORBA demos and programming examples, links to other CORBA-related sites, and a CORBA FAQ.

http://www.omg.org/
Object Management Group (OMG) — OMG is a non-profit consortium dedicated to promoting the theory and practice of object technology (OT) for the development of distributed computing systems. OMG was formed to help reduce the complexity, lower the costs, and hasten the introduction of new software applications. Their goal is to provide a common architectural framework for object-oriented applications based on widely available interface specifications. Their international membership currently stands at over 600 software vendors, software developers and end users.

http://www.serve.com/mowbray/essential.html
The Essential CORBA Systems Integration Using Distributed Objects —This book was written by Dr. Thomas J. Mowbray (author of *CORBA Design Patterns*) and Ron Zahavi.
ISBN 0-471-10611-9, published by John Wiley & Sons.

http://www.webadvisor.com/corba.html
Web Advisor: CORBA — CORBA, or Common Object Request Broker Architecture, is a set of specifications defining the ways software objects should work together in a distributed environment. The organization which drives the specifications, OMG, or Object Management Group, has over hundreds of members representing a major portion of the software industry. The members work together to propose, review, and finally adopt the set of specifications to

allow software objects to be developed independently and yet work together in a harmonic fashion. The fundamental piece of CORBA is the ORB, or Object Request Broker. The ORB can be viewed like a bus carrying the 'objects' between the clients, those that consume the objects, and the servers, those that produce the objects. The consumers are provided with 'object interfaces' defined using a language called the Interface Definition Language. The detail implementation of the objects by the producers is totally shielded from the consumers. The promise of the inter-working of software objects from different vendors through CORBA has induced major players in the industry to aggressively endorse OMG's drive to forge ahead its agenda. This page contains a list of Useful Links.

6.1.2 Resources –– DCOM

http://www.execpc.com/~gopalan/misc/compare.html
A Detailed Comparison of CORBA, DCOM and Java/RMI — In this article, let us examine the differences between these three models from a programmer's standpoint and an architectural standpoint. At the end of this article, the reader will be able to better appreciate the merits and innards of each of the distributed object paradigms.

http://www.cetus-links.org/oo_ole.html
Cetus Links -- Distributed Objects & Components: COM/DCOM — This information rich source provides related sites, link collections, tutorials, FAQs, glossaries, newsgroups, books, magazines, organizations, conferences/workshops, and utilities/tools.

http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomarch.htm
DCOM Architecture by Markus Horstmann and Mary Kirtland — This article focuses on the inner workings of DCOM. It targets the application developer who wishes to create "state of the art" applications, which scale equally well on the intranet, on the Internet, and beyond.

http://www3.sagasoftware.com/cgi-bin/framer.exe?type=full&addr=http://www3.sagasoftware.com/site/solution/som/entirex/dcom_gl.htm
DCOM Glossary — This site offers short definitions of terms used when working with DCOM.

http://www.dalmatian.com/dcom.htm
DCOM Overview — The Distributed Component Object Model (DCOM) is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner.

http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomtec.htm

DCOM Technical Overview — Microsoft® Distributed COM (DCOM) extends the Component Object Model (COM) to support communication among objects on different computers — on a LAN, a WAN, or even the Internet. This white paper provides a high-level overview of how you can use DCOM to solve the hardest problems associated with distributed applications.

http://www1.bell-labs.com/user/emerald/dcom_corba/Paper.html
DCOM and CORBA Side by Side, Step by Step, and Layer by Layer — DCOM (Distributed Component Object Model) and CORBA (Common Object Request Broker Architecture) are two popular distributed object models. In this paper, the authors make architectural comparison of DCOM and CORBA at three different layers: basic programming architecture, remoting architecture, and the wire protocol architecture. A step-by-step description of remote object activation and method invocation is provided to demonstrate the similarities and differences of the two frameworks. A primary goal is for people who are already familiar with one model to quickly understand the basic architecture of the other.

http://www.ttginc.com/dcomarticle.htm
DCOM — A Technical and Business Overview — The Distributed Component Object Model protocol is an application-level protocol for object-oriented remote procedure calls useful for distributed, component-based systems of all types.

http://www.microsoft.com/com/tech/DCOM.asp
Microsoft COM Technologies — This site contains articles in the press, white papers, case studies, useful downloads, samples, specifications, books, and links to related sites.

http://www.sei.cmu.edu/str/descriptions/com.html
Software Technology Review — Component Object Model (COM), DCOM, and Related Capabilities — Distributed COM (DCOM) is an extension to COM that allows network-based component interaction. While COM processes can run on the same machine but in different address spaces, the DCOM extension allows processes to be spread across a network. With DCOM, components operating on a variety of platforms can interact, as long as DCOM is available within the environment.

http://www.microsoft.com/ntserver/zipdocs/dcom_architecture.exe
Understanding the Distributed Object Component Model (DCOM) Architecture — DCOM extends the Component Object Model (COM) to support communication among objects on different computers -- whether on a local area network (LAN), a wide area network (WAN), or the Internet.

6.1.3 Books –– DCOM

http://www.smartbooks.com/bw712comdcom.htm
COM and DCOM: Microsoft's Vision for Distributed Objects — by Roger Sessions; December 1997; Wiley Computer Publishing; ISBN: 0-471-19381-X

http://www.amazon.com/exec/obidos/ASIN/0672313529/qid=963855390/sr=1-5/002-4377486-0324804
COM/DCOM Unleashed — by Randy Abernethy, Jesus Chahin, Randy Charles Morin; April 1999; MacMillan Publishing Company; ISBN: 0672313529

http://www.amazon.com/exec/obidos/ASIN/1555582168/qid=963855390/sr=1-2/002-4377486-0324804
DCOM Explained — by Rosemary Rock-Evans August 1998; Digital Press; ISBN: 1555582168

http://www.amazon.com/exec/obidos/ASIN/157231849X/qid=963855390/sr=1-4/002-4377486-0324804
Inside Distributed COM — by Guy Eddon, Henry Eddon April 1998; Microsoft Press; ISBN: 157231849X

http://www.amazon.com/exec/obidos/ASIN/1565925815/qid=963855390/sr=1-1/002-4377486-0324804
Learning DCOM — by Thuuan L. Thai, Andy Oram (Editor) April 1999; O'Reilly & Associates; ISBN: 1565925815

http://www.amazon.com/exec/obidos/ASIN/186100060X/qid=963855390/sr=1-6/002-4377486-0324804
Professional DCOM Programming — by Richard Grimes June 1997; Wrox Press Inc.; ISBN: 186100060X

6.1.4 Resources –– Java/RMI

http://www.amazon.com/exec/obidos/ASIN/0764580434/qid%3D963843914/002-4377486-0324804
Java™ RMI: Remote Method Invocation — by Troy Bryan Downing February 2, 1998; IDG Books Worldwide; ISBN: 0764580434

http://www.execpc.com/~gopalan/misc/compare.html
A Detailed Comparison of CORBA, DCOM and Java/RMI — In this article, let us examine the differences between these three models from a programmer's standpoint and an architectural standpoint. At the end of this article, the reader will be able to better appreciate the merits and innards of each of the distributed object paradigms.

http://www.cetus-links.org/oo_java_rmi.html
Cetus Links -- Distributed Objects & Components: Java RMI — This information rich resource includes examples, demos, FAQ, related sites, bibliographies, books, and software tools.

http://java.sun.com/products/jdk/1.2/docs/guide/rmi/getstart.doc.html
Getting Started Using RMI: a Tutorial — This tutorial shows you the steps to follow to create a distributed version of the classic Hello World program using Java Remote Method Invocation (RMI).

http://www.enteract.com/~manish/howtormi.html
How to Develop a Java Application Using RMI — This tutorial contains a brief explanation of the process of developing an RMI application in Java. This page is focused on laying out all of the details, in sequence, concisely, so that readers do not lose track of them.

http://www.javacoffeebreak.com/articles/rmi_corba/
Java RMI & CORBA: A Comparison of Two Competing Technologies — With the introduction of CORBA support to Java (as of version 1.2), developers now face the question of whether to continue to use remote method invocation (RMI), or make a move to CORBA. This article discusses the pros and cons, and evaluates the potential of these two technologies.

http://www.javasoft.com/marketing/collateral/javarmi.html
Java Remote Method Invocation: Distributed Computing for Java — This paper describes the benefits of RMI, and how you can connect it to existing and legacy systems as well as to components written in Java.

http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html
Java Remote Method Invocation Specification — The Java RMI specification is available here.

http://www.ccs.neu.edu/home/kenb/com3337/rmi_tut.html
Northeastern University's RMI Tutorial — This tutorial attempts to show the essence of RMI, without discussing any extraneous features.

http://java.sun.com/products/jdk/rmi/
Sun's Java Remote Method Invocation (RMI) — This site provides documentation, the RMI specification, a white paper, and a data sheet.

http://java.sun.com/products/jdk/1.1/docs/guide/rmi/index.html
Sun's Remote Method Invocation (RMI) — This site provides a tutorial, examples, the RMI Specification, RMI API Reference, RMI tools, and release notes.

6.2 Component Models –– ActiveX and JavaBeans

JavaBeans and ActiveX both serve the same basic function: to facilitate communication among software components within framework "containers." These containers include Web browsers and other document viewers. Like JavaBeans, ActiveX serves as a component framework; the two are direct competitors.[4]

A loosely defined set of technologies developed by Microsoft, ActiveX is an outgrowth of two other Microsoft technologies called OLE (Object Linking and Embedding) and COM (Component Object Model). This set of technologies from Microsoft provides tools for linking desktop applications to the World Wide Web. Using a variety of programming tools — including Java, Visual Basic, and C+ allow users to view Word and Excel documents directly in a browser.[5] ActiveX is a standard that enables software components to interact with each other in a networked environment, regardless of the language in which they were created.[6]

JavaBeans is a portable, platform-independent component model written in the Java programming language. It enables developers to write reusable components once and run them anywhere. JavaBeans acts as a bridge between proprietary component models and provides a seamless means for developers to build components that run in ActiveX container applications. JavaBeans components, or Beans, are reusable software components that can be manipulated visually in a builder tool. Beans can be combined to create traditional applications, or their smaller web-oriented brethren, applets. In addition, applets can be designed to work as reusable Beans.[7]

6.2.1 Resources –– ActiveX

http://www.active-x.com/
Active-X.com — This site provides components for downloading.

http://www.microsoft.com/windows/ie/support/docs/tech30/activex.htm
ActiveX Technology and ActiveX Controls — ActiveX is a standard that enables

[4] Java World — JavaBeans vs. ActiveX: Strategic Analysis. http://www.javaworld.com/javaworld/jw-02-1997/jw-02-activex-beans.html
[5] CNET Glossry. http://coverage.cnet.com/Resources/Info/Glossary/Terms/activex.html/
[6] Microsoft Internet Explorer — Technical White Papers. http://www.microsoft.com/windows/ie/support/docs/tech30/activex.htm
[7] The Source for Java Technology — java.sun.com; JavaBeans FAQ: General Questions. http://java.sun.com/beans/faq/faq.general.html#Q1

software components to interact with each other in a networked environment, regardless of the language in which they were created.

http://www.shorrock.u-net.com/axintro.html
ActiveX Unofficial Guide — This guide takes the reader through all aspects of ActiveX control creation starting with Visual Basic ActiveX creation, progressing to Java and Visual C++ methods of creation. It contains an introduction to using Visual Basic Scripting with ActiveX controls, a starter on some of the ActiveX OLE controls that are built into Windows, a couple of "create your own" ActiveX Control Examples, the Scripting Model in Internet Explorer/Netscape and a guide to the best ActiveX sites on the web.

http://browserwatch.internet.com/activex/activex-big.html
Browser Watch: ActiveX Arena! — This site provides a collection of downloadable ActiveX controls.

http://www.javaworld.com/javaworld/jw-02-1997/jw-02-activex-beans.html
JavaBeans vs. ActiveX: Strategic Analysis — In this article the author presents a technical overview and investigates the philosophies, marketing strategies, and agendas of JavaSoft and Microsoft, cutting through the hype and fervor to provide an objective look at their component models.

http://www.weblearningcenter.com/scripting/activex.htm
Web Learning Center: ActiveX — This site contains an open forum where anyone can submit or answer questions about ActiveX programming or using ActiveX controls; ActiveX examples; and links to resources for Web scripting and programming.

http://www.whatis.com/activexc.htm
What is an ActiveX control? — An ActiveX control is a component program object that can be re-used by many application programs within a computer or among computers in a network.

6.2.2 Books –– ActiveX

http://www.amazon.com/exec/obidos/ASIN/1558515038/qid=964114986/sr=1-2/002-4377486-0324804
Designing and Using ActiveX Controls — by Tom Armstrong December 30, 1996; IDG Books Worldwide; ISBN: 1558515038

http://www.amazon.com/exec/obidos/ASIN/0079132286/qid=964114986/sr=1-13/002-4377486-0324804
Active Xpert — by Tom Armstrong, Jim Crespino, Rob Alumbaugh September 1997; Computing McGraw-Hill; ISBN: 0079132286

http://www.amazon.com/exec/obidos/ASIN/1576760162/qid=964114986/sr=1-20/002-4377486-0324804
ActiveX and the Internet — by Forest Lin, Richard Jones April 1998; Scott/Jones; ISBN: 1576760162

http://www.amazon.com/exec/obidos/ASIN/0764531506/qid=964114986/sr=1-22/002-4377486-0324804
Discover ActiveX — by Richard Mansfield October 30, 1997; IDG Books Worldwide; ISBN: 0764531506

http://www.amazon.com/exec/obidos/ASIN/0201485362/qid=964114986/sr=1-16/002-4377486-0324804
Mr. Bunny's Guide to ActiveX — by Carlton Egremont, III July 1998; Addison-Wesley Publishing Co.; ISBN: 0201485362

http://www.amazon.com/exec/obidos/ASIN/1572312165/qid=964114986/sr=1-1/002-4377486-0324804
Understanding ActiveX and OLE — by David Chappell September 1996; Microsoft Press; ISBN: 1572312165

6.2.3 Resources –– JavaBeans

http://www.cetus-links.org/oo_javabeans.html
Cetus Links — Distributed Objects and Components: JavaBeans and Enterprise JavaBeans — This site offers a wealth of information on JavaBeans and Enterprise JavaBeans. Topics include examples, demos, central sites, related sites, tutorials, FAQs, standards, literature, organizations, conferences/workshops, and development environments.

http://www.flashline.com/
Flashline.com: The Software Component Marketplace — This marketplace is available to research, buy, and sell Java, COM, and CORBA software components. JavaBeans, Enterprise JavaBeans (EJB), and custom components are also available.

http://www.javaworld.com/javaworld/jw-02-1997/jw-02-activex-beans.html
JavaBeans vs. ActiveX: Strategic Analysis — In this article the author presents a technical overview and investigates the philosophies, marketing strategies, and agendas of JavaSoft and Microsoft, cutting through the hype and fervor to provide an objective look at their component models.

http://www.vb-bookmark.com/JavaBeans.html
Java Bookmark: JavaBeans Information Directory — This site offers articles, class libraries, development tools, FAQ, source code, and tutorials.

http://developer.java.sun.com/developer/
Java Developer Connection — This site offers tutorials, technical articles, technical tips, code samples, and related resources.

http://java.sun.com/products/jdk/1.1/index.html
Java Development Kit (JDK) — The Java Development Kit (JDK) contains the software and tools that developers need to compile, debug, and run applets and applications written using the Java programming language. The JDK software and documentation is free per the license agreement.

http://splash.javasoft.com/beans/software/bdk_download.html
JavaBeans Development Kit (BDK) Download — The BDK is intended to support the development of JavaBeans components and to act as a standard reference base for both component developers and tool vendors. The BDK provides a reference Bean container, the BeanBox and a variety of reusable example source code for use by both tool and beans developers.
http://www.ibm.com/java/education/jb-guidelines.html
JavaBeans Guidelines — This document presents supplementary guidelines that enable a user to develop good Beans that are well-behaved in the greatest number of environments, including the popular IDEs and browsers.

http://www.javasoft.com/beans/index.html
JavaBeans Home Page: Javasoft — This site contains software, documentation, FAQ, development tools, training and support, and information on marketing beans.

http://www.javasoft.com/beans/spec.html
JavaBeans Specification — The JavaBeans 1.01 specification describes JavaBeans as present in JDK 1.1.

http://www.javabeans-zone.com/
JavaBeans Zone — This site contains an FAQ, a collection of articles, an *Ask the Pro* service, and access to discussion groups.

http://www.javashareware.com/
JavaShareware.com — This site contains applications, applets, JavaBeans, development tools, and links to related sites.

http://www.ccs.neu.edu/home/lorenz/JavaBeans/oo_javabeans.html
Links JavaBeans — This rich collection of resources includes links to examples, demos, related sites, tutorials, FAQ, support, references, standards, newsgroups, mailing lists, articles, bibliographies, books, magazines, organizations, conferences, tools, and components.

http://java.sun.com/beans/FAQ.html
Sun's Java Beans FAQ — This collection of FAQ answers common questions and provides help with common problems.

http://java.sun.com/beans/
Sun's JavaBeans Home Page — This site offers software, documentation, FAQ, development tools, training and support, events, and a marketplace for beans.

http://www.alphaworks.ibm.com/alphabeans
alphaBeans: JavaBeans by IBM — This site makes available a collection of JavaBean components developed by IBM.

6.2.4 Books –– JavaBeans

http://www.amazon.com/exec/obidos/ASIN/1562057162/qid%3D962041588/sr%3D1-16/002-4377486-0324804
JavaBeans Developer's Reference — by Dan Brookshier, Ramesh Santanam; March 1997; New Riders Publishing; ISBN: 1562057162

http://www.amazon.com/exec/obidos/ASIN/0782120970/qid%3D962041588/sr%3D1-14/002-4377486-0324804
Mastering JavaBeans — by Laurence Vanhelsuwe; May 1997; Sybex Inc; ISBN: 0782120970

http://www.amazon.com/exec/obidos/ASIN/1575213168/qid%3D962041588/sr%3D1-19/002-4377486-0324804
Teach Yourself Javabeans in 21 Days — by Don Doherty; August 1997; Sams; ISBN: 1575213168

http://www.amazon.com/exec/obidos/ASIN/1852330325/qid%3D962041588/sr%3D1-29/002-4377486-0324804
Essential JavaBeans Fast — by John Hunt; October 1998; Springer Verlag; ISBN: 1852330325

http://www.amazon.com/exec/obidos/ASIN/007882477X/qid%3D962041588/sr%3D1-5/002-4377486-0324804
JavaBeans Programming from the Ground Up — by Joseph O'Neil, Herbert Schildt (Editor); April 1998; Osborne McGraw-Hill; ISBN: 007882477

http://www.amazon.com/exec/obidos/ASIN/067231424X/qid%3D962041588/sr%3D1-32/002-4377486-0324804
JavaBeans Unleashed — by Don Doherty, Rick Leinecker; December 22, 1999; Sams; ISBN: 067231424X

http://www.amazon.com/exec/obidos/ASIN/0137903383/qid%3D962041588/sr%3D1-33/002-4377486-0324804
JavaBeans by Example — by Henri Jubin, Jalapeno Team; January 1998; Prentice Hall Computer Books; ISBN: 0137903383

http://www.amazon.com/exec/obidos/ASIN/0079137040/qid%3D962041588/sr%3D1-26/002-4377486-0324804
Programming JavaBeans 1.1 : Hands-On Web Development — by Reaz Hoque, Tarun Sharma; May 1998; Computing McGraw-Hill; ISBN: 0079137040

http://www.javaworld.com/javaworld/jw-05-1998/jw-05-beans.html
JavaBeans Book Review: JavaWorld — This site presents a book review of three titles. In addition, links to several other books and related resources are included on the site.

6.2.5 Educational Materials –– JavaBeans

http://developer.java.sun.com/developer/onlineTraining/Beans/JBShortCourse/beans.html
JavaBeans Short Course: Introduction to JavaBeans — This tutorial covers the following: the JavaBeans architecture, the Beans Event Model, Introspection to query Beans about their contents, Bean component creation, Customization of Beans, Persistence to store and retrieve Beans, the development of applications comprised of Bean components, the creation of simple builder application, and the BDK BeanBox application.

http://www.cse.iitb.ernet.in/stuff/internet/java/javabeans/
JavaBeans Tutorial — This tutorial defines a Bean and Bean concepts, describes the JavaBeans Development Kit (BDK) contents and the demonstration Beans, and discusses future Bean directions. It describes the BeanBox and writing both simple and advanced Beans.

http://java.sun.com/docs/books/tutorial/javabeans/index.html
JavaBeans Tutorial by Andy Quinn — This tutorial covers JavaBeans Concepts and the Beans Development Kit (BDK), Using the BeanBox, Writing a Simple Bean, Properties, Manipulating Events in the BeanBox , the BeanInfo Interface, Bean Customization, Bean Persistence, using the BeanContext API, and New Features.

6.3 Jini

A Jini system is a Java technology-centered, distributed system designed for simplicity, flexibility, and federation. The Jini architecture provides mechanisms for machines or programs to enter into a federation where each machine or program offers resources to other members of the federation and uses resources as needed. The design of the Jini architecture exploits the ability to move Java programming language code from machine to machine and unifies, under the notion of a service, everything from the user of a Jini system to the software available on the machines to the hardware components of the machines themselves.[8]

6.3.1 Resources –– Jini

http://www.devdaily.com/Dir/Java/Articles_and_Tutorials/Jini/
Developers Daily Jini Tutorials — This site provides a collection of Jini tutorials.

http://www.litefaden.com/sv/jd/
Directory of Jini Resources — This site provides a collection of articles, reviews, books, tutorials, examples, FAQ, Jini documentation, and related sites.

http://www.jdance.com/jini.shtm
JDance — This site presents a collection of Jini related articles.

http://pandonia.canberra.edu.au/java/jini/tutorial/Jini.xml
Jan Newmarch's Guide to Jini Technologies — This tutorial provides an overview of Jini, troubleshooting tips, and extensive coverage of related topics.

http://www.javaboutique.internet.com/jini/
Java Boutique's Jini Watch — This site provides news articles on Jini. In addition, the site provides access to a discussion group and related Jini resources.

http://triton.cc.gatech.edu/ubicomp/224
Jini Architecture — This site contains the following specifications: Jini Distributed Event Specification, Jini Distributed Leasing Specification, Jini Lookup Service Specification, Jini Lookup Attribute Schema Specification, Jini Discovery and Join Specification, and Jini Device Architecture Specification.

[8] Jini Architecture Specification, Revision 1.0, January 25, 1999, cover

http://www.artima.com/jini/index.html
Jini Corner at Artima.com: Resources for Jini Developers — At this site visitors will find a Jini FAQ, a comprehensive set of links to Jini resources on the web, a discussion forum, an interface repository, and more for the Jini developer.

http://www.sun.com/jini/specs/
Jini Specifications — The complete set of Jini specifications is available for download from this site.

http://www-rohan.sdsu.edu/doc/jini/doc/api/overview-summary.html
Jini Technology 1.0 API Documentation — This site provides an overview of Jini technology API documentation.

http://www-rohan.sdsu.edu/doc/jini/
Jini Technology Index — This site provides release notes and specifications for the Jini Technology Core Platform (JCP), Jini Technology Extended Platform (JXP), Jini Software Kit (JSK), and JavaSpaces Technology Kit (JSTK) Eval.

http://developer.java.sun.com/developer/products/jini/
Jini Technology Web Site — The purpose of this site is to provide visitors with a central location for accessing and downloading Jini technology infrastructure software from Sun Microsystems, including the Jini Technology Starter Kit, Jini specifications, and the Jini Technology Core Platform Compatibility Kit (TCK).

http://triton.cc.gatech.edu/ubicomp/429
Jini Tutorial from Georgia Tech — This tutorial provides an introduction to Jini, five key concepts to understanding Jini, and examples and resources.

http://www.eli.sdsu.edu/courses/spring99/cs696/notes/index.html
Lecture Notes: Emerging Technologies: Java Distributed Computing — This page contains links to lecture notes for the
CS 696 Emerging Technologies: Java Distributed Computing course.

http://www.enete.com/download/#_nuggets_
Noel Enete's Nuggets — This site provides example Jini code.

http://www.sun.com/jini/community/
Sun Jini Community — A key concept behind Jini™ connection technology (Jini technology) is the community. This site provides the Sun community program overview, the Jini technology licensee community, a downloadable copy of the Jini Technology Starter Kit, white papers and other documentation, a demo, specifications, books, and other related resources.

http://developer.java.sun.com/developer/products/jini/product.offerings.html
Sun Microsystems Jini System Software 1.0 Product Offerings — There are three Jini System Software product offerings: the Jini System Software Starter Kit (Jini Starter Kit), the Jini Technology Core Platform Compatibility Kit (TCK), and the JavaSpaces Technology Kit (JSTK). This site provides both software and documentation downloads.

http://www.sun.com/jini/subscribe.html
Sun's Jini Mailing List — Get the latest Jini™ technology-related news by subscribing to the Jini technology mailing list. Subscribers will receive announcements that include the following: software specifications, downloadable software, developer support programs, and other relevant information.

http://www.sun.com/jini/overview/index.html
Sun's Jini Overview — This site provides an introduction to Jini technology, FAQs, and demonstrations.

http://wwwjini.org/
The Jini Community — This web site is a focal point for the growing community that is exploring, developing, and evolving Jini connection technology. It is a place to share not just ideas, but related source code, documentation and other development work based on Jini technology. The site facilitates collaborative development by providing discussion forums, newsgroups, chats, mailing lists, and anything else that brings the community together.

http://java.sun.com/docs/books/jini/
The Jini Technology Series — This site provides descriptions of Jini books. In addition, it contains example code, errata, and information on forthcoming books.

http://webopedia.internet.com/TERM/J/Jini.html
Webopedia's Definition of Jini — Software from Sun Microsystems that seeks to simplify the connection and sharing of devices, such as printers and disk drives, on a network.

6.3.2 Books –– Jini

http://www.amazon.com/exec/obidos/ASIN/013014469X/o/qid%3D961431788/sr%3D2-1/002-4377486-0324804
Core Jini — by W. Keith Edwards; June 25, 1999; Prentice Hall; ISBN: 013014469X

http://www.amazon.com/exec/obidos/ASIN/0130863866/qid%3D961431788/sr%3D1-2/002-4377486-0324804
Core Jini -- The Complete Video Course — by W. Keith Edwards; January 2000; Prentice Hall; ISBN: 0130863866

http://www.amazon.com/exec/obidos/ASIN/1565927591/o/qid%3D961431788/sr%3D2-2/002-4377486-0324804
Jini in a Nutshell: A Desktop Quick Reference — by Scott Oaks, Henry Wong; March 2000; O'Reilly & Associates; ISBN: 1565927591

http://www.amazon.com/exec/obidos/ASIN/0764545914/qid%3D961431788/sr%3D1-11/002-4377486-0324804
Jini, a Primer — by Jamie Jaworski, et. al. August 2000; IDG Books Worldwide; ISBN: 0764545914

http://www.amazon.com/exec/obidos/ASIN/1861002777/qid%3D960497042/sr%3D1-4/002-4377486-0324804
Professional Java Server Programming: with Servlets, Java Server Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and JavaSpaces — by Andrew Patzer, et. al. August 1999; Wrox Press Inc; ISBN: 1861002777

http://www.amazon.com/exec/obidos/ASIN/1861003552/qid%3D961431788/sr%3D1-12/002-4377486-0324804
Professional Jini & JavaSpaces Programming — by Sing Li, Alvin Chin; June 2000; Wrox Press Inc; ISBN: 1861003552

http://www.amazon.com/exec/obidos/ASIN/0201616343/o/qid%3D961431788/sr%3D2-3/002-4377486-0324804
The Jini Specification (The Jini Technology Series) — by Ken Arnold, et. al. June 1999; Addison-Wesley Publishing Co.; ISBN: 0201616343

http://www.aw.com/cseng/javaseries/index.shtml#jini
Addison-Wesley Jini Technology Series — From the creators of the technology at Sun Microsystems comes the official series for reference material and programming guides on Jini technology. Written by those who design, implement, and document the technology, these books show you how to use, deploy, and create applications using the Jini technology.

http://www.artima.com/jini/booklist.html
Artima's List of Recommended Books about Jini and JavaSpaces — This site provides a comprehensive list of books about Jini and JavaSpaces.

6.3.3 Frequently Asked Questions (FAQ) –– Jini

http://www.sun.com/jini/faqs/index.html
Sun's Jini Technology Frequently Asked Questions (FAQ) — This site provides a general overview of Jini, details about Jini technology, information on developing with Jini technology, and Jini and Java technologies.

http://www.sun.com/jini/faqs/
Jini Technology Frequently Asked Questions — This is Sun Microsystems's FAQ site for Jini.

6.4 JavaSpaces

The JavaSpaces technology package provides a distributed persistence and object exchange mechanism for code written in the Java programming language. Objects are written in entries that provide a typed grouping of relevant files. Clients can perform simple operations on a JavaSpaces server to write new entries, lookup existing entries, and remove entries from the space. Using these tools, you can write systems to store state, and also write systems that use flow of data to implement distributed algorithms and let the JavaSpaces system implement distributed persistence for you.[9]

JavaSpaces technology is a simple unified mechanism for dynamic communication, coordination, and sharing of objects between Java technology-based network resources like clients and servers. In a distributed application, JavaSpaces technology acts as a virtual space between providers and requesters of network resources or objects. This allows participants in a distributed solution to exchange tasks, requests and information in the form of Java technology-based objects. JavaSpaces technology provides developers with the ability to create and store objects with persistence, which allows for process integrity.[10]

This technology provides a cooperative marketplace for posting and retrieving groups of related objects across a network. "Buyers" and "Sellers" dynamically post their requests and/or services to a space and receive a response from interested parties.[11]

6.4.1 Resources –– JavaSpaces

http://www-rohan.sdsu.edu/doc/jini/
Jini Technology Index — This site provides release notes and specifications for the Jini Technology Core Platform (JCP), Jini Technology Extended Platform (JXP), Jini Software Kit (JSK), and JavaSpaces Technology Kit (JSTK) Eval.

http://developer.java.sun.com/developer/products/jini/product.offerings.html
Sun Microsystems Jini System Software 1.0 Product Offerings — There are three Jini System Software product offerings: the Jini System Software Starter Kit (Jini Starter Kit), the Jini Technology Core Platform Compatibility Kit (TCK), and

[9] JavaSpaces Specification
[10] The Source for Java Technology — java.sun.com; JavaSpaces Technology. http://java.sun.com/products/javaspaces /
[11] Java & Internet Glossary. http://mindprod.com/jglossj.html

the JavaSpaces Technology Kit (JSTK). This site provides both software and documentation downloads.

http://java.sun.com/products/javaspaces/demos/index.html
Java Distributed Computing Demos — This site offers a collection of JavaSpaces demonstrations.

http://java.sun.com/products/javaspaces/specs/index.html
Java Distributed Computing Specifications — This site includes the JavaSpaces Specifications.

http://sern.ucalgary.ca/Courses/CPSC/547/W2000/webnotes/JavaSpaces/JavaSpaces.htm
JavaSpaces presented by Jonathan Neitz — This paper discusses the Java technology, JavaSpaces. This technology is based upon the services provided by Sun's Jini platform, and Java's serialization and RMI services.

http://www.sun.com/consumer-embedded/cover/jstk-990615.html
JavaSpaces™ Technology Kit (JSTK) 1.0 Download — The JSTK provides implementations of the JavaSpaces architecture specified in the Jini technology extended core platform. The 1.0 release of the JSTK software consists of the JavaSpaces software source and binary code, its API documentation, release notes for those classes and interfaces, and examples.

http://java.sun.com/products/javaspaces/faqs/jsfaq.html
Sun's JavaSpaces FAQ — This site provides answers to Frequently Asked Questions about JavaSpaces.

http://java.sun.com/products/javaspaces/
Sun's JavaSpaces Technology — JavaSpaces technology is a simple unified mechanism for dynamic communication, coordination, and sharing of objects between Java technology-based network resources like clients and servers.

http://wwwwswest.sun.com/jini/specs/js-spec.html
The JavaSpaces Specification; Version 1.0, January 1999 — This specification describes the architecture of JavaSpaces™ technology, which is designed to help you solve two related problems: distributed persistence and the design of distributed algorithms.

http://jiniworld.chonnam.ac.kr/document/javaspace/The%20Nuts%20and%20Bolts%20of%20Compiling%20and%20Running%20JavaSpaces(TM).htm
The Nuts and Bolts of Compiling and Running JavaSpaces Programs — This article steps you through the details of setting up your machine environment, compiling JavaSpaces programs, and correctly running them.

6.4.2 Books –– JavaSpaces

http://www.amazon.com/exec/obidos/ASIN/0201309556/qid=963583564/sr=1-1/002-4377486-0324804
JavaSpaces Principles, Patterns and Practice — by Eric Freeman, Susanne Hupfer, Ken Arnold; June 1999; Addison-Wesley Publishing Co.;
ISBN: 0201309556

http://www.amazon.com/exec/obidos/ASIN/1861002777/qid%3D960497042/sr%3D1-4/002-4377486-0324804
Professional Java Server Programming: with Servlets, Java Server Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and JavaSpaces — by Andrew Patzer, et. al. August 1999; Wrox Press Inc; ISBN: 1861002777

http://www.amazon.com/exec/obidos/ASIN/1861003552/qid%3D961431788/sr%3D1-12/002-4377486-0324804
Professional Jini & Java Spaces Programming — by Sing Li, Alvin Chin; June 2000; Wrox Press Inc; ISBN: 1861003552

http://www.artima.com/jini/booklist.html
Artima's List of Recommended Books about Jini and JavaSpaces — This site provides a comprehensive list of books about Jini and JavaSpaces.

6.5 Java Message Service (JMS)

JMS is an application program interface (API) for accessing enterprise messaging systems from Java programs.[12] Enterprise messaging provides a reliable, flexible service for the asynchronous exchange of critical business data and events throughout an enterprise. The JMS API adds to this a common API and provider framework that enables the development of portable, message based applications in the Java programming language. JMS supports the formal communication known as messaging between computers in a network.
Sun's JMS provides a common interface to standard messaging protocols and also to special messaging services in support of Java programs. Sun advocates the use of the Java Message Service for anyone developing Java applications, which can be run from any major operating system platform.[13]

6.5.1 Resources –– Java Message Service (JMS)

http://www.vistabonita.com/papers/JMS/JMS.html
Java Message Service — This paper discusses the JMS, JMS essentials, and the JMS environment.

http://java.sun.com/products/jms/
Java Message Service API 1.0.2 — This site provides documentation, FAQ, and a vendor list.

http://www.execpc.com/~gopalan/jms/jms.html
Java Message Service by Gopalan Suresh Raj — This article explores JMS as a standard Java-based interface to the message services of a Message-Oriented-Middleware (MOM) of some other provider.

http://www.jguru.com/jguru/faq/faqpage.jsp?name=JMS
jGuru Java Message Service FAQ — This site offers a collection of Frequently Asked Questions about the JMS.

[12] Java Message Service Version 1.0.2, November 9, 1999, cover
[13] WhatIs.Com. http://www.whatis.com/jms.htm

6.6 Enterprise JavaBeans (EJB)

Enterprise beans are server components written in the Java programming language. Enterprise beans contain the business logic for an application. There are two types of beans: session beans and entity beans.[14]

Enterprise JavaBeans servers reduce the complexity of developing middleware by providing automatic support for middleware services such as transactions, security, database connectivity, and more. EJB technology is based on the Java programming language, components can be deployed on any platform and operating system that supports the Enterprise JavaBeans standard, and any operating system.[15]

6.6.1 Resources –– Enterprise JavaBeans (EJB)

http://developer.java.sun.com/developer/onlineTraining/Beans/EJBTutorial/index.html
Enterprise JavaBeans Tutorial — This tutorial focuses on Building your first stateless Session Bean.

http://www.cetus-links.org/oo_javabeans.html
Cetus Links — Distributed Objects and Components: JavaBeans and Enterprise JavaBeans — This site offers a wealth of information on JavaBeans and Enterprise JavaBeans. Topics include examples, demos, central sites, related sites, tutorials, FAQs, standards, literature, organizations, conferences/workshops, and development environments.

http://patriot.net/~tvalesky/ejb.html
EJB Resources — This site includes a list of available products supporting EJB and open-source and/or freeware EJB implementations.

http://www.javasoft.com/products/ejb/
Enterprise JavaBeans Home Page: Javasoft —This site offers news and articles, customer success stories, an EJB Directory, white papers, FAQ, and specifications.

http://java.sun.com/products/ejb/docs.html
Enterprise JavaBeans Specifications — EJB specifications and downloads are available on this site.

[14] The Java 2 Enterprise Edition Developer's Guide, Version 1.2.1, May 2000, pg. 7
[15] The Source for Java Technology — java.sun.com. Enterprise JavaBeans Technology. http://www.javasoft.com/products/ejb/

http://www.flashline.com/
Flashline.com: The Software Component Marketplace — This marketplace is available to research, buy, and sell Java, COM, and CORBA software components. JavaBeans, Enterprise JavaBeans (EJB), and custom components are also available.

http://java.sun.com/products/ejb/faq.html
Sun's Enterprise JavaBeans FAQ — This site provides answers to Enterprise JavaBeans Frequently Asked Questions.

http://java.sun.com/products/ejb/
Sun's Enterprise JavaBeans Home Page — This site provides products and APIs, a developer connection, documentation and training, online support, community discussion, industry news, and case studies.

6.6.2 Books –– Enterprise JavaBeans (EJB)

http://www.amazon.com/exec/obidos/ASIN/0201604469/qid%3D962041588/sr%3D1-6/002-4377486-0324804
Enterprise Javabeans: Developing Component-Based Distributed Applications — by Thomas C. Valesky; May 1999;Addison Wesley Publishing Company; ISBN: 0201604469

http://www.amazon.com/exec/obidos/ASIN/1565928695/o/qid%3D962041104/sr%3D8-1/ref%3Daps%5Fsr%5Fb%5F1%5F3/002-4377486-0324804
Enterprise Javabeans — by Richard Monson-Haefel; March 2000; O'Reilly & Associates; ISBN: 1565928695

http://www.amazon.com/exec/obidos/ASIN/0471332291/o/qid%3D962041104/sr%3D8-2/ref%3Daps%5Fsr%5Fb%5F1%5F4/002-4377486-0324804
Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition — by Ed Roman; September 1999; John Wiley & Sons; ISBN: 0471332291

http://www.amazon.com/exec/obidos/ASIN/1861002777/qid%3D960497042/sr%3D1-4/002-4377486-0324804
Professional Java Server Programming: with Servlets, Java Server Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and JavaSpaces — by Andrew Patzer, et. al. August 1999; Wrox Press Inc; ISBN: 1861002777

6.7 Extensible Markup Language (XML)

XML is the Extensible Markup Language, a system for defining specialized markup languages that are used to transmit formatted data. XML is conceptually related to HTML, but XML is not itself a markup language. Rather it is a metalanguage, a language used to create other specialized languages.[16]

XML describes a class of data objects which are stored on computers, and partially describes the behavior of programs that process these objects. XML is a subset or restricted form of SGML, the Standard Generalized Markup Language (ISO 8879). The goal of XML is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.[17]

6.7.1 Resources –– Extensible Markup Language (XML)

http://xml.apache.org/
Apache XML Project — The goals of the Apache XML Project are: to provide commercial-quality standards-based XML solutions that are developed in an open and cooperative fashion; to provide feedback to standards bodies (such as IETF and W3C) from an implementation perspective; and to be a focus for XML-related activities within Apache projects.

http://www.w3.org/XML/
Extensible Markup Language (XML) — This site is provided by the W3C Architecture Domain. It provides events, specifications, working groups, forums, access to related sites, and software.

http://www.w3.org/TR/REC-xml
Extensible Markup Language 1.0 — This site contains the specification for XML 1.0. The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

[16] CNET Glossary. http://coverage.cnet.com/Resources/Info/Glossary/Terms/xml.html/
[17] Graphics Communications Association. http://www.gca.org/whats_xml/default.htm

http://www.gca.org/whats_xml/default.htm
Graphic Communications Association: What is XML? — This site provides access to W3C standards, a glossary, vocabularies, XML books, and conferences.

http://www.ibm.com/developer/xml/
IBM Developer Works: XML Zone — This site provides XML news, tutorials, reference materials, and software.

http://java.sun.com/xml/
Java Technology and XML — Implement XML technology using the Java programming language and the Java API for XML Parsing (JAXP) technology, and you've got something even more powerful: XML with cross-platform capabilities built in at the binary level, so that even the tools you use to parse and debug your XML code are platform-independent.

http://www.projectcool.com/developer/xmlz/
Project Cool XML Zone — Project Cool's XML Zone takes visitors from a basic introduction to XML, through detailed how-to tutorials and advanced techniques. A glossary is also available on the site.

http://www.cwi.nl/www.python.org/topics/xml/
Python and XML Processing — This site provides access to general XML resources, Python software for XML, and the Python/XML Special Interest Group.

http://www.xml.com/axml/testaxml.htm
The Annotated XML Specification by Tim Bray — This site presents an annotated specification for the Extensible Markup Language (XML) 1.0, W3C Recommendation, 10 February 1998.

http://www.oasis-open.org/cover/
The XML Cover Pages — This site provides a comprehensive online reference work for the Extensible Markup Language (XML) and its parent, the Standard Generalized Markup Language (SGML). The reference collection features extensive documentation on the application of the open, interoperable markup language standards, including XSL, XSLT, XPath, XLink, XPointer, HyTime, DSSSL, CSS, SPDL, CGM, ISO-HTML, and others.

http://www.voicexml.org
VoiceXML Forum — The VoiceXML Forum is an industry organization founded by AT&T, IBM, Lucent and Motorola, and chartered with establishing and promoting the Voice eXtensible Markup Language (VoiceXML), a new standard essential to making Internet content and information accessible via voice and phone.

http://www.w3.org/XML/Activity
W3C Extensible Markup Language (XML) Activity — Activity statements provide an executive overview of W3C's work in this area.

http://www.WDVL.com/Authoring/Languages/XML/Resources.html
WDVL: XML Resources — WDVL's XML Resources section is an extensive collection of XML resources, including links to all major XML sites, news, mailing lists, books, references, FAQs and more.

http://msdn.microsoft.com/xml/index.asp
Web Workshop: XML Home — This page provides information on Microsoft's support of the Extensible Markup Language (XML).

http://hotwired.lycos.com/webmonkey/xml/?tw=xml
Webmonkey, The Web Developer's Resource: XML — This site offers an overview of XML and related resources.

http://developer.netscape.com/tech/xml/index.html
XML Developer Central — This site offers a broad collection of resources in specifications, tools, newsgroups, tutorials, articles, sample code, demos, and news.

http://www.cs.caltech.edu/~adam/LOCAL/xml.html
XML Links by Adam Rifkin and Rohit Khare — This site provides a wealth of information on XML including recommended reading, books, links to XML for beginners, recommendations, working drafts, specifications, and notes, advanced XML and SGML topics, XML applications, and XML software.

http://www.heise.de/ix/raven/Web/xml/
XML Page — This site offers working notes, drafts, and notes; schemes, linking, formatting, history, namespaces, querying, and a developer's list of resources.

http://www.xml-zone.com/
XML Zone — This zone of the Development Exchange (DevEx) provides access to XML news, FAQ, articles, sites, and newsgroups. Visitors can subscribe to free news, tips, and article updates.

http://www.xml.org/xml-dev/index.shtml
XML-DEV @XML.ORG Discussion Group — This site contains the XML-DEV archives. This group is for XML developers to share ideas and stay up-to-date on issues in the XML development community.

http://www.xml.com/
XML.com — XML.com is a collaborative partnership between Seybold Publications and Songline Studios, an affiliate of O'Reilly & Associates. The site is designed to serve both people who are already working with XML and those

HTML users who want to "graduate" to XML's power and complexity. A core feature of the site is the Annotated XML Specification, created by Tim Bray, co-editor of XML 1.0 and a contributing editor for XML.com.

http://www.xmlinfo.com/
XMLINFO — This site provides resources to new XML users, specifications and drafts, books, conferences and seminars, courses and training, papers, and examples.

http://www.ebxml.org/
ebXML — The mission of ebXML is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties.

6.7.2 White Papers and Books –– Extensible Markup Language (XML)

http://www.amazon.com/exec/obidos/ASIN/0130866016/qid%3D960499184/sr%3D1-51/002-4377486-0324804
Building Web Sites with XML — by Michael Floyd; December 21, 1999; Prentice Hall; ISBN: 0130866016

http://www.amazon.com/exec/obidos/ASIN/1861002270/qid%3D960497042/sr%3D1-42/002-4377486-0324804
Designing Distributed Applications with XML: Asp Ie5 Ldap and Msmq — by Stephen T. Mohr; May 1999; Wrox Press Inc; ISBN: 1861002270

http://www.amazon.com/exec/obidos/ASIN/0596000162/qid%3D960497042/sr%3D1-43/002-4377486-0324804
Java and XML — by Brett McLaughlin, Mike Loukides (Editor); O'Reilly Java Tools; July 2000; O'Reilly & Associates; ISBN: 0596000162

http://www.amazon.com/exec/obidos/ASIN/1861002777/qid%3D960497042/sr%3D1-4/002-4377486-0324804
Professional Java Server Programming: with Servlets, Java Server Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and JavaSpaces — by Andrew Patzer, et. al. August 1999; Wrox Press Inc; ISBN: 1861002777

http://www.amazon.com/exec/obidos/ASIN/1861003110/qid%3D960497042/sr%3D1-7/002-4377486-0324804
Professional XML — by Mark Birbeck, et. al. January 2000; Wrox Press Inc; ISBN: 1861003110

http://www.amazon.com/exec/obidos/ASIN/1861002289/qid%3D960499184/sr%3D1-72/002-4377486-0324804
XML Design and Implementation — by Paul Spencer; April 1999; Wrox Press Inc; ISBN: 1861002289

http://www.amazon.com/exec/obidos/ASIN/1565927095/qid%3D960497042/sr%3D1-9/002-4377486-0324804
XML Pocket Reference — by Robert Eckstein; October 1999; O'Reilly & Associates; ISBN: 1565927095

http://www.amazon.com/exec/obidos/ASIN/0672315149/qid%3D960497042/sr%3D1-14/002-4377486-0324804
XML Unleashed — by Michael Morrison, et. al. December 21, 1999; Sams; ISBN: 0672315149

http://www.amazon.com/exec/obidos/ASIN/0789722429/qid%3D960497042/sr%3D1-1/002-4377486-0324804
XML by Example — by Benoit Marchal December 14, 1999; Que; ISBN: 0789722429

http://www.amazon.com/exec/obidos/ASIN/0201433354/qid%3D960497042/sr%3D1-27/002-4377486-0324804
XML: A Manager's Guide — by Kevin Dick; Addison-Wesley Information Technology Series; October 13, 1999; Addison-Wesley Publishing Co.; ISBN: 0201433354

http://www.amazon.com/exec/obidos/ASIN/0764532367/qid%3D960497042/sr%3D1-12/002-4377486-0324804
XML™ Bible — by Elliotte Rusty Harold; July 1999; IDG Books Worldwide; ISBN: 0764532367

http://www.amazon.com/exec/obidos/ASIN/076453310X/qid%3D960497042/sr%3D1-37/002-4377486-0324804
XML™: A Primer — by Simon St. Laurent; September 1999; IDG Books Worldwide; ISBN: 076453310X

http://www.oasis-open.org/cover/bib-strt.html
Cover's List of Basic Books on SGML/XML and Related Standards — This list is an annotated collection of titles on SGML, XML, and related standards.

http://www.w3.org/TR/REC-xml
Extensible Markup Language 1.0 — This site contains the specification for XML 1.0. The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with

HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

http://www.csclub.uwaterloo.ca/u/relander/XML/
Generally Markup: XML Resources — This site provides white papers, tutorials, online courses, and a list of related sites.

http://www.xmlbooks.com/
Goldfarb's Guide to All the XML Books in Print — This collection is sorted by introduction to XML, its major applications, and tools; program development with XML; DTDs and schemas; XML reference; knowledge management; learning the foundations of XML; and standards.

http://www.oasis-open.org/cover/xmlIntro.html
The XML Cover Pages: Introducing the Extensible Markup Language (XML) — This page offers a collection of articles that introduce XML.

http://webreview.com/wr/pub/XML
Webreview.com — XML articles on using the eXtensible Markup Language as a mechanism for defining standard Web-based formats for the interchange of information.

http://www.vb-bookmark.com/vbXml.html
XML Bookmark: XML Programming Resource Directory — This rich resource offers a collection of FAQ, tutorials, and articles on XML.

http://developer.netscape.com/tech/xml/index.html
XML Developer Central — This site offers a broad collection of resources in specifications, tools, newsgroups, tutorials, articles, sample code, demos, and news.

http://xml.superexpert.com/
XML Superexpert — This site contains tutorials, software, access to experts, and articles.

http://www.xml.org/xmlorg_resources/whitepapers.shtml
XML.ORG White Papers — This collection includes papers targeted to general audiences, managers, and developers.

http://www.xml.com/pub
XML.com — XML.com is a collaborative partnership between Seybold Publications and Songline Studios, an affiliate of O'Reilly & Associates. The site is designed to serve both people who are already working with XML and those HTML users who want to "graduate" to XML's power and complexity. A core feature of the site is the Annotated XML Specification.

6.7.3 Educational Materials –– Extensible Markup Language (XML)

http://metalab.unc.edu/xml/
Cafe con Leche XML News and Resources —This site provides access to XML daily news updates, specifications, books, resources, development tools, non-validating parsers, online validators and syntax checkers, formatting engines, browsers, class libraries, editors, XML applications, and additional sites.

http://www.dacs.dtic.mil/training/xml-course.shtml
DACS Course: Introduction to XML — This web-based course presents the fundamentals of XML. The objectives of the course are to understand what XML is; to understand what XML is not; to understand the difference between XML and HTML; to learn how to develop well-formed and valid XML; to see some real XML; and to understand the link between XML and Java.

http://www.finetuning.com/tutorials.html
Fine Tuning XML Tutorial — The goal of this tutorial is to start at the very, very, beginning. That's right: all the way back to HTML! And then progress from there gradually all the way through to well-formed XML, through to using XML with CSS style sheets, and then XSLT style sheets, then XPath, XLink, and XPointer, and eventually reaching DTDs and XML Schemas (along with whatever else should appear that seems relevant along the way).

http://www.csclub.uwaterloo.ca/u/relander/XML/
Generally Markup: XML Resources — This site provides white papers, tutorials, online courses, and a list of related sites.

http://www.projectcool.com/developer/xmlz/
Project Cool XML Zone — This site takes visitors from a basic introduction to XML, through detailed how-to tutorials and advanced techniques. A glossary is also available on the site.

http://www.geocities.com/SiliconValley/Peaks/5957/xml.html
What is XML? — This collection of resources was compiled by L.C. Rees. It includes resources for learning the eXtensible Markup Language and selected links of continuing relevance.

http://www.vb-bookmark.com/vbXml.html
XML Bookmark: XML Programming Resource Directory — This rich resource offers a collection of FAQ, tutorials, and articles on XML.

http://developer.netscape.com/tech/xml/index.html
XML Developer Central — This site offers a broad collection of resources in specifications, tools, newsgroups, tutorials, articles, sample code, demos, and news.

http://www.katungroup.com/xml.htm
XML Links — This site presents a collection of XML links including articles, developer resources, tutorials, publications, examples, and tools.

http://www.xmlpitstop.com/
XML Pitstop — The mission of this website is to provide the developer community with a central location to learn about XML, locate resources and network with fellow developers. The site is organized by the following sections: examples, tools, tutorials, developers, resources, books, and user group.

http://www.w3schools.com/default.asp
XML School — This site contains a tutorial for XML. It provides an introduction, describes how XML can be used, and provides syntax and related topics.

http://xml.superexpert.com/
XML Superexpert — This site contains tutorials, software, access to experts, and articles.

http://developerlife.com/
developerlife.com — The tutorials found here show students how to use Java2, XML, Swing, Servlet, JDBC and RMI APIs to create real world applications using XML and Java2. Complete source code files (with documentation) are available for download in all the tutorials and articles.

http://msdn.microsoft.com/xml/tutorial/
msdn Online Web Workshop: XML Tutorial — This tutorial consists of a set of interactive lessons, listed below, that walk a student through typical XML authoring and development tasks.

6.7.4 Frequently Asked Questions (FAQ) –– Extensible Markup Language (XML)

http://www.inquiry.com/techtips/xml_pro/
Ask the XML Pro — This site offers an archive of responses to XML related inquiries.

http://builder.cnet.com/Authoring/Xml20/
CNET Builder.com: 20 Questions on XML — This site presents answers to several basic questions on XML.

http://www.perlxml.com/faq/perl-xml-faq.html
Perl XML FAQ — This FAQ contains information related to using and manipulating XML with Perl.

http://www.vb-bookmark.com/vbXml.html
XML Bookmark: XML Programming Resource Directory —This rich resource offers a collection of FAQ, tutorials, and articles on XML.

http://www.finetuning.com/faq.html
finetuning.com's FAQ — This site offers a wide variety of XML FAQ.

http://developer.irt.org/script/xml.htm
irt.org: Extensible Markup Language (XML) FAQ Knowledge Base — This site provides a collection of XML FAQ.

6.7.5 Tools–– Extensible Markup Language (XML)

http://metalab.unc.edu/xml/
Cafe con Leche XML News and Resources — This site provides access to XML daily news updates, specifications, books, resources, development tools, non-validating parsers, online validators and syntax checkers, formatting engines, browsers, class libraries, editors, XML applications, and additional sites.

http://www.garshol.priv.no/download/xmltools/
Free XML Tools and Software — The resources on this site are grouped by category, name, platform, vendor, and standard.

http://www.goxml.com/
GoXML.com — GoXML.com is an XML context-based search processor. The GoXML Project was launched to create a new breed of search vehicle that can index, store and allow accurate searching of XML data.

http://java.sun.com/xml/download.html
Java Technology and XML — The Java API for XML Parsing (JAXP) Optional Package provides basic functionality for reading, manipulating, and generating XML documents through pure Java APIs. It is a thin and lightweight API that provides a standard way to seamlessly integrate any XML-compliant parser with a Java application. The software can be downloaded from this site.

http://www.infotek.no/sgmltool/guide.htm
The Whirlwind Guide to SGML & XML Tools and Vendors — This guide by Steve Pepper provides tool information by tool/resource category, by product name, by vendor name, and by service providers.

http://www.oasis-open.org/cover/check-xml.html
The XML Cover Pages: Check or Validate XML — This page provides access to a collection of tools for checking or validating XML.

http://developer.netscape.com/tech/xml/index.html
XML Developer Central — This site offers a broad collection of resources in specifications, tools, newsgroups, tutorials, articles, sample code, demos, and news.

http://www.katungroup.com/xml.htm
XML Links — This site presents a collection of XML links including articles, developer resources, tutorials, publications, examples, and tools.

http://www.xmlpitstop.com/
XML Pitstop — The mission of this website is to provide the developer community with a central location to learn about XML, locate resources and network with fellow developers. The site is organized by the following sections: examples, tools, tutorials, developers, resources, books, and user group.

http://xml.superexpert.com/
XML Superexpert — This site contains tutorials, software, access to experts, and articles.

http://www.xmlsoftware.com/
XMLSoftware: The XML Software Site — This site contains a collection of XML software. The site contains browsers, editors, parsers, utilities, and related tools.

6.8 DII COE (Defense Information Infrastructure Common Operating Environment)

The DII COE concept is best described as a flexible architecture and approach for building interoperable systems. The COE is a multifaceted concept. It is not a system, but is a foundation upon which open systems can be built. A foundation that provides functionality to target systems for services such as data manipulation, network communications, database storage and others. The COE includes rules, methodologies and tools which form a framework for system development and integration.[18]

The DII COE is an open architecture designed around a client/server model. The COE is not a system; it is a foundation for building an open system. The DII COE is best described as:

-- an architecture that is fully compliant with the DoD's Technical Architecture for Information Management (TAFIM), Volumes 2 and 3,
-- an approach for building interoperable systems,
-- a collection of reusable software components,
-- a software infrastructure for supporting mission area applications,
-- a set of guidelines and standards.[19]

The DII COE concept encompasses the following:

-- an architecture and approach for building interoperable systems;
-- an infrastructure for supporting mission area applications;
-- a rigorous definition of the runtime execution environment;
-- a rigorous set of requirements for achieving COE compliance;
-- an automated toolset for enforcing COE principles and measuring COE compliance;
-- an automated process for software integration;
-- a collection of implemented, reusable software components;
-- an approach and methodology for software reuse;
-- a collection of application program interfaces (APIs) for accessing COE components.[20]

[18] Defense Information Infrastructure Common Operating Environment. http://www.dmcdayton.day.disa.mil/Services/Midtier/DIICOE.htm
[19] DE Guidebook — Appendix D: DE Process Foundations — Part III — Coordinating with DoD and USAF Standards and Mandates. http://www.asset.com/stars/loral/domain/guide/degaxd51.htm
[20] AdaIC News Fall 1997. http://adaic.org/news/Newsletter/1997/fall/7.htm

6.8.1 Resources –– DII COE

http://dii-sw.ncr.disa.mil/coe/
Defense Information Infrastructure Common Operating Environment (DII COE) Home Page — Serving as a DISA/JIEO/IPESO Information Clearinghouse, the IPESO provides DISA, DoD Program Managers, and other users with access to DII COE products and services and responds to customer questions, problems, and issues via on-line help desk support.

http://coeeng.ncr.disa.mil/REFERENCE_PAGES/JCSCOT/JCSCOT.HTM
COTS Inclusion in the DII COE — This paper discusses considerations surrounding the inclusion of Commercial-off-the-shelf (COTS) products in the Defense Information Infrastructure (DII) Common Operating Environment (COE).

http://dod-ead.mont.disa.mil/cm/cm_page.html
DII COE Configuration Management — This page provides access to a collection of databases and documentation.

http://diicoe.disa.mil/coe
DII COE: Defense Information Infrastructure Common Operating Environment — The DII COE originated with a simple observation about command and control systems: certain functions (mapping, track management, communication interfaces, etc.) are so fundamental that they are required for virtually every command and control system. The DII COE is presently used in two systems: the Global Command and Control System (GCCS), and the Global Combat Support System (GCSS). Both systems use the same infrastructure and integration approach, and the same COE components for functions that are common.

http://dii-sw.ncr.disa.mil/cseic/listserv/maillist.html
DII COE Listserv — The DISA Information Processing Engineering Support Organization (IPESO) has developed a DII COE listserv. The DII COE listserv provides a moderated forum for COE developers and users to disseminate information, share ideas and lessons learned, collaborate, and post questions and answers.

http://ssed1.ncr.disa.mil/
DISA DII Enterprise Licensing Program — The mission of the Defense Information Systems Agency (DISA) Defense Information Infrastructure (DII) Enterprise Licensing Program is to provide very low cost, easily purchased and managed Commercial-off-the-shelf (COTS) components for the DII community.

http://dii-sw.ncr.disa.mil/coe/topics/atd/
DISA's Advanced Technologies — Links to briefings on topics in OOT, CORBA, and Java, including applications to Ada and reengineering.

http://www.dmcdayton.day.disa.mil/Services/Midtier/DIICOE.htm
Defense Information Infrastructure (DII) Common Operating Environment (COE) — The DII COE concept is best described as a flexible architecture and approach for building interoperable systems.

http://www.hanscom.af.mil/dii-coe/faq.htm
Defense Information Infrastructure -- Implementing DII Compliant Systems -- Frequently Asked Questions — This is a collection of answers to FAQ regarding the DII COE.

http://www.disa.mil/disahomejs.html
Defense Information Systems Agency (DISA) — This is DISA's main home page. DISA is the Department of Defense (DOD) agency responsible for information technology and is the central manager of major portions of the Defense Information Infrastructure (DII). At this site, you will find information on COE, GCCS and GCSS.

http://www-jta.itsi.disa.mil/jta/jtamemo2.pdf
DoD Joint Technical Architecture (JTA), Version 2.0, 30 November 1998 -- Implementation Memo — The JTA implementation memorandum is signed by the Tri-Chairs of the Architecture Coordination Council (ACC). It makes JTA Version 2.0 effective for immediate use, superseding VTA Version 1.0.

http://dod-ead.mont.disa.mil/aug_home/index.html
GCCS -- Global Command and Control System — The Global Command and Control System (GCCS) is an automated information system designed to support deliberate and crisis planning with the use of an integrated set of analytic tools and the flexible data transfer capabilities. GCCS will become the single C4I system to support the warfighter from foxhole to command post.

http://www.disa.mil/gcss/gcsshome.html
GCSS -- Global Combat Support System — The C4I for the Warrior (C4IFTW) concept is committed to the challenge of meeting the warrior's quest for information needed to achieve victory for any mission, at any time and at any place. C4IFTW is the vision and roadmap for creating a broadly connected joint system that provides total battle space information to the warrior. GCSS is the final piece of the C4IFTW concept. It is a demand-driven, joint warfighter-focused initiative to accelerate delivery of improved combat support capabilities. Using the same approach, methodology, practices, tools, and integration procedures as the Global Command and Control System (GCCS), GCSS is a strategy that integrates existing combat support systems to gain efficiency and interoperability in support of the warfighter. GCSS will provide the warfighter with a fused, real-time combat support view of the battle space.

http://www.stsc.hill.af.mil/CrossTalk/1999/sep/engert.asp
Introduction to the Defense Information Infrastructure (DII) Common Operating Environment (COE) — The DII COE provides a foundation for building interoperable command and control systems using reusable software components. This article appeared in CrossTalk, a publication of the Software Technology Support Center (STSC).

http://www.sei.cmu.edu/str/descriptions/diicoe_body.html
Software Technology Review -- Defense Information Infrastructure Common Operating Environment (DII COE) — The purpose of DII COE is to field systems with increasing interoperability, reusability, portability, and operational capability, while reducing development time, technical obsolescence, training requirements, and life-cycle cost.

http://www.oasis-open.org/cover/dii-coeXMLRegistry.html
The XML Cover Pages -- DII Common Operating Environment (COE) XML Registry — This Registry enables the consistent use of XML, both vertically within projects and horizontally across organizations. The DII COE XML Registry constitutes guidance in the generation and use of XML within the COE v4.x data environment and is the authoritative source for approved XML data and metadata components.

6.9 Unified Modeling Language (UML)

UML is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. UML is a standard notation for the modeling of real-world objects as a first step in developing an object-oriented program. Its notation is derived from and unifies the notations of three object-oriented design and analysis methodologies: Grady Booch's methodology for describing a set of objects and their relationships; James Rumbaugh's Object-Modeling Technique (OMT); and Ivar Jacobson's approach which includes a use case methodology.[21]

6.9.1 Resources –– Unified Modeling Language (UML)

http://home.earthlink.net/~salhir/applyingtheuml.html
Applying the Unified Modeling Language (UML) — The UML is a modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system-intensive process. This paper elaborates on the application of the UML.

http://liinwww.ira.uka.de/bibliography/SE/uml.html
Bibliography on the Unified Modeling Language — This bibliography contains references to over 300 publications about the UML.

http://www.sente.ch/cetus/oo_uml.html
Cetus Links -- Architecture and Design: Unified Modeling Language (UML) — This site provides access to Unified Modeling Language (UML) tutorials, FAQs, references, standards, mailing lists, articles, books, projects, conferences/workshops, utilities/tools, software, and other resources.

http://home.pacbell.net/ckobryn/uml.htm#OMG UML Specification
Cris Kobryn's UML Resource Page — This site provides links to the OMG UML specification, UML events, UML vendors, UML books, and other UML links.

http://www.omg.org/uml/
Object Management Group (OMG) UML Resource Page — This site offers documentation, articles, information, and useful links.

http://www.objectsbydesign.com/tools/umltools_byCompany.html
Objects by Design: UML Modeling Tools — This site presents a collection of UML tools and includes information on the vendor, product, platform, and price.

[21] WhatIs?Com. http://www.whatis.com/

http://www.objectsbydesign.com/tools/umltools_byCompany.html
The UML Zone — This site presents FAQ, UML articles, links to related sites, UML newsgroups, and UML book reviews.

http://www.platinum.com/corp/uml/uml.htm
UML Center — This site provides UML information, UML events, articles and publications, industry links, and UML partners.

http://www.softdocwiz.com/UML.htm
UML Dictionary — This Unified Modeling Language (UML) Dictionary, put together by Kendall Scott, contains over 600 terms.

http://www.jeckle.de/uml_pub.htm
UML Publications — This site contains links to a large collection of online UML publications and articles.

http://www.holub.com/class/oo_design/uml.html
UML Reference Card — This site is a printable reference page of UML diagrams. It is segmented into Static-Model Diagrams, Dynamic-Model Diagrams, and Activity Diagrams, each of which is accompanied by definitions of the diagrams' terms.

http://www.rational.com/uml/index.jtmpl
UML Resource Center — The Unified Modeling Language™ (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, making a "blueprint" for construction. This site contains UML documentation.

http://home.earthlink.net/~salhir/whatistheuml.html
What is the Unified Modeling Language (UML)? — The UML is a modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system-intensive process. This paper elaborates on the definition of the UML.

6.9.2 Books –– Unified Modeling Language (UML)

http://www.amazon.com/exec/obidos/ASIN/020130998X/qid=964461209/sr=1-1/102-5607525-4562541
The Unified Modeling Language Reference Manual — by James Rumbaugh, Ivar Jacobson, Grady Booch December 1998; Addison-Wesley Publishing Co.; ISBN: 020130998X

http://www.amazon.com/exec/obidos/ASIN/0201571684/qid=964461209/sr=1-2/102-5607525-4562541
The Unified Modeling Language User Guide — by Grady Booch, Ivar Jacobson, James Rumbaugh October 30, 1998; Addison-Wesley Publishing Co.; ISBN: 0201571684

http://www.oreilly.com/catalog/umlnut/
UML in a Nutshell — by Sinan Si Alhir September 1998; O'Reilly and Associates; ISBN: 1-56592-448-7

6.9.3 Frequently Asked Questions (FAQ) –– Unified Modeling Language (UML)

http://www.rational.com/uml/gstart/faq.jtmpl
UML FAQ for Beginners — This site presents answers to FAQ often posed by UML beginners.

http://www.microgold.com/Stage/UML_FAQ.html
UML Frequently Asked Questions — This collection of FAQ addresses general information, overview, metamodel, notation, and process questions about UML.

http://www.uml-zone.com/umlfaq.asp
UML Zone Frequently Asked Questions — This site presents a collection of common questions.

6.10 COnstructive COTS (COCOTS)

COCOTS is a cost estimation model designed to capture explicitly the most important costs associated with COTS component integration. COCOTS is actually an amalgam of four related sub-models, each addressing individually what the authors have identified as the four primary sources of COTS software integration costs. These are costs due to the effort needed to perform (1) candidate COTS component assessment, (2) COTS component tailoring, (3) the development and testing of any integration or "glue" code needed to plug a COTS component into a larger system, and (4) increased system level programming due to volatility in incorporated COTS components.[22]

6.10.1 Resources –– COCOTS

http://fast.faa.gov/pricing/c1919-19E.htm#19E.3
Toolsets / FAA Pricing Handbook — This site reviews the COTS cost estimation model status. The current model provides insight into the most important factors that should be considered when estimating the cost of integrating COTS components, regardless of the specific tool or methodology used to perform that estimation.

http://sunset.usc.edu/research/COCOTS/cocots_main.html
University of Southern California Center for Software Engineering: COnstructive COTS — This site provides a model rationale, model description, a description of four submodels, and overall cost estimation guidelines.

[22] University of Southern California — Center for Software Engineering. Model Rationale. http://sunset.usc.edu/research/COCOTS/cocots_main.html#rationale

7 Essential Reading on COTS

This section focuses on key contributions on COTS-based development. This list is not definitive or comprehensive by any means.

The following papers give a broad introduction to the topic.
Wallnau, K.C., Carney, D., Pollak, B., "How COTS Software Affects the Design of COTS-Intensive Systems". Available WWW http://interactive.sei.cmu.edu/Features/1998/June/COTS_Evaluation/COTS_Evaluation.htm (1998).

Carney, D.J., Oberndorf, P.A., "The Commandments of COTS: Still in Search of the Promised Land", CrossTalk, May 1997, Vol.10, No.5, pp. 25-30.

COTS and Open Systems — An Overview
http://www.sei.cmu.edu/str/descriptions/cots_body.html

Component-Based Software Development/COTS Integration
http://www.sei.cmu.edu/str/descriptions/cbsd_body.html

This paper describes key issues encountered by projects using COTS.
Vidger, M.R., Gentleman W.M., Dean, J., "COTS Software Integration: State of the Art", National Research Council of Canada, 1998. Available at http://wwwsel.iit.nrc.ca/abstracts/NRC39198.abs

This paper is essential for design issues when COTS are involved.
Garlan, D., Allen, R., Ockerbloom, J., "Architectural Mismatch or Why It's Hard to Build Systems out of Existing Parts", Proceedings of the 1995 International Conference on Software Engineering, Seattle, WA, USA, 1995, pp. 179-185.

This book describes Catalysis, probably the most mature process and notation for component based development. Although not specific to COTS, it highlights some essential design issues and notation.
Desmond D'Souza, Alan Wills: *Objects, Components and Frameworks With UML: The Catalysis Approach*, Addison-Wesley, 1998.

8 Concluding Remarks

COTS-based development is probably here to stay. There is evidence of both successful use of COTS, and failures in projects using them. Overall, there is no way to state that COTS are only a success or a failure factor. Success depends on several factors. We try to summarize them and analyze their influence on the success of a project.

Criticality of the Project. If a project is highly critical (meaning that a failure in the software system can lead to loss of life, or environmental hazard, or jeopardy on business critical functions), the use of COTS should be seriously evaluated. A user has limited control on both the quality of the COTS product and the quality of the process used to produce it. Techniques to certify the quality of COTS are still immature. As of today, a COTS can be trusted only if it is used by a large user base, during a long period. In hardware engineering, life critical or mission critical systems use off-the-shelf hardware that has been used by a large community for several years. A similar approach should be used for COTS software too. On the other hand a piece of software that satisfies this condition is usually more reliable than an equivalent module built in-house.

Dependability of the COTS and the Vendor. There are little or no techniques that allow a user to assess the dependability of a COTS (in the sense of availability of the functionality promised by the documentation, reliability of the functionality, availability and quality of documentation). Using the product is the best option. An indirect indicator is the existence of the product since a long time, and a satisfied community of users. As for the dependability of the vendor, the same applies: trying it, and checking the community of users. Clearly, COTS just released on the market by start ups and with few users are the most risky.

COTS Domain. We observe that dependable COTS, with a large customer base, are pretty common in the domain of software services common to any application: operating systems, networking, databases, user interfaces, GUIs, office automation, mathematical libraries, and so on. In these cases the need for a product appeared early, and spurred companies that satisfy them. Further, competing vendors usually exist for these products.
In these cases, using the COTS is the default choice. *Not* using a COTS (say a database, or a GUI) is a choice that has to be justified. Usually this can happen when very specific needs demand for particular solutions.
Dependable, diffused COTS exist also for some domains specific to only a class of applications. Accounting, warehouse management, payroll management, and more recently ERP (Enterprise Resource Planning) are some of these domains.

Project Architecture. Using a single COTS avoids integration problems with other COTS. If several COTS have to be integrated, it is more likely that conflicts among them (due to different assumptions in control, GUIs, interfaces in general) arise.

By aggregating the factors above we can argue that using a dependable COTS in a non critical project based on a single COTS is probably a reasonable choice. On the other hand integrating several COTS, just released, in a highly critical project means probably asking for trouble. In between lies a twilight zone where the decision on using COTS has to be carefully evaluated case by case.

9 References

[Abst 2000] Abst, Chris, Boehm B., Clark E.B., Empirical Observations on COTS Software Integration Effort Based on the Initial COCOTS Calibration Database, *ICSE 2000 COTS Workshop*, Limerick, Ireland, June 2000 available at http://wwwsel.iit.nrc.ca/projects/cots/icse2000wkshp/index.html

[Beizer 1995] Beizer, Boris, *Black-Box Testing Techniques for Functional Testing of Software and Systems,* New York: Wiley 1995

[Brownsword et al. 1998] Brownsword, L., Carney, D., Oberndorf, T., The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components, 1998, available at http://interactive.sei.cmu.edu/Features/1998/June/Applying_COTS/Applying_COTS.htm

[Carney 1997] 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.

[Carney, Long 2000] Carney, D., Long, F., What Do You Mean by COTS?, *IEEE Software*, March/April 2000, pp. 83-86.

[D’Souza 1998] Desmond D'Souza, Alan Wills: *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison-Wesley, 1998.

[Davis, Williams 1997] Davis, M.J., Williams, R.B., Software Architecture Characterization, Proceedings of the 1997 Symposium on Software Reusability (SSR'97), Boston, USA, May, 1997, pp. 30-38.

[DeMillo 1987] DeMillo, Richard, McCracken, W. Michael, Martin, R. J., Passafiume, John F., *Software Testing and Evaluation*, Menlo Park, CA: Benjamin/Cummings Publishing Co. Inc., 1987

[Fowler 1997] Martin Fowler, Kendall Scott: *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, 1997.

[Fox 1998] Fox, G., Marcom, S., Lantner, K., A Software Development Process for COTS-based Information System Infrastructure. Part II: Lessons Learned, *CrossTalk*, April 1998, available at http://www.stsc.hill.af.mil/CrossTalk/1998/apr/process.asp

[Gallagher 1991] Gallagher, K B, Lyle, J R, Using Program Slicing in Software Maintenance, *IEEE Transactions on Software Engineering*, 17:751-761, Aug 1991

[Garlan et al. 1995] Garlan, D., Allen, R., Ockerbloom, J., Architectural Mismatch or Why it's Hard to Build Systems out of Existing Parts, Proceedings of the 17th International Conference on Software Engineering, Seattle, 1995.

[Ghosh 1998] Ghosh, A. K., Schmid, M., Shah, V., *Testing the Robustness of Windows NT Software*. 1998 International Symposium on Software Reliability Engineering (ISSRE98), available at http://www.rstcorp.com

[Ghosh 1999] Ghosh A. K., Schmid M. An Approach to Testing COTS Software for Robustness to Operating System Exceptions and Errors, 1999 International Symposium on Software Reliability Engineering (ISSRE99) Nov 1-2 1999, Boca Raton Fl, available at http://www.rstcorp.com

[Haynes et al. 1997] Haynes, G., Carney, D., Foreman, J., Component-Based Software Development / COTS Integration, 1997, available at http://www.sei.cmu.edu/str/descriptions/cbsd.html

[Hetzel 1984] Hetzel, William, *The Complete Guide to Software Testing*. Wellesley, MA: QED Information Sciences, Inc. 1984

[Hissam 1998] Hissam, Scott, Experience Report: Correcting System Failure in a COTS Information System, 1998 In Proceedings of 15th International Conference on Software Maintenance, November 1998.

[Kitchenham 1997] Kitchenham, B.A. and Linkman, S.G. and Law, D. DESMET: A Methodology for Evaluating Software Engineering Methods and Tools. *IEE Computing & Control Journal*, June 1997, pp120-126.

[Knight 1986] Knight, J.C. and N.G. Leveson, An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming, *IEEE Transactions on Software Engineering*, SE-12(1):96-109, January 1986.

[Kohl] Kohl, Ronald J., *V&V of COTS Dormant Code: Challenges and Issues*, http://wwwsel.iit.nrc.ca/projects/cots/icsewkshp/papers.html

[Kontio 1996] J. Kontio. A Case Study in Applying a Systematic Method for COTS Selection, *Proceedings of the 18th International Conference on Software Engineering*, Berlin, March 1996.

[Kropp 1998] Kropp, N. P., *Automatic Robustness Testing of Off-the-Shelf Software Components*. Institute for Complex Engineering Systems, Carnegie Institute of Technology, Carnegie Mellon University, CMU/ICES-TR-01-27-98

[McGraw] McGraw, G, Viega, J, *Why COTS Software Increases Security Risks*, available at http://www.rstcorp.com

[Morisio 1997] Morisio M., Tsoukiàs A., IUSWARE: A Formal Methodology for Software Evaluation and Selection, *IEE Proceedings on Software Engineering*, vol. 144, 162 - 174, 1997.

[Morisio 2000] Morisio M., Seaman C., Parra A., Basili V., Kraft S., Condon S., Investigating and Improving a COTS-Based Software Development Process, 22nd International Conference on Software Engineering, Limerick, Ireland, June 2000.

[Oberndorf 1997] Oberndorf, T., COTS and Open Systems - An Overview, 1997, available at http://www.sei.cmu.edu/str/descriptions/cots.html#ndi

[Parra 1997] 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 Proceedings of the 22nd Software Engineering Workshop, NASA/Goddard Space Flight Center, December 1997, pp. 21-56.

[Reifer 1999] Reifer D., Ragan T., Kalb G.E., COTS Software Management: Taming the Beast, May 1999, available at http://www.reifer.com

[Rumbaugh 1998] James Rumbaugh, Ivar Jacobson, Grady Booch: *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1998.

[Saaty 1980] T. Saaty, *The Analytic Hierarchy Process*, McGraw Hill, New York, 1980.

[Schneidewind 1998] Schneidewind, N. Methods for Assessing COTS Reliability, Maintainability, and Availability, 1998 In Proceedings of 15th International Conference on Software Maintenance, November 1998.

[SEL 1998] NASA/SEL, SEL COTS Study, Phase 1, Initial Characterization Study, SEL-98-001, August 1998.

[Sha 1998] Lui Sha, John B. Goodenough, Bill Pollak, Simplex Architecture: Meeting the Challenges of Using COTS in High-Reliability Systems, *CrossTalk*, April 1998.

[Shaw 1995] Shaw, M., Architectural Issues in Software Reuse: It's Not Just the Functionality, It's Packaging, Proceedings of the Symposium on Software Reusability, 1995, Seattle, WA, USA, pp. 3-6.

[Sparks et al. 1996] Sparks, S., Benner, K., Faris, C., Managing Object-Oriented Framework Reuse, *IEEE Computer*, September 1996, pp. 52-61.

[Stamelos 2000] Stamelos I., Vlahavas I., Refanidis I., Tsoukiàs A., Knowledge Based Evaluation of Software Systems: a Case Study, *Information and Software Technology*, 42(5) 2000, pp. 333-345.

[Swanson 1997] Swanson, B.D., MacMagnus, J.G., C++ Component Integration Obstacles, *CrossTalk*, May 1997, Vol.10, No.5, pp. 22-24.

[USC 2000] USC-CSE-2000-501 COCOTS: A COTS Software Integration Lifecycle Cost Model - Model Overview and Preliminary Data Collection Findings, Computer Science Department, USC Center for Software Engineering, available at http://sunset.usc.edu, 2000.

[USC 1997] USC-CSE97 - Center for Software Engineering , COCOMO II Model Definition Manual, Computer Science Department, USC Center for Software Engineering, http://sunset.usc.edu/Cocomo.html, 1997.

[Vidger, Dean 1997] Vidger, M.R., Dean, J., An Architectural Approach to Building Systems from COTS Software Components, The 22nd Software Engineering Workshop, NASA/Goddard Space Flight Center Software Engineering Laboratory (SEL), Greenbelt, MD, December 1997, pp. 99-131

[Vincke 1992] P. Vincke, *Multicriteria Decision Aid*, John Wiley, New York, 1992.

[Voas 1998] Voas, J. Defensive Approaches to Testing Systems that Contain COTS and Third-Party Functionality www.rstcorp.com 1998 In Proceedings of 15th International Conference and Expo on Testing Computer Software, June 1998.

[Voas 1999] Voas, J, Certifying Software for High-Assurance Environments, *IEEE Software*, July/August 1999, p 48-54.

[Voas Charron] Voas, J, Charron F, Miller, K, Tolerant Software Interfaces: Can COTS-Based Systems be Trusted without Them? Available at http://www.rstcorp.com

[Voas Payne] Voas, J, Payne, J, Dependability Certification of Software Components, available at http://www.rstcorp.com

[Voas] Voas, J, Software Component Dependability Assessment, available at http://www.rstcorp.com

[Weiser 1984] Weiser, Mark, Program Slicing. *IEEE Transactions on Software Engineering*, 10:352-357, July 1984.

[Yakimovich 1999] Yakimovich, D., Bieman, J.M., Basili, V.R., Software Architecture Classification for Estimating the Cost of COTS Integration, Proceedings of the 21st International Conference on Software Engineering, Los Angeles, USA, 1999, pp. 296 –302.

[Yeh 1996] Yeh, Y.C., Triple-Triple Redundant 777 Primary Flight Computer, *Proceedings of the 1996 IEEE Aerospace Applications Conference*, Vol. 1, New York, NY, Feb. 3-10, 1996, pp. 293-307.