Rapid Integration Tools For Rapid Application Development

A Case Study on Legacy Integration

Amit Midha
Ravindra Singh
Lakshmi Pratha Hari

Patrick R. Place, Advisor

December 2004

TECHNICAL REPORT
CMU/SEI-2004-TR-023
ESC-TR-2004-023

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Rapid Integration Tools for Rapid Application Development

A Case Study on Legacy Integration

CMU/SEI-2004-TR-023
ESC-TR-2004-023

Amit Midha
Ravindra Singh
Lakshmi Pratha Hari

Patrick R. Place, Advisor

December 2004

Integration of Software-Intensive Systems Initiative

This report was prepared for the

SEI Joint Program Office
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Christos Scondras
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Background and Acknowledgements

The project served as an educational elective that is a requirement for the PDC [Professional Development Center] Scholars at Carnegie Mellon University-West Coast Campus for their MSIT-SE Curriculum.

We would like to thank Patrick Place, Member of the Technical Staff at the Software Engineering Institute, Pittsburgh, for mentoring the research and providing guidance and direction.

We would also like to thank those listed below for their approval and support.

- Dr. Lynn Robert Carter, Principal Fellow, West Coast Campus, Carnegie Mellon University
- Patricia Oberndorf, Director, Dynamic Systems Program SEI
- Dennis Smith, Lead, Integration of Software Intensive Systems Initiative
- Dr. Scott Lewis, Senior Faculty, West Coast Campus

Industry Experts:

- Gerry Miller, Chief Technology Officer, Microsoft Corporation, U.S. Central Region
- Eric Newcomer, Chief Technology Officer, IONA Technologies
- Richard. W. Metz, Director, Process, Architecture and Tools, Boeing Computer Services
- Jim Farrell, Product Management, IBM Rational Software

Abstract

This report investigates the rapid integration tools available in the current market. These tools aid in the rapid integration of software systems and components. The research centers on a model problem that requires such a tool to address legacy integration challenges. The report presents a generic evaluation framework for identifying and evaluating rapid integration tools and an evaluation of three identified tools. This evaluation engaged selected evaluation criteria based on the demands of the model problem. A process reference is also included; this forms the guidelines for identification and evaluation of the tools with respect to other model problems.

1 Introduction

1.1 Purpose and Objective

This project involves the analysis of rapid integration tools available in the market, which aid in rapid integration of software systems/components. The project is centered on a model problem that requires such a tool to address legacy integration challenges. The main outcome of this research includes

- a generic evaluation framework for identifying and evaluating rapid integration tools. The evaluation criteria are geared towards the model problem that belongs to a class of model problems having integration/interoperability as the key concern.
- an evaluation of three identified tools with respect to the evaluation criteria and the model problem which forms the framework for evaluation of tools.
- a process reference to the Integration of Components Certificate at Carnegie Mellon West, which forms the guidelines for the identification and evaluation of the tools with respect to other model problems.

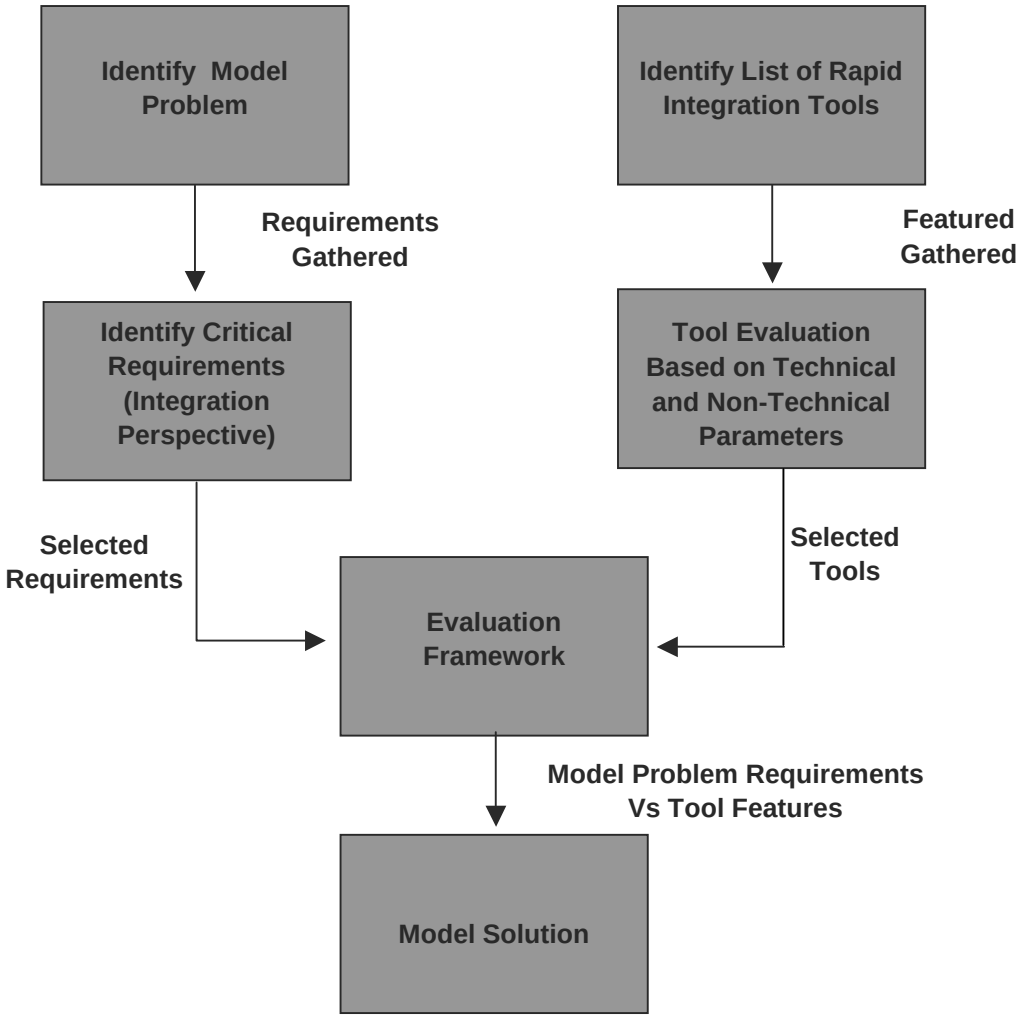


Figure 1: Evaluation Process for the Rapid Integration Tools

The above diagram symbolizes the process followed for determining the evaluation framework. The team identified the model problem and the list of tools, quantified requirements from the model problem description, came up with a tool evaluation report and finally came up with an evaluation framework. The figure below illustrates the evaluation framework defined for the tools that have been selected to satisfy the requirements specified as critical by the model problem. In both the preceding and following diagrams technical factors are those directly related to the model problem and are derived from both functional and non-functional parameters. The non-technical parameters are softer, but no less important, factors such as the quality of vendor support or market share of the tool.

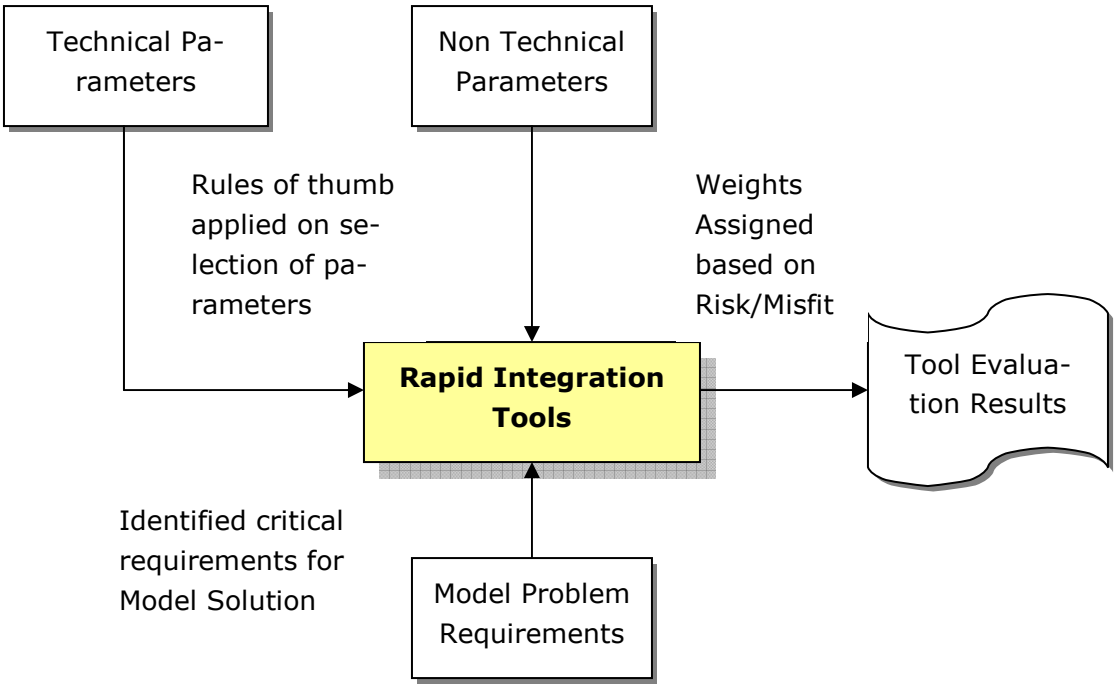


Figure 2: Evaluation Framework

1.2 Project Requirements

As demand for new functionality grows and new systems to fulfill it go into operation, the need to integrate new systems with existing systems has increased. The development of resulting extended systems is frequently based on the integration of existing components, leading to demand for new integration tools. Modern integration tools all promise the ability to integrate components more quickly and cheaply than traditional technologies.

The project described here is aimed at surveying the field of rapid integration tools with a view to informing the reader on how to select among the choices. The task was divided into the following steps.

Survey and classify the tools.

The first step was to identify the tools that claim to provide rapid integration. Since these tools were known to provide a wide range of services, the identification also required the development of a classification scheme for characterizing the various types of tools.

Deliverables: 1) list of rapid integration tools 2) classification scheme 3) classified list of tools

Evaluate the tools using a model problem

We expected that one or more of the classifications would contain a number of interesting rapid integration tools. We chose a problem typical of the type of integration the tools were designed for and applied a number of tools to that problem.

Deliverables: 1) preliminary evaluation scheme 2) model problem definition 3) reports detailing evaluation of tools' applicability to the model problem

Develop and document general evaluation criteria

Following the evaluation, the final step was to refine the evaluation criteria and document the refined versions. The purpose was the creation of an instrument that would assist a developer in choosing the "right" rapid integration tool.

Deliverable: documented evaluation criteria for rapid integration tools. Depending on time, steps 2 and 3 may be repeated within another classification.

1.3 Project Plan and Tracked Report

We followed a simple phased approach for executing the project with each phase divided into tasks and related deliverables. Each deliverable is considered as a milestone and is derived from the initial list provided by the SEI. Since the project is exploratory, it does not follow any standard software development life cycle, but we followed software engineering principles from the start. We used the work breakdown structure (WBS) and effort available from the elective to estimate a completion date based on a given start date. The project ran over schedule perhaps indicating the problem of using available effort as an artificial constraint on work to be performed.

1.4 Structure of the Document

This report is organized into three major chapters.

Chapter 1: **Introduction to the Technical Report** presents the purpose and objective of the project, the project description, background, requirements, project plan and tracked report and the structure of the document.

Chapter 2: Identification and Classification of the Tools describes the list of tools identified as the rapid integration tools and the evaluation framework applied to them for selection to work with the model problem. The classification parameters that support the evaluation framework are the technical and non-technical parameters.

Chapter 3: Model Problem and Tool Implementation explains the model problem selection and identification of critical requirements as well as application of the tools and their assessments.

Chapter 4: Conclusions documents the lessons learned arising from the use of the specific tools. Additionally, questions for future research are listed as are some concluding comments on the development of the evaluation framework, including factors to consider before and after applying the evaluation framework.

Appendices feature detailed descriptions of the tools evaluation, model problem, and other estimations.

2 Identification and Classification of Tools

2.1 List of Tools

The first step in our approach is to identify the tools used for rapid integration. We discovered little difference between integration tools and rapid integration tools.[1] We identified 11 rapid integration tools designed for the rapid integration of applications from existing components.

1. Pervasive Data Junction
2. RoughWave's LEIF
3. IBM Rational Rapid Developer
4. Microsoft SQL Server
5. Host Integration Server
6. Microsoft BizTalk Server
7. IBM WebSphere Business Integration
8. Artix Relay, Encompass and Mainframe
9. PiiE Smart Client and Fusion Server
10. InterSystem Ensemble
11. Jboss

For the above-listed tools, we collected information about their vendors' name and features. See Appendix A: Tools Studies and Analysis for more information.

2.2 Tool Selection Criteria

The identified tools were filtered based on the model problem that will be described in Section 3. Since 11 tools seemed too many for starting the evaluation process, a short list was created based on the following criteria.

1. Tool should be capable of solving a wide range of Enterprise Application Integration problems, especially the Legacy Integration problem.
2. Tool is able to provide communication between Java and C++ Components.
3. Tool has solid success stories associated with it.

[1] *Rapid* is a concept that depends on the user's context. In some contexts, six months may be considered rapid and in others, six hours could be too long. The tools themselves are, essentially, the same and a better question is whether the integration tools speed the integration sufficiently to both produce timely applications and cost less than not using the tools.

4. Tool has been in market for at least two to three years.
5. An evaluation version of the tool is available and the evaluation period is sufficient to evaluate the tool.

Through application of these selection criteria the above list of 11 tools was short-listed to 3. The three tools were

1. IBM Websphere
2. IBM Rapid Developer
3. LEIF (Light-weight Enterprise Integration Framework)

2.3 Classification Parameters

The Classification Parameters used to evaluate tools can be technical or non-technical in nature. The functional and non-functional requirements of the model problem form the technical parameters. Powell and colleagues observed] that apart from these technical parameters, some non-technical parameters arise from other business-oriented issues, such as cost and vendor, which play an important role in the selection of a tool for rapid application development [Powell 97].

We identified 16 parameters (5 non-technical parameters and 11 technical parameters) for classifying rapid integration tools. The table below gives a brief description of these parameters.

Table 1: Classification Parameters - Technical and Non-Technical

#	Parameter	Description
	Non-Technical Parameters	
1	Business	market price of the tool return on investment (ROI) of the tool (based on cost of the tool compared to the estimated cost of manually integrating the components) foreseen risk in using the tools (lifespan of the tool, ease of use, change frequency and so on)
2	Evaluation-Specific	project life cycle in which the tool can be used (software configuration, project planning, oversight and tracking and so on) comparative report of other tools in similar domain
3	External References	visibility and popularity of the tool in the market
4	Vendor Support	quality and cost of the vendor support access to architecture and design aspects of the tool

5	Tool-Specific	integration with other tools and platform it can support solution space the tool belongs to with respect to the problem (domain specific) reliability of the tool and the vendor maturity level based on industry standards skill set required to operate the tool tailorability of the tool extent to which data generated by the tool (performance logs and so on) is configurable. number of well-defined components that can be used separately performance of the tool interactivity of the tool sufficiency of documentation (user manual, installation guide and so on) bundled with the tool degree to which data generated by the tool can be used by other tools
Technical Parameters		
6	Security	support offered by the tool for developing secure or safety critical systems
7	Correctness	capability of the tool for producing accurate results
8	Availability and Robustness	capability of the tool for surviving system failure
9	Ease of Use - Usability	degree of learning curve associated with the tool
10	Downward and	portability of applications developed using one version of the

	Upward Compatibility	tool to higher and/or lower versions of the same tool
11	Flexibility	capability of tool for operating in different operating system environments
12	Product Performance	response time of the tool
13	Tailorability	customizability of tool for meeting user-specific requirements (user interface, enabling/disabling of features, enhancing the tool by adding plug-ins, and so on)
14	Service Implementa-tion Coverage	technical support/licensing cost associated with the tool
15	Interoperability	capability of tool to interoperate with other systems
16	Testability	ability to test the functionality of the tool

Each rapid integration tool is analyzed based on the classification parameters above; it is rated on a scale of 0 to 10, depending on how well it satisfies the parameters. The detailed evaluation of the tools is found in Appendix A: Tool Studies and Analysis.

Assigning values to parameters while analyzing any tool may be tricky. Different individuals may come up with different analysis results. In order to avoid this, we defined some rules of thumb, shown in Table 2 below. These rules are so generic that they can be used to analyze any rapid integration tool.

Table 2: Weights Assigned to Parameters Based on Rules of Thumb

Weights				
Parameters	**0**	**1 to 3**	**4 to 7**	**8 to 10**

Business	Cost is very high and has special installation requirements (e.g., specific operating system, runtime libraries).	Cost of the tool doesn't support the ROI; there are frequent changes to the tool and the cost of learning the tool is high.	Cost of the tool supports the ROI; there are frequent changes to the tool and the cost of learning the tool is high.	Cost of the tool supports the ROI; there are two or three releases of the tool a year and the cost of learning the tool is low (e.g., because of extensive Graphical User Interface).
Evaluation-Specific	Tool is single-user and supports no integration with organization's software development life cycle and other tools.	Tool assists in the collaborative development but cannot be integrated with the organization's software development life cycle.	Tool supports collaborative development by team, has its own configuration management and project management utility but cannot be integrated with other tools.	Tool supports collaborative development by team, supports configuration and project management and can be integrated with other tools to expand its current capabilities.
External References	Tool has recently launched in the market.	Tool has received average response from the user, has been in market for one to two years, and a similar tool by leading vendors (e.g., Microsoft, IBM) is available in the market.	Tool has been used by several large organizations, has very few competitors, and has several success stories associated with its use.	Tool has been in market for four or more years, owned by software market leaders like IBM or Microsoft, used by large organizations, and has many success stories associated its use.
Vendor Support	Tool has no customer support.	Tool has limited customer support through mail and telephone conversations only.	Tool has good customer support through online discussion forum, mail and telephone conversations. There is immediate response to queries posted to Customer	Tool has effective customer support through online discussion forum, email, and on-site consultation. Response is immediate to queries posted to

			Support Center.	Customer Support Center.
Tool-Specific	Tool doesn't support integration with other tools.	Tool supports integration with two or three other tools and has complex integration process.	Tool supports integration with software configuration management tools, testing tools, application servers, and so on, and integration process is moderate and requires manual settings.	Tool supports integration with software configuration management tools, testing tools, application servers and so on, and integration process is easily performed via wizards. Tool supports custom development to enhance its features and usability.
Security	Tool doesn't provide any features to aid in the implementation of security mechanism (encryption, authentication, authorization etc.)	Tool supports few standard security mechanisms like encryption and authentication.	Tool supports most security mechanisms currently used in the market and but doesn't support any custom development of security mechanisms.	Tool supports most security mechanisms currently used in the market and also supports custom development of security mechanisms.
Correctness	Tool has no utility for testing the application developed by it.	Tool supports limited testing for the application developed.	Tool supports standard testing of the application developed through testing utilities bundled with the tool.	Tool performs validation at every step while developing the application. Also supports integration of other testing tools (e.g., third party application servers) to verify the correctness of the application created.

Availability and Robustness	Tool doesn't save the data should system failure occur.	Tool backs-up the application data, which it uses to recover from system failure.	Tool backs up the application data and provides automatic recovery from any type of system failure (sudden shutdown of the desktop, sudden crashing of desktop etc.).	Tool takes backs up of the application data, provides automatic recovery from any kind of system failure (sudden shutdown of the desktop, sudden crashing of desktop etc.) It also supports restoration points so that user can switch between restoration points.
Usability	Tool has non-GUI interface and no feature to automate the execution of tasks or operations.	Tool has GUI Interface, but requires a lot of navigation across the screen to perform any operation.	Tool has GUI Interface with minimum overhead of navigation while performing any task. Also it provides quick links to commonly used operations.	Tool has effective GUI Interface which not only eases in performing tasks but also reduces the learning curve associated in performing any task. Also tool supports has wizards to guide operations step by step and single-click execution of any operation.
Upward and Downward Compatibility	Application created by the tool is not supported by earlier or newer versions of the same tool.	Application created by the tool can only be exported to new version under a few circumstances.	Application created by the tool can be exported to new version but requires manual changes in the configurations.	Application created by the tool can be exported to new versions. All the necessary changes are automatically handled by the tool itself.

Flexibility	Tool has a specific Operating System requirement.	Tool is available for several Operating System environments and does not support portability of applications between platforms.	Tool is available for several Operating System environments and supports portability of applications between platforms.	Tool is available for several Operating System environments, supports portability of application between platforms and has interfaces for communication between instances running on different platforms.
Product Performance	Time taken by tool to perform any operation is more than six minutes and the system hangs up while performing any operation.	Time taken by tool to perform any operation is four to six minutes and 80% of the time tool performs its operation successfully.	Time taken by tool to perform any operation is two to four minutes and 90% of the time tool performs its operation successfully.	Time taken by tool to perform any operation is between two to four minutes and 100% of the time tool performs its operation successfully.
Tailorability	Tool doesn't allow user to configure / enhance its features.	Tool allows user to configure / enhance its features by installing plug-ins or add-ons available from the tool vendor only.	Tool allows user to configure / enhance its features by installing suitable plug-ins or add-ons available from any vendor.	Tool allows user to configure / enhance its features by installing suitable plug-ins or add-ons available from any vendor or by programming the tool itself.

Service Implementation Coverage	Tool has stringent licensing policy and does not promote evaluation copies to experiment with the tool. Also it has high licensing cost and purchasing a new license is almost equal to the cost of the tool itself.	Tool promotes evaluation copy but the period is not sufficient enough to evaluate the tool. Also the licensing cost is very high.	Tool promotes evaluation copy with adequate customer support for any issues that arise during the evaluation period but the evaluation period is not sufficient enough to evaluate the tool. The licensing cost is nominal.	Tool promotes evaluation copy with adequate customer support for any issues that arise during the evaluation period and also the evaluation period is sufficient enough to evaluate the tool fully. The licensing cost is nominal.
Interoperability	Results produced by the tool cannot be exported to other formats (Word document, html, jpeg, etc.) and it doesn't have any interface to communicate with other tools.	Results produced by the tool can be exported to other formats (Word document, html, jpeg, etc.) but it does not support inter-tool communication.	Results produced by the tools can be exported to other formats (Word document, html, jpeg, etc.) and it supports inter-tool communication.	Results produced by the tools can be exported to other formats (Word document, html, jpeg, etc.) and it supports inter-tool communication. Also tool can produce results that can be ported to any platform.
Testability of Output	Tool doesn't validate the output produced.	Tool does minimal validation and the accuracy of the result is about 70%.	Tool does validation of the results and the accuracy of the result is about 80%.	Tool does validation throughout and the accuracy of the result is 100%.

2.4 Tool Evaluation

The three short-listed tools were evaluated for these technical and non-technical parameters. See Appendix A: Tool Studies and Analysis for the Tool Evaluation Report.

The following graph shows the summary of the parameter values for each tool.

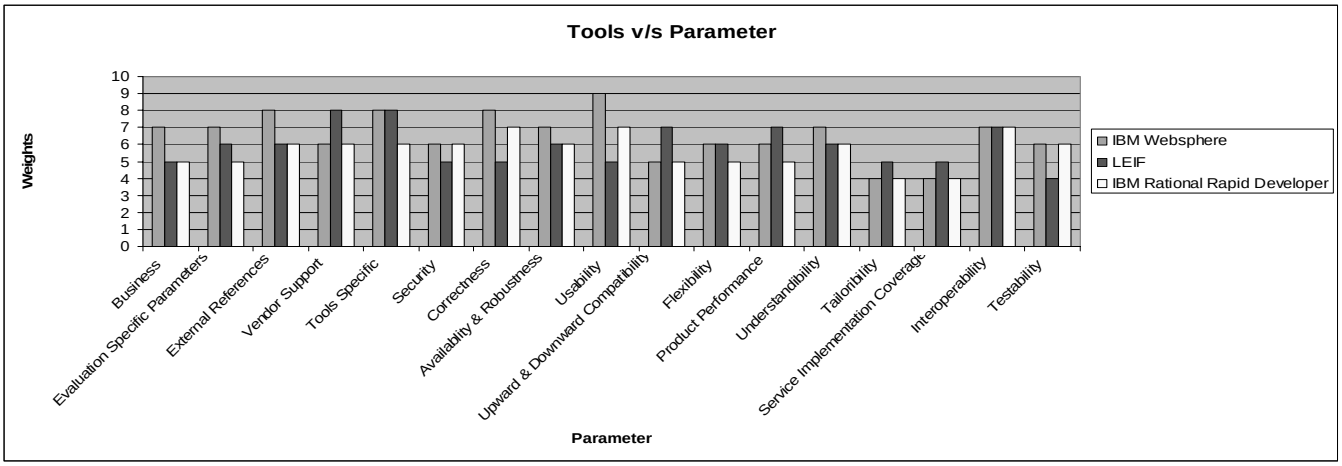


Figure 3: Graph Showing Characteristics of the Three Tools Selected

The above graph shows the comparative analysis of three rapid integration tools—IBM Websphere, IBM Rapid Developer, and LEIF—with respect to technical and non-technical parameters. The X-axis of the graph lists the parameters (technical and non-technical) and the Y-axis represents the value assigned to these parameters from 0 to 10. Such graphs can be used to prioritize the list of identified rapid integration tools. A similar graph including model problem requirements in terms of parameters can help us to identify which tool is the best fit for the model problem.

3 Evaluation Using a Model Problem

3.1 Purpose

This section describes the evaluation of the tools using a selected model problem. The model problem selection criteria, the description of the model problem, and the evaluation of the tools are highlighted. In describing the model problem selection criteria we also explain the sequence of steps we followed in making our selection. We briefly describe the model problem with respect to the non-functional requirements and the problem statement. Finally, we explain the tool evaluation using the model problem; here we've applied the structure of the selected model problem as described by Kurt Wallnau in *Building Systems from Commercial Components* [Wallnau 02]. The assessment results obtained from this evaluation show the extent to which the tools satisfied the post evaluation criteria and the problem's non-functional requirements.

3.2 Model Problem Selection

3.2.1 Model Problems

We found three potential model problems (descriptions follow); each problem is appropriate to a particular type of integration.

Web Service Enablement

A company uses the enterprise integration technology as well as XML technology to make customer account information accessible via a Simple Object Access Protocol (SOAP)-based Web services interface.

The implementation of this solution requires retrieving and combining information from two source applications. The first application is a custom CORBA system that provides historical customer support information. The second application is Siebel, which provides customer purchase information.

Legacy Integration

Bond traders working online must send prices for a large number of bonds to several different trading venues, each with its own user interface; this disrupts the workflow of their bond trading desk.

The system solution should minimize the minutiae of pricing all traders' bonds and provide an advanced analytic functionality, specific to the bond market, in a single encapsulated user interface. This system would utilize legacy components on the server side.

Business Application Integration

Three companies use different business processes involving different sets of assumptions. Middleware must be utilized as the integration point for communicating among the business processes.

3.2.2 Problem Selection

The following five steps were followed

Step 1: Identify the problem that must be solved through integration.

In this case we identified the following types of integration based on our prior knowledge.

1. Middleware Integration
2. Service Oriented Integration
3. Web Service Integration
4. Legacy Integration
5. Enterprise Application Integration

Step 2: Based on research on the various application integration types we chose three that would provide the best opportunity for using the tools that we have selected.

1. Legacy Integration
2. Business Application Integration
3. Web Service Enablement

Step 3: Analysis and study on the three model problems were made based on answering the following questions.

1. Which problem gives a way to integrate two different technologies?

Of the three problems presented, we found that the Trading Bond System required a solution that would integrate components built using two different technologies (in this case programming languages), as evidenced by its case study report:

Traders needed a very responsive application on both Windows NT and Solaris workstations. Therefore, we decided to implement the client application as a Java thick client because of its platform independence and its ability to quickly respond to user input and market data. On the server side, we are inheriting legacy C++ components that our system will utilize [Simon 03].

The Trading Bond system meets the criterion of providing a way to integrate two technologies. The solution requires integration of a Java based component and C++ based components, which can be done by building a pair of Java gateways to communicate with the C++ server-side components [Simon 03]. Details of the Trading Bond System case are available at http://www.eaipatterns.com/BondTradingCaseStudy.html.

2. Which problem is faced in the real world often?

A demand exists in government and industry for existing systems to be updated or integrated with the current technology. Many applications require technology independence and interoperability with various applications that were developed using different technologies. Moreover, in the current industrial scene there is a drive toward production of tools for the integration of Java-based components and C++ based components. The Trading Bond System is therefore quite typical of real-world scenarios.

3. Which problem is very specific and solvable within the specified time constraint?

We found that legacy integration for the Trading Bond System best met these criteria. We were able to download two specific components provided by Dukascopy, an online trading application, which mapped with the Trading Bond System scenario. This bode well for solving the problem on time.

We did not find the same with the customer account information problem to be solved through Web Service enablement: Here the application was generic and we were not able to find the specific attributes to be addressed or specific requirements to be fulfilled. We realized that it might take considerable time to establish which tools were needed. This presented a problem, given time constraints and resource availability. We did not have enough time to create a simulation of the CORBA System and the Siebel system.

The problem that might have been solved through Business Integration did not meet this criterion. This involved three companies who required communication among their different business processes. The business processes to be integrated were not well defined or specific. The time required for creating simulated processes and then integrating them did not meet our constraints.

4. Which of the other classes of integration does this model problem address?

The chosen problem could also be used to assess service-oriented integration insofar as it is reasonable to treat the problem components as services. Additionally, the middleware, application and Web-based integration classes could be addressed by the Trading Bond system.

Step 4: Identify the model problem that fit into the evaluation framework based on the above identified questions.

Step 5: Having identified the model problem to be solved, we now present details regarding the Trading Bond system relevant to further application development using the integration tools.

3.3 Model Problem Description

The Trading Bond system best met the problem criteria and highlights the necessity of integration, for communication purposes, between various components with various user interfaces and communication protocols. The following table shows an analysis of the original problem statement into a problem description describing the various actors and also constraints on the solution.

The problem of	bond traders to send prices for a large number of bonds to several different trading venues, each with its own user interface
affects	bond traders
the impact of which is	to disrupt the streamline of the workflow of their bond trading desk
a successful solution would be	a bond pricing system to minimize the minutiae of pricing all of their bonds combined with advanced analytic functionality specific to the bond market in a single encapsulated user interface.

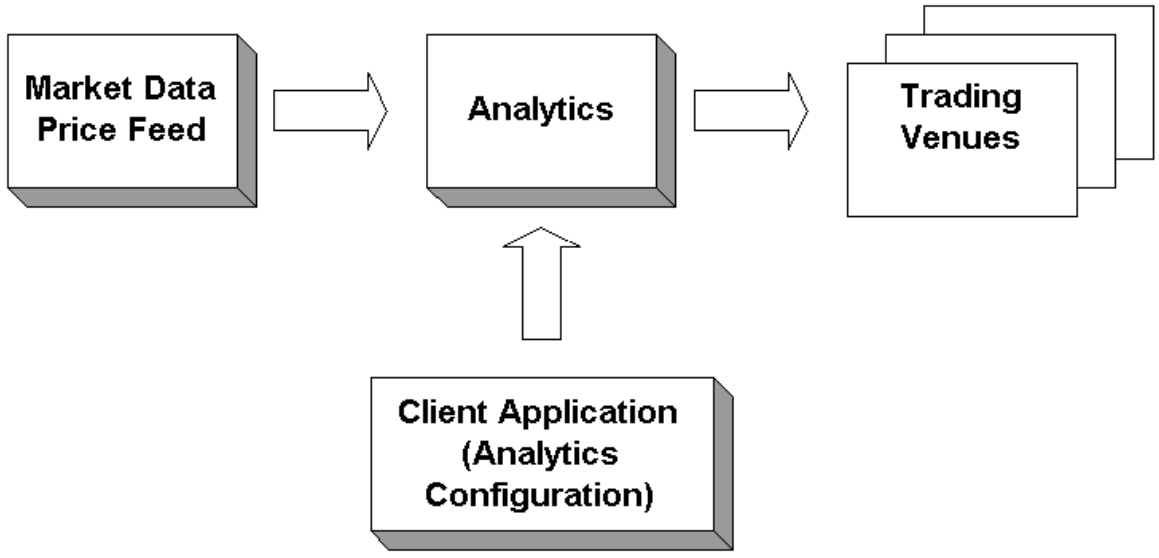


Figure 4: High-Level Context Diagram of Trading Bond System

First, market data comes into the system. Market data is data regarding the price and other properties of the bond representing what people are willing to buy and sell the bond for on the free market. The market data is immediately sent to the analytics engine that alters the data. Analytics refers to mathematical functions for financial applications that alter the prices and other attributes of bonds. These are generic functions that use input variables to tailor the results of the function to a particular bond. The client application that will run on each trader desktop will configure the analytics engine on a per trader basis, controlling the specifics of the analytics for each bond the trader is pricing. Once the

analytics are applied to the market data, the modified data is sent out to various trading venues where traders from other firms can buy or sell the bonds.

The following are some of the non-functional requirements that the system should address, in order of priority. The scope of this model problem extends to only the highest priority quality attributes selected by the team.

Integrability:

"On the server side, we are inheriting legacy C++ components that our system will utilize."

The system should be integrable with the legacy C++ components which forms the Market data feed pricing subsystem and the thick client application which will be a Web-based thick Java client.

Performance:

"**Traders** need a very **responsive** application"

Two attributes of performance are essential to this responsiveness.

1. Scalability: Measured as the number of traders who will be accessing the system and the system's capability for accommodating them.
2. Response Time: The system should be able to respond to the user without significant delay [here we say less than 5 sec, assuming that it is a Web-based application]

Portability:

"Traders need a very responsive application on both Windows NT and Solaris workstations."

The application should be portable to any platform based on the demands of the trader's needs.

The quality attribute that will be addressed in this execution of the model problem is highlighted in Table 3.

Table 3: Prioritized List of Quality Attributes

Quality Attribute	Prioritization	Rationale
Integrability	1	Use of the rapid integration tools to integrate legacy systems with Web-based application. In this case we are integrating the C++ legacy system with the Web-based Java Client.
Performance	2	We address the response time specific to this application as determined by the team for a responsive application.
Portability	3	Client application is developed in Java which automatically supports platform independence.

3.4 Tool Evaluation using Model Problem

The following diagram shows the elements of the assessment. The trading bond problem is used as the model problem and the criteria coupled with the design question lead to the tool assessment, and COTS components forming the trading bond system.

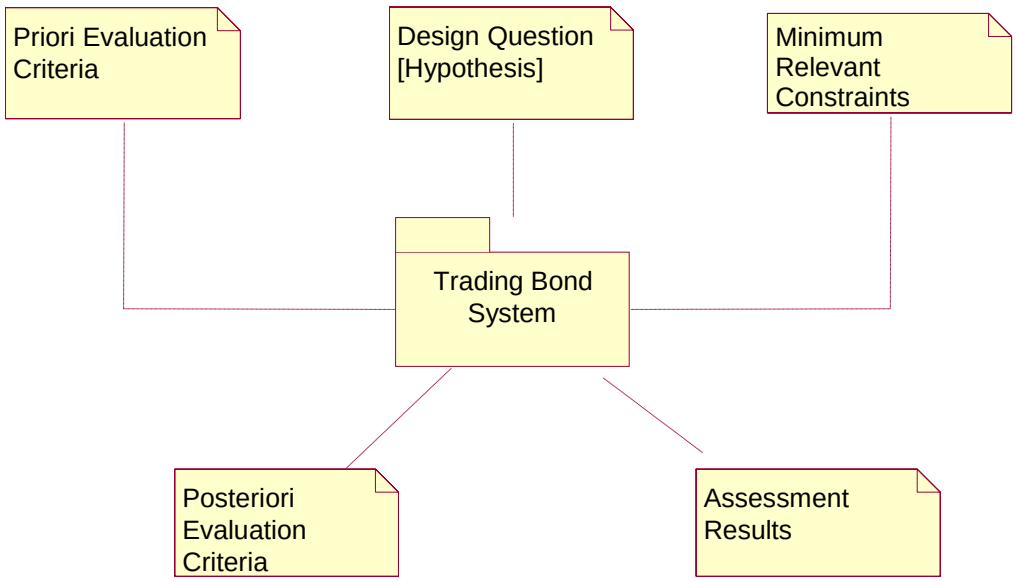


Figure 5: Structure of the Model Problem

Design Question:

This is the initiating element of the model problem.

In this case study of the Trading Bond system, the design question is

Is it possible to integrate the Java and legacy C++ components that are obtained off the shelf from the Dukascopy stock quote Web site, using the rapid integration Tools?

Hypothesis: A wrapper component [integration point] that provides the communication between the Java thick client and the legacy server side C++ component can be constructed using the rapid integration Tools.

Priori Evaluation Criteria:

These are the criteria to be satisfied by the model solution. They were obtained by analyzing the application specifics given in the case study. They are centered on integration techniques and use of integration tools. These evaluation criteria are formulated based on the hypothesis that we have addressed related to the model problem. These criteria help in defining with the Standard's compliance that the tools must meet in order to satisfy the requirements.

Criterion #1: A Java to C++ translator is required. Java thick Client talking with C++ Legacy Servers

Criterion #2: Messaging Bridge to support the communication between cross-language applications [C++ and Java]

Criterion #3: Single point of access is required to communicate with the gateways of the legacy servers.

The criteria form the model problem requirements for the integration implementation using the tool. Thus according to Criteria #1, #2, and #3, the tool should be able to provide a communication mechanism, a messaging bridge and a single point access between the Java and C++ components. In this case the tools IBM WebSphere and the LEIF help in achieving these developments.

Minimum Relevant Constraints

The following constraints are based on what is feasible to provide in the model solution to address the above mentioned priori criteria:

1. This is a short-term project that involves rapid development; hence the use of rapid integration tools to create the Java to C++ translator, messaging bridge and single point access mechanism, which are the priori criteria of the model problem.
2. The business processes of the model problem are not a focal point, since they are addressed by the off-the-shelf components that are downloaded from the Dukascopy site.
3. The tool selection is restricted to the major functionalities provided by the tool with respect to the model problem's priori criteria.

The development and deployment environment are the same; hence the performance of the model solution is constrained.

Model Solution: Trading Bond System

A simple solution that clarifies how the model solution was implemented is provided below:

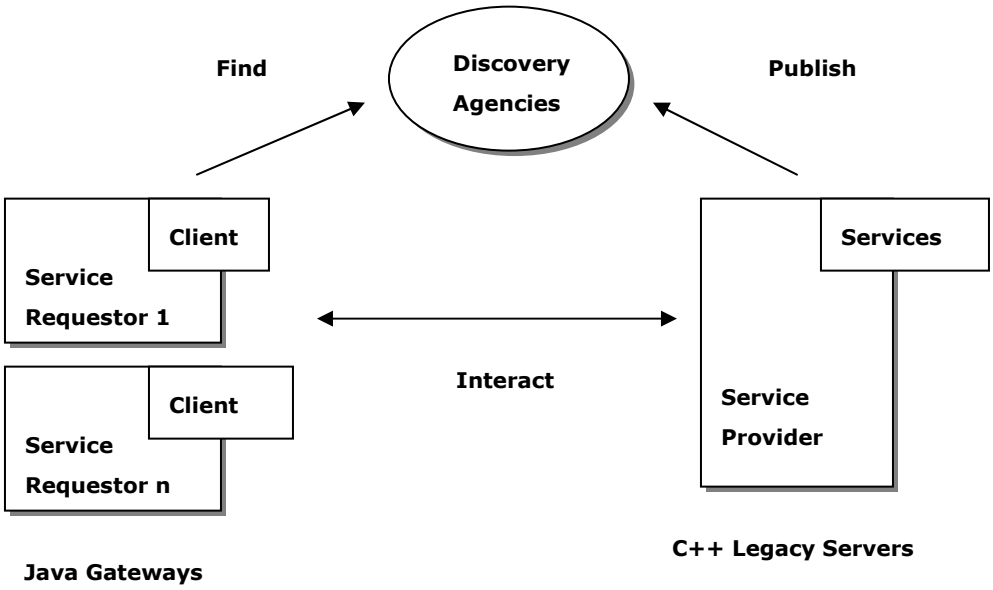


Figure 6: Model Solution—High-Level Context Diagram

The Java Gateways are considered the Service Requestor and the Java Web Services are implemented using the IBM WebSphere. These Java Web Services are the Client side application required for communicating with the C++ Legacy servers, in this case the Market Data Feed Component obtained from the Customized Dukascopy Data Feed (CDDF) http://www.dukascopy.com/english /ddf_main/ . The C++ Legacy Servers are the service providers. The inner workings of the C++ Component and the Java Component were not considered; it was the integration between these components that was implemented using the Tools. The Discovery Agencies used were the UDDI Services, which were automatically set in the IBM WebSphere tool.

The System uses the simple publish-subscribe model for the implementation of the integration through discovery agencies and SOAP is the communication protocol that establishes the interaction between the two Web services.

Posteriori Evaluation Criteria

Criterion #1:
Installation and development environment for the identified solution tools are in place.

Criterion #2:
The off-the-shelf components architecture and design maps with the model problem requirements.

Criterion #3:
The integration of the two COTS components is accomplished using the rapid integration tools

The above criteria help in evaluating whether the tools are able to meet the requirements of the model problem and whether they are able to conceptualize the hypothesis that has been defined for this model problem.

Assessment Results

The assessment results are enumerated based on the following factors:

1. risks encountered and mitigated while using the contingency approach
2. the size, effort, and cost variance involved when using the rapid integration tools and when not using the rapid integration tools
3. product outcome explaining the steps that brought success and those that resulted in failure in the development using rapid integration tools

These assessments help in evaluating the tools as they apply to the model problem. In this case they are restricted with respect to the legacy integration of cross-language platforms.

Risks Encountered and Mitigated

The following table describes the major risks that we encountered and mitigated through contingency plans.

Table 4: Top Three Risk List

No.	Risk	Risk Management Strategy						Status
		Impact	Occurrence Probability	Exposure (Rank)	Mitigation Activities	Contingency		
						Trigger	Activities	
1	Tools identified are not suitable for solving the model problem	90%	.9	1	Collect the tools based on the model problem's critical requirements	The tools are not able to produce a mechanism that solves the communication between Java Gateways and the C++ Legacy Server.	Determine which tools support the communication mechanism. In this case C++ Web Services are created using the Apache Axis C++. The integration of the C++ Component with the Java Component is accomplished via the LEIF.	Close

No.	Risk	Risk Management Strategy						Status
		Impact	Occurrence Probability	Exposure (Rank)	Mitigation Activities	Contingency		
						Trigger	Activities	
2	Learning curve	60%	.8	2	Estimate the effort and execute a short-term plan for learning only the required tools.	Understanding the process of using the tool for the specified model problem.	Approach the technical support for the specific tool or the interactive manual for the understanding of the tool.	Close
3	Installation and troubleshooting	80%	.7	3	Test the development environment using Evaluation Software and samples.	Installation is problematic or the tool is unable to produce the required functionality.	Use separate testing machine for testing the installation and run sample problems that are related to the model problem requirements	Close

Size, Effort, Cost Variance

The following table explains the size, effort and cost variance. The size, effort and cost are estimated using the COCOTS calculator; this includes estimates of the glue code to be written and calculation of the respective effort and cost for writing the glue code. Using the actual size, effort, and cost recorded while doing the development, the variance is calculated as shown below:

Table 5: Variance Calculator

Factor	Estimated without using the rapid integration Tools	Actual after using the rapid integration Tools	% Variance (Estimated-Actual)/Estimated *100)
Size (KLOC)	1.01	0 [Source Code auto generated]	100
Effort (Person Months)	17.63	8	54.6
Cost (in $ excluding software costs)	123,403	55,996	54.6

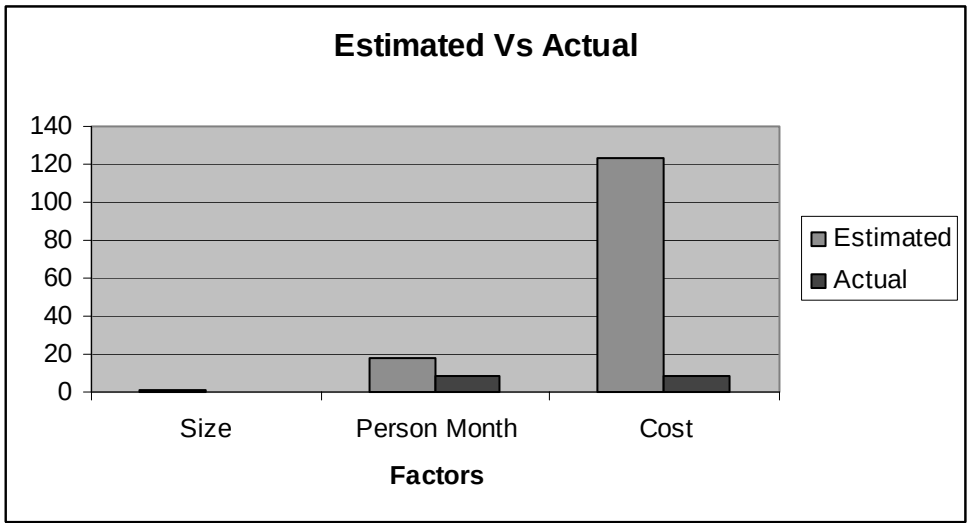


Figure 7: Graph that Explains the Estimated vs. Actual Effort and Cost

Product outcome

The product developed using these tools should have adhered to the a posteriori evaluation criteria that we arrived at and also the non-functional requirements of the model problem.

Table 6: Posteriori Evaluation Criteria Satisfied by the Tools

Criterion #	Description	Observation
1	Installation and development environments are in place for the identified solution tools.	Yes. All three tools satisfied this criterion.
2	The off-the-shelf components architecture and design satisfy the model problem requirements.	The components do not map exactly with respect to the implementation model as required by the model problem. They satisfied the functional requirements.
3	The integration of the two COTS components is accomplished using the rapid integration tools.	The tools, especially IBM WebSphere, LEIF, and Apache Axis C++, were used for creating and integrating the Web services of the COTS components.

The product outcome is also validated when the non-functional requirements are satisfied by the various tools.

Table 7: Tools Observations Conforming to Non-Functional Requirements

Requirement	**LEIF**	**IBM Websphere**	**IBM Rapid Developer**
Integrability	Provides services for integration rather than integration itself and is limited regarding C++ Technology	Provides integration capability and is limited regarding Java Technology	Integration capability is lower and is limited regarding Java Technology
Performance (development time provided by the tool, not involving the prerequisites)	Simple interface with fewer inputs and quick response (2 minutes)	Requires knowledge about Web services and complex user interaction and is highly responsive (4 minutes)	Ease of use, and good user interface, and good response time (3 minutes)
Portability (based on the platform independence of the tool)	Portability is very possible (able to create services for various operating systems).	Portability is not possible. Caters to only J2EE applications / middleware applications	Portability is supported to a limited extent.

4 Conclusions

In this section, the lessons learned from the highlights and lowlights of the whole research work have been documented. Additionally, we suggest some directions for potential future work based on this investigation.

4.1 Lessons Learned

What Went Right

1. Being able to download evaluation copies helped in the installation and testing of the tools.
2. The required additional software necessary for the installation and configuration of the selected tools (for instance, LEIF requires VC++) was provided by our university.
3. The creation of services out of the COTS components took almost no time when the tools were used. (The user should be aware of the component and the business logic required to create a service from that component.)
4. IBM WebSphere proved to be a highly interactive tool which enhanced the usability and intelligibility of the feeder component (Java) and was able to generate the Web services from these components in just 4-5 minutes.
5. The Communication between the two components using the SOAP mechanism was successfully completed using the IBM WebSphere.

What Went Wrong

1. Expiration of the evaluation copies often forced us to change machine configurations and settings in the development environment.
2. The COTS component was revised and is no longer freely available, thus this experiment isn't freely repeatable.
3. No configuration management of source files is maintained due to the auto generation of the source code by the tools.
4. We could only run the application in the version of the evaluation copy that created it. Running it in a different version required extra effort and time for reconfiguration and caused data loss.
5. LEIF was unable to generate the WSDL file for the C++ Component, so it involved extra effort to find an alternative to do the same. [This was due to the incompatibility in the versioning of the source code of the Market Data Feed Component (C++).]

4.2 Future Directions of the Research

In this section we want to highlight the Evaluation Framework's applicability to the other model problems and tools by answering the following questions.

- Is the framework applicable to all tools and all model problems?
- How much time is needed to modify your framework when you must support multiple model problems?
- How much effort is needed in terms of searching for technologies and characterizing the model problem different ways?

4.3 Remarks

While the development of the evaluation framework took more time than expected, we believe that the result is worthwhile. The framework, without change, can be used for a significant number of similar evaluations and, with minor change, could be used for a wider range of problems. Further, the evaluations contained herein demonstrate that it is possible to use the framework to distinguish between tools.

The difficulties we had with the various tools suggest that, although rapid integration technologies are being widely hyped, in practice the tools still leave something to be desired. While it is possible to use the tools to integrate legacy components more efficiently than without the tools, the difficulties suggest that more work remains to be done on the tools themselves (as well, perhaps, as the target environment of Web services).

Appendix A Tool Studies and Analysis

List of Tools and Tool Information

The information provided here comes directly from online literature provided by the vendors.

#	Name	Description	Vendor	Features Provided	References
1.	IBM Rational Rapid Developer	IBM Rational Rapid Developer is a single, integrated application development environment that combines model driven development, architected Rapid Application Development (RAD) techniques, and automated construction to develop, integrate, deploy, and maintain high-quality, n-tier business applications—without most of the complexities of the underlying technology platforms	IBM Rational	1. provides single seamless environment for rapid design, integration, construction, and deployment of business applications and portals 2. leverages mainstream development skills on complex n-tier development projects 3. focuses development efforts on high-value business requirements—not infrastructure coding 4. accelerates application and portal delivery through architected RAD techniques such as reverse-engineering, rapid user interface design, functional prototyping, automated n-tier construction, and deployment features 5. simplifies legacy integration 6. creates agile, standards-based applications 7. provides a visual model-driven and point-and-click environment	http://www-306.ibm.com/software/awdtools/rapiddeveloper/ http://www-306.ibm.com/software/awdtools/rapiddeveloper/features/index.html

				8. provides automatic construction and hot deployment of applications and portlets from models 9. supports mainframe and relational database connectivity and CICS visual and non-visual transaction integration 10. XML message mapping	
2.	Artix	Artix is family of service-oriented integration products that will renovate existing IT assets and consolidate legacy middleware to reduce the complexity and cost of IT operations. At the same time, it reduces vendor dependencies and enhancing future IT innovation. Following are the 3 Artix products. Relay Encompass Mainframe	IONA Technologies	Artix Relay 1. supports a wide array of middleware formats, including SOAP and IIOP 2. supports the most popular middleware technologies, including Web services, CORBA/IIOP, WebSphere mq (mqseries), bea tuxedo, TIBCO Rendezvous 3. extends security, transaction, and routing features of enterprise middleware across middleware boundaries Artix Encompass 1. adds enterprise features to Web services integration solutions, including a. secure Web services a. stateful Web services b. transactional Web services c. multi-transport Web services Artix Mainframe 1. high-performance—shatters the misconception	http://www.iona.com/products/artix/ http://www.iona.com/devcenter/artix/ http://www.iona.com/whitepaper.htm#artix

				that Web services cannot be fast 2. automated—creates new Web services in a few minutes, start to finish 3. secure—integrated with SAF for host authentication and authorization 4. standard—WSDL interface definitions can be used by many third-party client development tools	
3.	Lightweight Enterprise Integration Framework (LEIF)	Rogue Wave's LEIF is a complete best-of-breed solution framework for developers needing to enable their applications for communication across the enterprise. By providing modern solutions based on standards such as HTTP (client and server), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Extensible Markup Language (XML), LEIF reduces development costs and offers the agility to rapidly evolve applications as business needs change. With Rogue Wave's LEIF, it is possible	Rogue Wave Software	1. Rogue Wave's LEIF includes three tiers of functionality that combine to create a comprehensive, platform-independent framework for working with Web services and XML in C++: a. the Data Tier—a high-performance C++ development platform for XML b. the Network Tier—a complete networking and Web services client framework for C++ c. the Service Tier—a robust C/C++ Web services hosting platform 2. The LEIF framework consists of a broad range of C++ components coupled with code generators for extremely fast and easy-to-maintain application development. The LEIF code generators work together to create a high-level API in C++ that handles all the hard work of communication under the covers. 3. LEIF also includes the LEIF Project Wizard, an intuitive graphical user interface (GUI) that creates project files on all supported platforms	www.roguewave.com/products/leif

		to quickly transform a monolithic application into one that can natively interact with all other parts of the technology infrastructure.		for most LEIF tasks and makes it easy to integrate LEIF code generation into an automated build process.	
4.	PiiE Platform	PiiE is a composite application development and runtime environment that delivers integration to the desktop by making it easy to build applications that are composed from many back-end systems, and exposing them to end-users in an actionable interface. The PiiE Platform consists of primary components: • PiiE Smart Client • PiiE Fusion Server	Digital Harbor	**PiiE Smart Client** 1. Drag and drop application design: The PiiE Console Builder makes it easy for developers to build new applications by using extensive drag-and-drop capabilities to create correlated behaviors. Such drag-and-drop correlation is unprecedented in any client/server or Web development environment today. 2. Smart application delivery and management: Digital Harbor PiiE provides over 20 Web services to manage just-in-time, just-enough streaming of applications, as well as security, caching, versioning, and correlation of behavior. These services comprise JavaBeans and XML applied at the interface level. 3. Standards-based application development environment: PiiE Smart Client is designed for integration with your existing infrastructure, including data sources and application servers. By providing a standards-based platform for managing the application consumption layer, PiiE dramatically reduces the technical complexity of developing applications.	http://www.dharbor.com

				4. Security: PiiE Smart Client extends real-time integration with external security systems crucial to extranet operations with support for Lightweight Directory Access Protocol (LDAP). PiiE also provides automatic support for end-to-end encryption. 5. Sample applications: The PiiE Smart Client product suite comes equipped with out-of-the-box sample applications that provide application developers and business analysts with existing solutions that can be leveraged for custom built applications. 6. Collaboration via e-mail: Users can leverage their existing email applications to send and receive smart applications. 7. Extensible IDE: PiiE provides application developers with an extensible integrated development environment that lets developers plug in re-usable custom-built components. **PiiE Fusion Server** 1. One-click discovery (Ask): Digital Harbor's PiiE allows users to ask questions about their information in real time and in context. Questions can pertain to information about relationships, roles, rules, processes, people, and activities. 2. Contextual collaboration with application linking and embedding (ALE™): Digital Harbor's	

				PiiE lets users drag and drop one piece of information onto another to provide context for collaboration. Enterprises get a “common operating picture.” 3. Dynamic workflow: Most workflow tools are stand-alone packages that focus on rigid, predefined processes. A new class of BPM (business process management) technologies is emerging to integrate workflows across back-end systems. What is still missing, however, is the ability to give end-users control over how flexibly they work together—and to do it in the context of the applications they use! 4. Rather than design-time workflow, PiiE Fusion Server provides run-time flexibility to end users: Like a project planning tool that is always live, rather than posted on the wall in the lunchroom, PiiE's Dynamic Workflow features allow end users to annotate, manipulate, and update their processes. 5. Drag-and-Drop Integration / Graphical Modeling: Digital Harbor’s PiiE makes integration easier by defining a relationship model of people, places, and things that can be used by business analysts to build applications. 6. Create rich entity relationships across systems by defining the meaning of the relationship (semantics), not just the syntax. a. Map activities and properties. b. Map events that result from cross-system	

				functions. c. Create or import rules that constrain the effects of events and activities on data and people. d. Map to external applications and data sources. 7. Optimal architecture: Standards-Based Digital Harbor's PiiE is designed to fit your existing infrastructure, including data sources and application servers. PiiE provides XML-based and JDBC compliant data transfer and tools for application layout and development. 8. Rich use of components and semantics: Each object is “aware” of its relationship to other objects. Each concept in a business is described as a person, place, or thing which has meaning to other information, other processes, and other applications. 9. Smart application delivery and management: Digital Harbor PiiE provides over 20 application services to manage just-in-time, just-enough streaming of applications and content, security, caching, versioning, and correlation of behavior.	

5.	SQL Server	SQL Server 2000 exceeds dependability requirements and provides innovative capabilities that increase employee effectiveness, integrate heterogeneous IT ecosystems, and maximize capital and operating budgets. SQL Server 2000 provides the enterprise data management platform your organization needs to adapt quickly in a fast-changing environment. With the lowest implementation and maintenance costs in the industry, SQL Server 2000 delivers rapid return on your data management investment. SQL Server 2000 supports the rapid development enterprise-class business applications that can give your company a critical competitive advantage. Benchmarked for scalability, speed, and performance,	Microsoft	1. easy-to-use business intelligence tools 2. self-tuning and management capabilities 3. data management applications and services 4. native XML Support 5. upcoming version of SQL Server code-named “Yukon” will allow integration with Microsoft CLR 6. advancements such as XQuery and a native XML data type will help enable organizations to seamlessly connect internal and external systems	http://www.microsoft.com/sql

		SQL Server 2000 is a fully enterprise-class database product, providing core support for XML and Internet queries.			
6.	Host Integration Server 2000	Host Integration Server 2000 extends Microsoft Windows to other systems by providing application, data, and network integration. Host Integration Server lets you quickly adapt to new business opportunities while preserving existing infrastructure investments. With its enterprise-class scalability, performance, and reliability, Host Integration Server can support the most demanding business needs.	Microsoft	1. Extending concept of a gateway: Get comprehensive Windows application integration. Host Integration Server 2000 provides host-to-Internet or host-to-intranet application development capabilities. 2. Rapid development: Help developers quickly build distributed client/server solutions that directly access mainframe or AS/400 resources without requiring detailed knowledge of or training on mainframe or AS/400 programming. 3. Complete, secure access to data: Provide object-oriented and programmatic access to relational DB2 data and flat file data on mainframes, AS/400, UNIX, Windows 2000, and Windows NT Server systems. Enable client/server-based applications to access this data transparently as if data were local to the server. 4. Transaction and e-commerce support: Integrate Microsoft Transaction Server and COM+ with IBM's CICS or IMS transaction environments for drag-and-drop simplicity to distributed,	http://www.microsoft.com/hiserver/default.asp

				transaction application development, including support for two-phased commit between platforms. 5. Microsoft Commerce Server and BizTalk Server support: Integrate with the fastest-growing server-based platforms for business-to-consumer and business-to-business services through the Web. 6. Object-oriented programming model: Reap the benefits of object-oriented, distributed application with rapid development, code reuse, simpler maintenance—preserving investment in existing applications and data. 7. Platform for complementary products: Work with a software developer kit (SDK) to provide customized solutions with third-party providers.	
7.	Pervasive Data Junction	Pervasive Data Junction addresses two closely related and converging problem domains in two important ways. First, the technology enables the rapid and reliable construction of interfaces between trading partners; and, secondly, the technology uses the same integration platform to connect all of the major applica-	Pervasive Solutions	Pervasive believes that as the convergence between B2Bi and EAI develops, it also seems that both integration challenges can be served by a common toolset. Developing, implementing, and maintaining two different toolsets simply costs too much and involves redundant solution architecture. This insight includes three basic precepts: 1. Rapid and large-scale integration can occur only via the loosely coupled exchange of business documents. 2. Distributed architectures with integration en-	http://pervasive. com/eai/

		tions and data systems within your enterprise. This two-in-one approach to inter- and intra-enterprise integration helps keep costs down and greatly simplifies the complexities of both B2Bi and EAI.		gines are required, which means that integration only really occurs when all the “edges” of your systems/applications are finally connected. 3. Complex business process automation and integration rely on hundreds and thousands of stateless, fully automated integration micro flows. Pervasive data junction addresses the above issues.	
8.	Microsoft BizTalk Server	BizTalk Server 2002 provides organizations with the server, tools, adapters, and vertical accelerators needed to integrate and automate their businesses.	Microsoft	1. support for SOAP Version 1.1 2. support for XML, XSLT, SMTP, HTTP, PKI Version	http://www.microsoft.com/biztalk/
9.	IBM WebSphere Business Integration	IBM WebSphere® is the leading software platform for e-business on demand, delivering a proven, secure, and reliable software portfolio.	IBM	1. IBM WebSphere Business Integration Connect V4.2 allows you to connect to and integrate with communities of trading partners. 2. support for XML, multiple security standards 3. IBM WebSphere Business Integration Server allows you to integrate business processes.	http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=highlights http://www-106.ibm.com/developerworks/websphere/zones/businessintegration

10.	Ensemble	Ensemble is a comprehensive application integration platform that enables exceptionally fast integration and extremely rapid development of composite applications. Ensemble excels at building new strategic business solutions that leverage the functionality of existing applications, orchestrate new business processes, and integrate data across the enterprise.	InterSystem	1. Full spectrum integration and development: Ensemble provides a unified graphical-, XML-, and code-based development environment for building custom adapters, orchestrating business processes, and creating composite applications. 2. Universal service architecture: Ensemble provides a consistent and efficient object representation of different programming models and data formats. 3. Persistent object engine: At its core, Ensemble uses a high-performance distributed object database for managing and storing all metadata, messages, and process state information. 4. Customizable end-to-end management and monitoring: Ensemble provides highly-customizable and extensible monitoring and management facilities that are tightly integrated with the modeling and development tools.	http://www.intersystems.com/ensemble/technology/index.html
11.	JBoss Application Server	JBoss is a Java-based open source application server with support for J2EE. This enables rapid porting of J2EE-based applications to the free JBoss server. JBoss supports both the construction of Java-based Web Services as well as the integration of possibly non-Java	JBoss Group	1. full microkernel approach based on Java Management eXtensions (JMX) 2. fully hot-deployable and cycleable service layer with Service Archive (SAR) format 3. fully automated and net-based installation with hot-deploy of applications 4. full J2EE 1.3 support (EJB, JCA, JSP, JMX, HTTP, etc.) 5. full security implementation and JAAS integration	www.jboss.org

		based external Web Services through the JBoss.Net extension. A special focus is placed on patterns and tools to expose J2EE™-based logic, such as session beans and entity beans.		6. full clustering of any Java objects (EJB, HTTP) 7. ground-breaking Aspect-Oriented Programming (AOP).	

Appendix B Tool Evaluation Reports

IBM WebSphere

Category	Non-Technical Parameters	Value (In Scale of 1 to 10)	Rationale
Business	Costs Benefits Risk Analysis	7	**Cost** Market Price: $1,000 and $3,500 **Benefits** ROI Good ROI with the tool which is clear from the success stories posted on the Web site. Lifespan of the tool 8. Create, build, test, deploy and publish Web services with support for Universal Description, Discovery and Integration (UDDI), Version 2; Simple Object Access Protocol (SOAP); Web Services Description Language (WSDL); and Web Services Inspection Language (WSIL). Create, validate and detect WS-I Basic Profile, Version 1.0- compliant Web services for greater interoperability between services. 9. Create, test, and deploy J2EE applications to BEA WebLogic Server V6.1 and V7.0 using the Deployment Toolkit for WebSphere Studio, WebLogic Edition. 10. Quickly build rich, interactive user interfaces for Web applications using reusable JavaServer Faces (JSF) **Risk Analysis** 1. Change Frequency Less with two releases in a year. 2. Up-gradation

			a. Easy up-gradation to new releases. (e.g., can easily upgrade the WebSphere Studio from 5.1 to 5.1.1)
Evaluation Specific	Project Specific and Independent Factors Comparison with Other Tools	7	**Project Specific and Independent Factors** 1. provides complete development environment for J2EE, Java, Web Services, XML and Web 2. provides Team Development environment 3. provides Server tools for testing 4. supports almost 10 languages **Comparison with Other Tools** WebSphere 4.0 Advanced Edition costs over 10 times more than Microsoft .NET. **For Web Service Development:** IBM WebSphere 4.0 requires 145% more manual coding than Visual Studio .NET. The following link gives more information on comparative analysis of IBM WebSphere with other tools: http://www.gotdotnet.com/team/compare/webservicecompare.aspx
External Reference	Market Awareness	8	**Market Awareness** Tool is popular among small- and medium-sized organizations and there are almost 20 success stories for this tool listed on the IBM Web site. http://www-306.ibm.com/software/success/cssdb.nsf/lateststoriesVW?OpenView&Count=10&RestrictToCategory=studioappdev

Vendor Support	Quality and Cost of Vendor Support Access to Internal Tool Information Version Choice	6	**Quality and Cost of Vendor Support** Technical support from the vendor is very good. (One user was not able to configure the J2EE Unit test server. I posted this problem to the technical support and got reply the next day.) **Access to Internal Tool Information** No access to architecture and design aspects of the tool **Version Choice** WebSphere Application Development Studio 5.1 best suit the requirements.
Tool-Specific	Integrability and Compatibility Tailorability Domain Tool Integrity and Standards Skills base and User Capability Data Configura-tion Control Tool Modularity Response Time User Interface	8	**Integrability and Compatibility** 1. built on Eclipse and can be customized and extended with a wide range of IBM Products and partner and Eclipse based plug-ins. 2. integration with software configuration management tools 3. supports Windows and Linux platform **Tailorability** Tool can be tailored to the user requirements using built-in wizards and plug-ins. **Domain** Provides J2EE, Java, Web Services, XML, and Web Development Environment and also provides relational database tools. **Tool Integrity and Standards** 1. Support for Universal Description, Discovery and Integration (UDDI), Version 2; SOAP; WSDL; and Web Services Inspection Language (WSIL), J2EE 2.0 and 3.0, UML, SQLJ, etc. 2. Tool is very reliable; it prompts the error messages as when user inputs any value. 3. Vendor Maturity is quite high.

	Documentation Data Portability		**Skills Base and User Capability** Normal user with knowledge of J2EE, Web Services and Web Development can operate this tool. The tool provides different development views (UI) known as per perspective and per terminology for each development environment. For Example in J2EE perspective the tool displays the J2EE Architecture Components on the left navigational bar. (In one case it took a week to get acquainted with the basic features provided by the tool and to develop a sample J2EE application which adds MP3 song information to the Cloudscape database using the Web interface in three hours.) **Configuration Control** Application Developer provides monitoring and profiling tools that features customizable views and logs enabling you to recognize, isolate, and fix performance problems early in the development cycle, support for object level profiling, analysis of WebSphere Application Server activity logs, and interaction with the symptom database and Log and Trace Analyzer. **Tool Modularity** 1. built on Eclipse 2. provides comprehensive development environment through visual tools, templates, code generation utilities **User Interface** GUI based development makes the tool very user friendly. Helpful error messages help in locating the cause of error. **Response Time** Performance of the tool is good. **User Interface Documentation** Tool is bundled with sufficient documentation. The official Web site provides technical information and has lots of discussion forums to elicit any kind of information.

			Data Portability Data generated by the tool is mostly in text format which can be viewed by any text editor.
Emplacement	Integration with Other Tools	7	**Integration with Other Tools** 1. Integration with several Software Configuration Management solutions; provides flexibility in asset management and team development 2. Tool uses Version Control Management interface provided by Eclipse V2.1 and an adaptor for IBM ClearCase LT and for CVS.

IBM Rational Rapid Developer

Category	Non-Technical Parameters	Value (In Scale of 1 to 10)	Rationale
Business	Costs Benefits Risk Analysis	5	**Cost** Market Price: $ 5995 per version **Benefits** 1. The tool has easy-to-use modeling capabilities. 2. Lifespan of the tool a. The tool supports the design, integration, deployment, construction of business applications. It automates many tasks so that developers focus on highest value areas. b. The Rapid Developer Runtime API allows application developers to create this business logic using a standardized interface that works under all supported technologies. c. Rapid Developer supports the design and construction of pages, messages, components, and Web Services using either J2EE or MSDNA Technologies.

			Risk Analysis 1. The tool is highly dependent on Rational Tools, so the application must be modeled by "Rational" methodology.
Evaluation Specific	Project Specific and Independent Factors Comparison with Other Tools	5	**Project Specific and Independent Factors** The tool provides "architecture-centric" development environment which only depicts a "tier-based" view of the entire application thus restricting the user to develop tier-based applications. **Comparison with Other Tools** Most of the modeling features are same as those in Rational Suite of Tools.
External Reference	Market Awareness	6	**Market Awareness** The tool is popular among different sized enterprises and there are success stories which are listed on the IBM Web site.
Vendor Support	Quality and Cost of Vendor Support Access to Internal Tool Information Version Choice	6	**Quality and Cost of Vendor Support** Vendor Support and Supporting Documentation are reasonable but not technically sound. **Access to Internal Tool Information** No access to Architecture and Design aspects of the tool is available. **Version Choice** Release 2003.06.00.105. is suitable.
Tool Specific	Integrability and Compatibility	6	**Integrability and Compatibility** Can be integrated with Rational Clear Case, Rose Diagrams, it is also compatible with IBM Web-

	Tailor-ability Domain Tool Integrity and Standards Skills Base and User Capability Data Configuration Control Tool Modularity Response Time User Interface Documentation Data Portability		Sphere Application Server, Oracle 9.1 Application Server. However, there is an over-dependency on IBM Tools. **Tailor-ability** Tool can be tailored to the user requirements using templates and patterns. **Domain** Provides J2EE, Java, Web Services, XML, Legacy Services, and Web Development Environment; also provides relational database tools. **Tool Integrity and Standards** 1. The tool supports Universal Description, Discovery and Integration (UDDI), Version 2; Simple Object Access Protocol (SOAP); Web Services Description Language (WSDL); and Web Services Inspection Language (WSIL), J2EE 2.0 and 3.0, UML, SQLJ, etc. 2. Tool is reliable and provides good exception-handling facilities. **Skills Base and User Capability** Any user with knowledge of J2EE, Web Services, or Web Development can operate this tool. The tool provides different “architects” as one goes about designing the application. However, the knowledge of legacy databases is a good add-on in case the tool is to be used for legacy integration purposes. **Configuration Control** The tool provides functionality for interfacing to a version-control system but does not have one built in. **Tool Modularity** Provides comprehensive development environment through templates, code generation utilities, patterns, and views **User Interface**

			The GUI-based functionality is easy to use—specifically the “tabs” to design the application. **Response Time** Performance of the tool is reasonably ok. **User Interface Documentation** The tool is bundled with sufficient documentation. The IBM Web site provides a good resource for white papers, demos, evaluation guides.
Emplacement	Integration with Other Tools	7	**Integration with Other Tools** Integrates with other non-Rational-based Source Control Systems, but does not integrate with a wide variety of applications

LEIF

Category	**Non-Technical Parameters**	**Value (In Scale of 1 to 10)**	**Rationale**
Business	Costs Benefits Risk Analysis	5	**Cost** LEIF Developer Network: License + Basic Support **$1,495.00** LEIF Developer Network: License + Premium **Support $1,790.00** **Benefits** **Business**

			Ease of use and comparatively shorter learning curve for a person with domain knowledge ROI is much higher in cases of applications developed using C++. **Product** 1. a rapid development tool for creating Service oriented applications involving C++ 2. creates Web services using the existing legacy applications without need for modification to the existing business logic, thereby reducing time spent in rewriting the code for the same 3. addresses scalability and high-performance attributes of C++ applications. **Risk Analysis** **Application** Applying service-oriented development and Web services only for C++ Applications **Change Frequency** The change frequency is high since it is under the debugging and fixing stage of the product. Although the LEIF Version 2.0 is a stable one, based on which up-gradation can be done, the change frequency may be high.
Evaluation Specific	Project Specific and Independent Factors Comparison with Other Tools	6	**Project Specific and Independent Factors** For the project that we have selected, the “Trading Bond System,” LEIF is used for converting the stand-alone C++ application into a client server application. The core C++ functionality is exposed through a Web service. It was specifically designed to bridge the C++ and Java gap and represented the least amount of work. **Comparison with Other Tools** There is not much competition or comparison to be made. since LEIF is one among the very few service-oriented rapid development tools for C++ applications. The other tool that supports the same is IONA Artix; but the complexity and the learning curve involved in IONA Artix are

			greater than that required for LEIF.
External Reference	Market Awareness	6	**Market Awareness** Rogue Wave Software Inc. is a leading provider of reusable software components and services for application development. Since December 19, 2003, it has been a subsidiary of Quovadx Inc. The following hyperlinks access the various press releases, articles that highlighted the product services of Rogue Wave, and the awards granted. http://www.roguewave.com/corp/press/pressrel/ http://www.roguewave.com/corp/press/articles/ http://www.roguewave.com/corp/press/awards/ Though the company enjoys significant market awareness, the product is still under experiment. It does provide good case studies that are proven success stories for LEIF. http://www.roguewave.com/products/leif/Putnamcs.pdf
Vendor Support	Quality and Cost of Vendor Support Access to Internal	8	**Quality and Cost of Vendor Support** Users can obtain technical support by 1. contacting technical support 2. using the knowledge base

	Tool Information Version Choice		3. accessing various support programs 4. reading product documentation “The response time or the turnaround time of the LEIF tool vendor support was one day when I faced the problem with the generation of a project.” http://www.roguewave.com/support/ **Access to Internal Tool Information** The user guide of the LEIF Framework gives detailed information about LEIF architecture with its three tiers and their benefits. It also talks about how the tool is actually processing the service-oriented applications **Version Choice** Rogue Wave issues three types of LEIF product releases: 1. **Maintenance (x.y.z)** releases provide bug fixes and/or platform updates. 2. **Minor (x.y.0)** releases provide bug fixes, platform updates, and minor product enhancements. 3. **Major (x.0.0)** releases provide bugs fixes, platform updates, and major product enhancements or new features. The current major version or release of LEIF is 2.0, which provides high benefits in terms of performance, integration and extensibility. http://www.roguewave.com/products/leif/whatsnew.cfm
Tool Specific	Integrability and Compatibility	8	**Integrability and Compatibility** • Broad messaging pattern support: Choose the appropriate messaging pattern from request/response, asynchronous (IOU), one-way, server-side notification, and server-side solicit

	Tailor-ability Domain Tool Integrity and Standards Skills base and User Capability Data Configuration Control Tool Modularity Response Time User Interface Documentation Data Portability		response. Clients can listen for server-initiated events. • IBM WebSphere MQ™ transport: Get superior reliability and integration by connecting LEIF services directly to the message queue—no need for complicated (and slow) bridges. • Exhaustive interoperability testing: Participation in groups like SOAP Builders ensures that LEIF services can be coupled with services exposed using other technologies. **Tailorability** Below are some of the characteristics of LEIF that makes the product tailorable. • Easily write custom transports and extend LEIF to work with virtually any middleware. • Add emerging Web services standards without affecting the business logic of the service. • Change deployment behavior, insert transports, or modify message handlers without changing the service code. **Domain** LEIF is based on industry-standard networking, XML, and Web services technologies. In particular, LEIF provides strong support for XML as a medium of data exchange, enabling interprocess communication that spans platforms and languages. LEIF fits well into the services-oriented architecture (SOA) approach to application development. **Tool Integrity and Standards (as per the User Guide Information)** LEIF products conform to the following standards: • The LEIF Web service code generator uses WSDL 1.1 . • LEIF provides documents to create components. The components produce SOAP that complies with the SOAP 1.1 specification. • The LEIF XML data binding supports the most common and useful features of the May 2001

XML Schema recommendation.

- The LEIF Web services container closely follows the Java Servlet specification. However, the specification requires that an implementation be written in the Java language. Therefore, a C++ implementation cannot strictly conform to all features of the specification. The Bobcat Reference Guide contains information on differences in individual classes.
- LEIF participates in the SOAP Builders initiative aimed at guaranteeing interoperability among SOAP implementations. To view the Rogue Wave test WSDLs, visit www.whitemesa.com/interop.htm. In addition, LEIF is extensively tested against the most frequently used Java and Windows-based (.NET) implementations of Web services.

Skills Base and User Capability

The user capability and skills base requires the user to be aware of the Web service applications and service-oriented architecture and also be capable of creating the WSDL for the application whose service has to be created.

Configuration Control

The configuration and the development environment set up are well explained in the user guide of the LEIF. It spans across three steps:

- Set RW_HOME (UNIX/Linux only).
- Run leifvars.
- Ensure the presence of a JRE.

Tool Modularity

The tool is not highly modularized; it depicts a three-tier interface for the SOA to be developed.

User Interface

A simple and chaste user interface of LEIF makes the work of the tool easily understandable and also reduces the complexity involved in the user interaction.

			Response Time Fastest available processing of Web service messages - up to 300% net performance gain over the previous release LEIF 1.2! **User Interface Documentation** LEIF offers a standard user interface documentation. http://www.roguewave.com/products/leif/gui.cfm
Emplacement	Integration with Other Tools	7	**Integration with Other Tools** LEIF includes a version of the Apache HTTP Server or Microsoft IIS preconfigured for use with LEIF in a development environment. LEIF is certified to use Xalan Java 2.4.1 and Xerces-J 2.3.0 for all supported versions of the JDK. Installing LEIF requires support of the following software, based on the operating system: • Visual Studio .NET 2003 • Sun ONE Studio 8 • gcc 3.2.3

Appendix C Model Problem and Analysis

The model problem chosen for this project was described as follows:

> A major Wall Street investment bank sets out to build a bond pricing system in an effort to streamline the workflow of its bond trading desk. Currently, bond traders have to send prices for a large number of bonds to several different trading venues, each with its own user interface.

The system that solves the above problem must also minimize the minutiae of pricing all the bonds and provide advanced analytic functionality specific to the bond markets. These capabilities must be provided through a single encapsulated interface.

Classification Scheme Approach

Step 1: Read the problem statement and identify functional and non-functional requirements.

The following requirements can be inferred from the above problem statement:

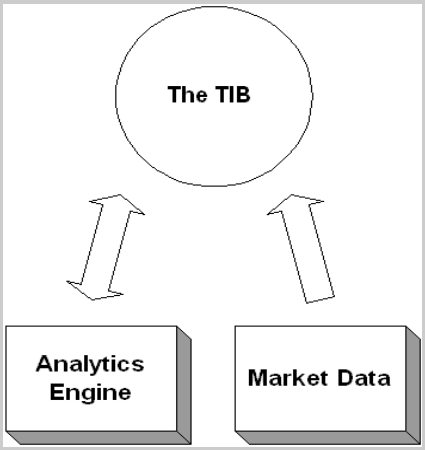
1. high user interaction
2. integration with the legacy system
3. communication and data exchange mechanism for component[2] interaction
4. communication between the C++ and the JAVA applications

Step 2: Map the requirements identified to the integration mechanism, which forms the classification parameters to be identified in the rapid integration tools.

Analysis of the Functional Requirements

For each of the requirements a specific integration mechanism is suggested as a solution. The mechanism will be specific to the particular application. Therefore, the integration mechanisms specified below cannot be generalized for all applications.

[2] Here we mean the three components specified by the application: Market Data, Analytics Configuration and Contribution Server [legacy servers].

Requirements	Solutions	Integration Mechanism Required
High User Interaction Traders need a very responsive application.	Client application as a Java thick client because of Java's platform independence and its ability to quickly respond to user input and market data	None
Integration with the legacy system On the server side, it will inherit legacy C++ components that the system will utilize. Also, the market data components communicate with the TIBCO[3] Information Bus (TIB) messaging infrastructure	The following components are to be integrated: • **Market Data Price Feed Server**: publishes incoming market data to the TIB • **Analytics Engine**: performs analytics on incoming market data and broadcasts the modified market data to the TIB • **Contribution Server**: performs all communication with trading venues. The trading venues are third-party components not controlled by the bank.  *Figure 8: Legacy Market Data Subsystem*	JAVA to C++ Translator (Java thick Client talking with C++ Legacy Servers)

[3] TIBCO means standard industry-specific messaging infrastructure component.

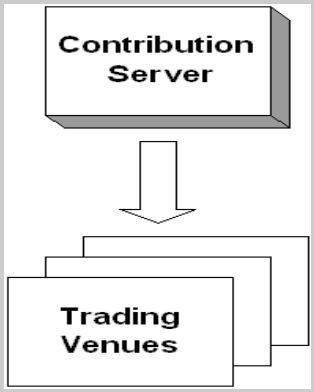
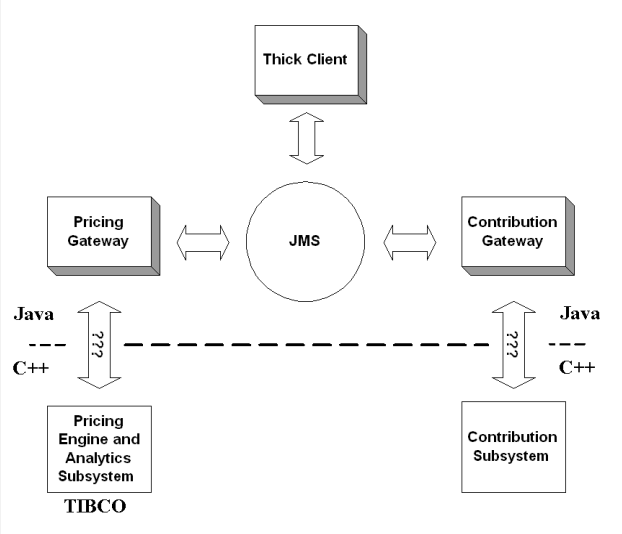
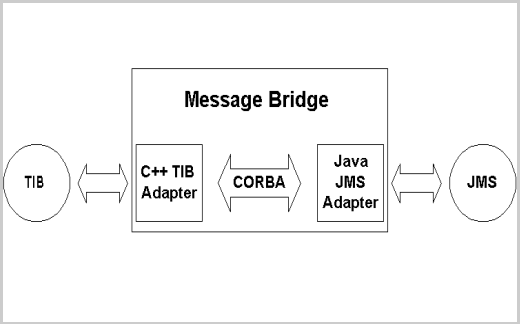


Figure 9: Legacy Contribution Subsystem

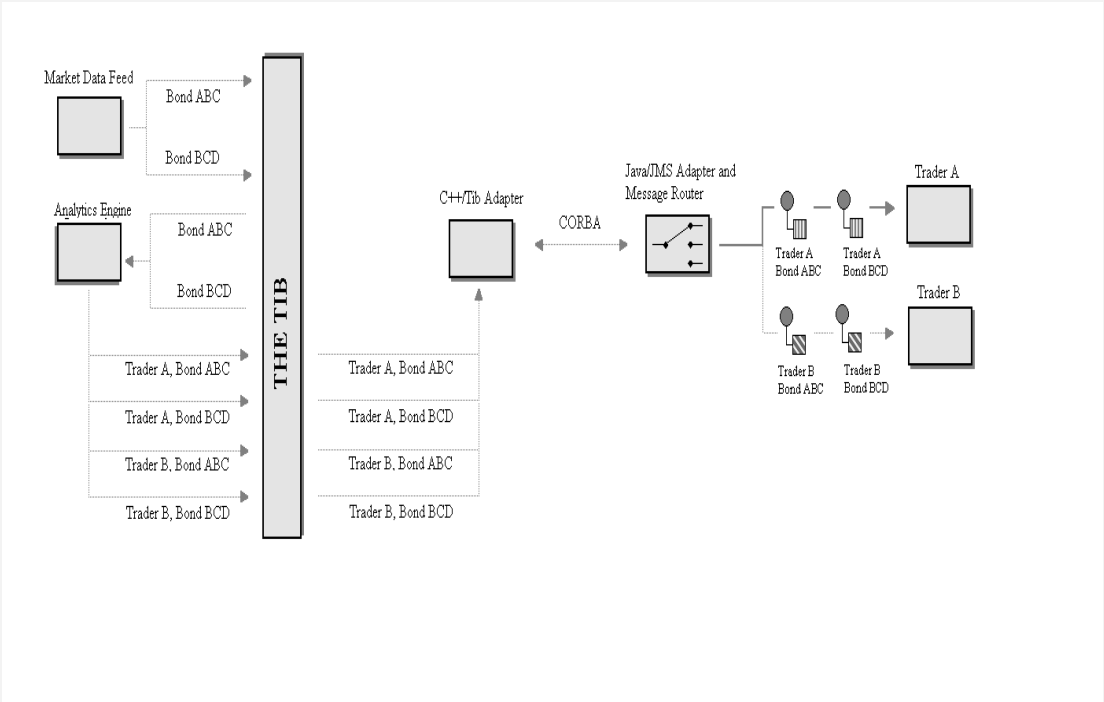
Communication and data exchange mechanism for component[4] interaction Communication and Data exchange mechanism between sub-components (Thick Client, Market Data and Contribution)	Two gateways to communicate with the legacy servers: • Pricing Gateway for market data • Contribution Gateway for sending prices to trading vendors For instance: With Messaging, we can define separate channels for the different types of pricing data. Then, when a Gateway gets a new piece of data, it will add a message containing that data to the Publish-Subscribe Channel for that data type. Meanwhile, all clients interested in a certain type of data will listen on the channel for that type. In this way, the Gateways can easily send out new data to whoever is interested, without needing to know how many listener applications there are or what they are.  	Single point of access through Gateways Messaging, Publish and Subscribe Channel, JMS (as components are written in JAVA)
Communication between the C++ and the JAVA application How to connect the JMS with the standalone C++	Cross language (C++ and JAVA) Messaging Bridge using a combination of Channel Adapters and CORBA.	Messaging Bridge, Channel Adapters and Communication Vehicle between Adapters

[4] Here we mean the three components specified by the application, that is, Market Data, Analytics Configuration and Contribution server [Legacy Servers].

Contribution server and the TIBCO based Market Data and Analytics Engine servers?		

Analysis of the Non-Functional Requirements

Non-functional Requirement	Description
Performance Here it refers to the scalability, which can be measured as the number of users it can scale to without noticeable decrease in response time.	One Channel per trader per Bond: Create one Message Channel per-trader per-bond solely for the modified market data of that bond. For example, the market data for bond ABC would be published on channel “Bond ABC” while the modified market data for trader A would be published on Message Channel “Trader A, Bond ABC,” modified market data for trader B on “Trader B, Bond ABC,” and so on.



Cost Custom development effort for integration	Effort: 47 person-months for developing the integration components (Refer to Appendix C: Trading Bond COCOTS Estimation Details.)
Hardware/Software Requirements	Below are the hardware and software requirements regarding components. 1. Analytic Engine and Contribution Server a. a high-end server class machine with minimum of 512 Mb of RAM b. Windows 2000 server 2. Traders Desktop Machine (Client): a. Windows NT, Solaris b. 128 MB of RAM c. Java Virtual Machine 3. TIBCO Information BUS Messaging infrastructure 4. Market Data Price Feed Server

Impacts/Change Analysis on Architecture

The high-level architecture of the system is represented in Figure 10.

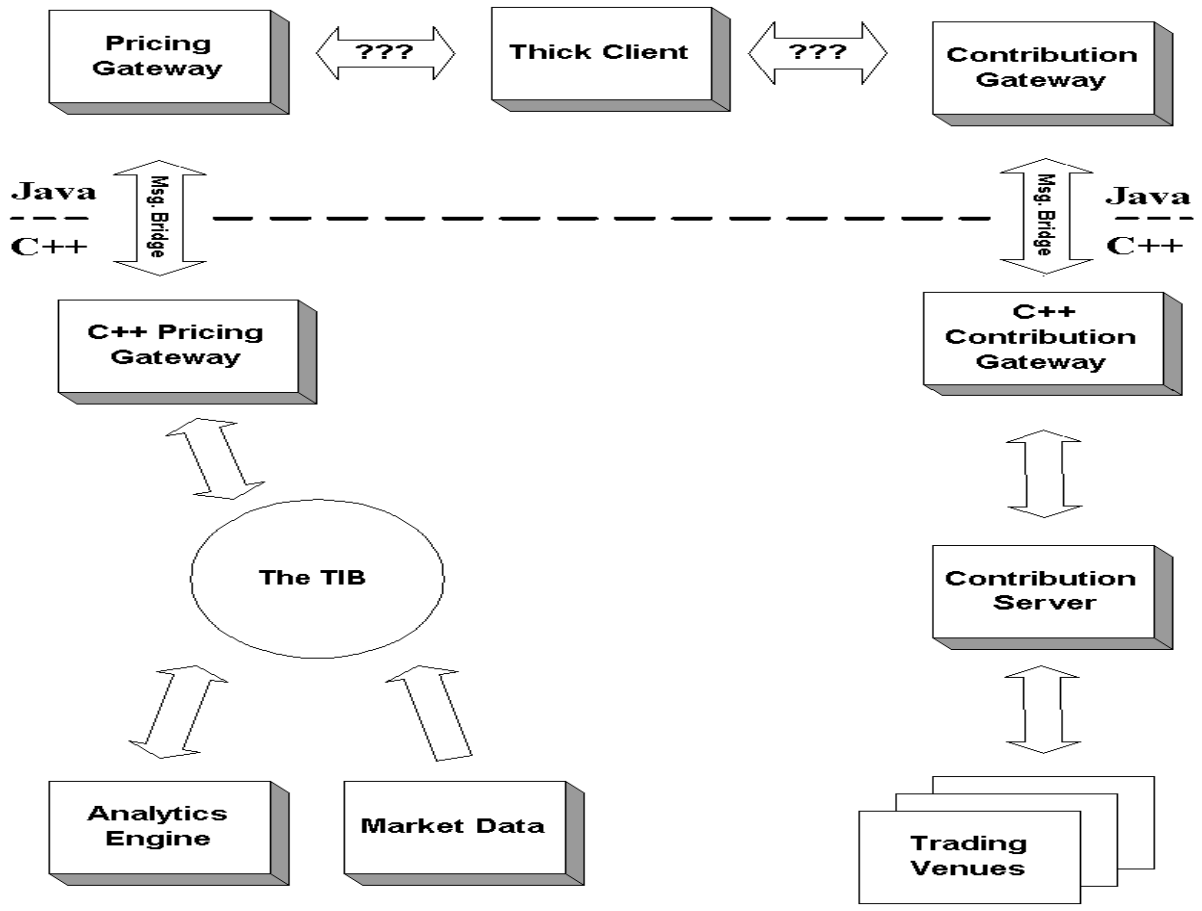


Figure 10: Logical View of the System

1. TIBCO Information Bus Messaging infrastructure has been selected to achieve three-way communications between Market Data Feed Server, Analytics Engine and Pricing Gateway as shown in Figure 10.
2. Two Java Gateways are used to provide communication between the Market Data Feed Server and the Trading Venues:
   a. Pricing Gateway for Market Data Feed Server
   b. Contribution Server for sending prices to Trader Venues
3. Message Bridge is used to provide communication between JMS used to provide communication between Pricing and Contribution Gateways and TIB (TIBCO Information Bus). This message bridge has C++ and Java Adapters and these adapters communicate with each other through CORBA.

Constraints and Assumptions Made about the Components

1. The system inherits C++ legacy components namely: Market Data Feed Server and Contribution Server.
2. The system also uses TIBCO Information Bus Messaging Infrastructure as a third-party component.
3. The traders' venue desktop can run on Windows NT or Solaris Operating System.

Step 3: List the integration mechanisms which are the classification parameters and categorize them into primitive classification types.

Categorization of Classification Parameters

Integration Patterns (Primitive Classification Type Parameters)	Primitive Classification Type
Legacy Translator (Java thick Client talking with C++ Legacy Servers)	Legacy Integration
Gateways	Application Integration
Messaging (JMS)	Middleware Integration
Publish and Subscribe Channel	Middleware Integration
Messaging Bridge	Application Integration
Channel Bridge	Application Integration
Communication Vehicle between Adapters	Application Integration

From this table we can infer that the current scenario is a composite of three primitive classification types, namely

- Legacy Integration
- Application Integration
- Middleware Integration

Step 4: Identify the rapid integration tools needed to quickly solve this problem.

In this step we try to represent the scenario as a set of classification parameters. Here we have the integration mechanisms that serve as the classification parameters.

Mathematically, scenario can be expressed as

Scenario1 = {Legacy Translator, Gateways, Messaging (JMS), Publish & Subscribe Channel, Messaging Bridge, Channel Bridge, Communication Vehicle between Adapters}

The parameters identified using this scenario form the elements of the primitive classification type.

1. Legacy Integration = {Legacy Translator}
2. Application Integration = {Gateways, Messaging Bridge, Channel Bridge, Communication Vehicle between Adapters}
3. Middleware Integration = {Messaging (JMS), Publish and Subscribe Channel}

For the current scenario the parameters assume following values:

1. Legacy Translator=“Java to C++ Translator”
2. Gateways = “Java Gateways”
3. Messaging Bridge=“Bridge C++ Messaging system to JAVA Messaging System”
4. Channel Bridge=“C++ TIB Adapter & JMS Adapter”
5. Communication Vehicle between Adapters =“CORBA”
6. Messaging (JMS) =“IBM MQ Series”
7. Publish and Subscribe Channel=“Channel for different types of pricing data with Gateways as Publisher and Clients as Subscriber”

From the analysis done to classify the rapid integration tool we determine it to be a set of the combinations of the primitive classification types:

RIT for Scenario = {Legacy, Application, Middleware}

Similarly when we generalize it

RIT for $Scenario_n$ = {Primitive Classification Type *}

Step 5: Select tools.

Through use of the Tool Classification Matrix the following tools are identified to support this integration.

1. Microsoft BizTalk Server
2. IBM WebSphere
3. Pervasive Data Junction

Tools Classification Matrix

Name of the Tool	Classification based on Primitive Integration Types				
	Legacy Integration	Middleware Integration	Service-Oriented Integration	Application Integration	Web-Based Integration
Pervasive Data Junction					
RogueWave's LEIF					
IBM Rational Rapid Developer					
Microsoft SQL Server					
Host Integration Server					
Microsoft BizTalk Server					
IBM WebSphere Business Integration					
Artix Relay					
Artix Encompass					
Artix Mainframe					
PiiE Smart Client					
PiiE Fusion Server					
InterSystem Ensemble					
Jboss					

Appendix D Commercial Off-the-Shelf Components

This section describes the Customized Dukascopy Data Feed Components (CDDF).
http://www.dukascopy.com/english/ddf_main/rdata/

Java Feeder Component:
The Feeder components are commercial off-the-shelf (COTS) products provided by Dukascopy. These components connect themselves to Dukascopy Market Machine data source and supply data every 10 seconds to the software application connected to it. Dukascopy Market Machine data source supplies data on liquid trading instruments.

> The data has the following format:
> stockId – integer
> Value – double
> tickVolume – integer (on every instrument)
>
> where
> stockID is the ID of trading instrument set by the user
> Value is an average 10 sec price value.

Besides providing real-time data, this component can also transfer historical data going back three days (nearly 22000 of 10 sec ticks) that can be used to fill in occasional gaps in the database.

Component Specification:

The Feeder component provides interfaces and methods listed below. These can be used by the application program to capture the data collected by this component from Dukascopy Market Machine data source:

1. DataListener Interface

   onNewTick(int id, double value, int volume): This method provides the data that is fetched from the Dukascopy Market Machine data source.
2. TickerListener Interface
   a. onNewTick(int id, double value, int volume): This method provides the data that is fetched from the Dukascopy Market Machine data source.
   b. onNewConnection(Connector conn)
3. addQuote(int id, String code) method in FeederConnector Class: This method allows the application program to add a specific trading instrument for which the data has to be collected.
4. removeQuote(String code) method in FeederConnector Class: This method allows the application program to remove a specific trading instrument for which the data has to be collected.

5. setDataListener(DataListener dl)) method in FeederConnector Class: This method provides the data that is fetched from the Dukascopy Market Machine data source. It eventually uses the on-NewTick(int id, double value, int volume) method to get the data.
6. connect() method in FeederConnector: This method initiates the connection of this component to the Dukascopy Market Machine data source.

Figure 11 below shows the interfaces and methods within those interfaces which are accessible to external programs.

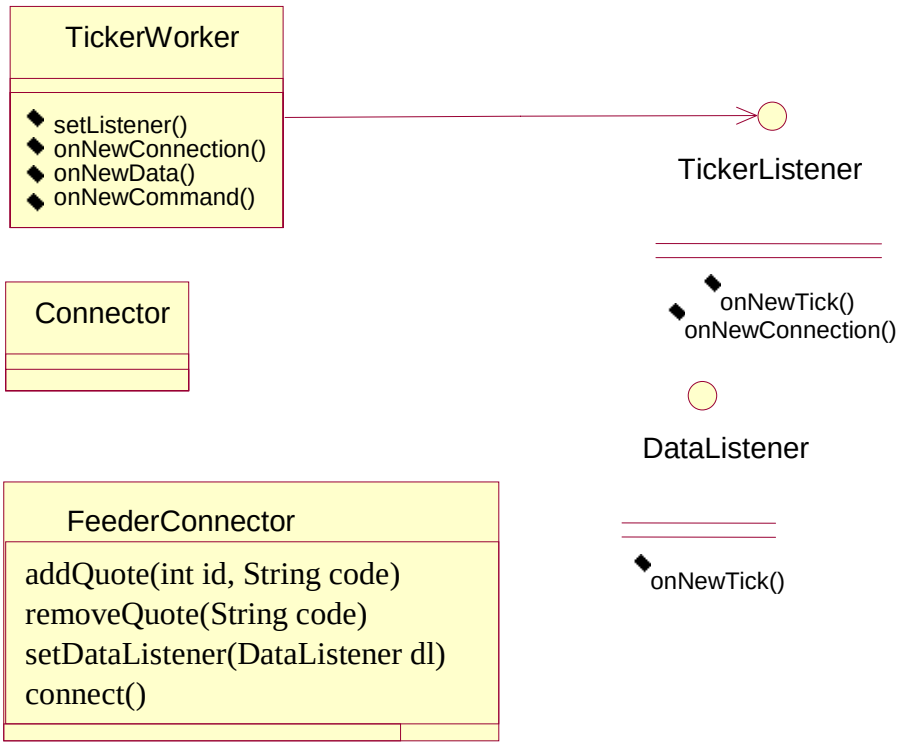
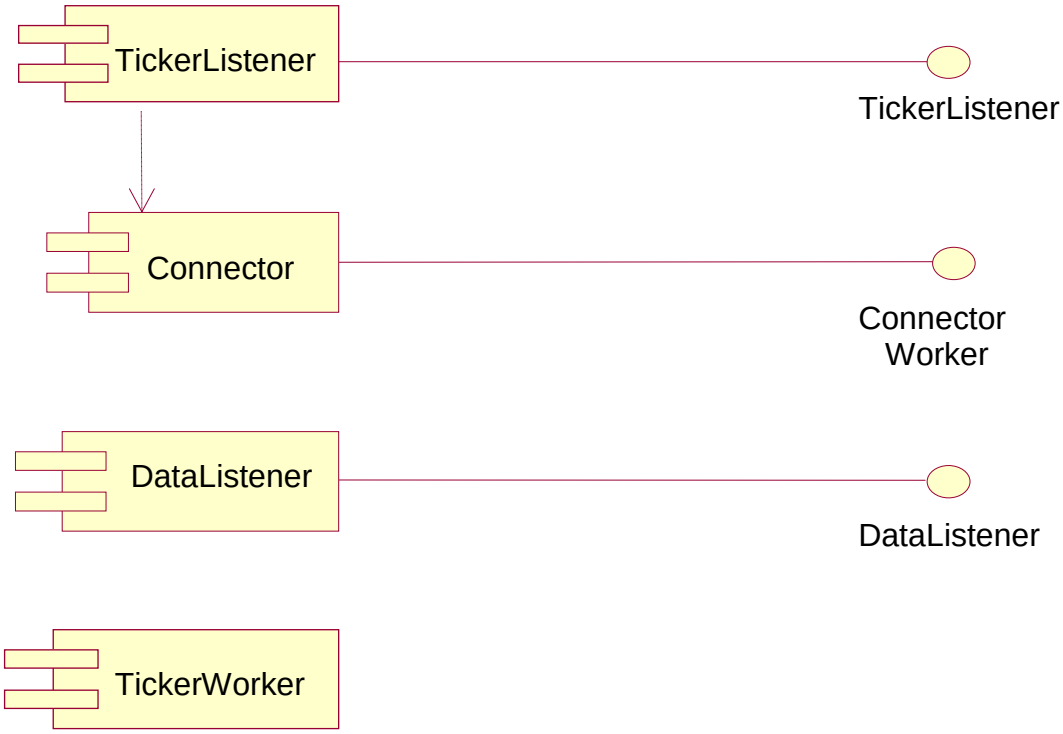


Figure 11: Feeder Component Specification

Component Realization:

The feeder component is implemented using the following Java classes and interfaces.

1. FeederConnector
2. ConnectorWorker
3. Connector: Protocol realization
4. DataListener: Client interface for working with data
5. TickerListener
6. TickerWorker



VC++ MarketDataFeed Component:

The following are the VC++ files that define the responsibility of the VC++ Component:

1. CConn.vcproj

   This is the main project file for VC++ projects generated using an Application Wizard. It contains information about the version of Visual C++ that generated the file and information about the platforms, configurations, and project features selected with the Application Wizard.

2. CConn.idl

   This file contains the IDL definitions of the type library, the interfaces and co-classes defined in the project. This file will be processed by the MIDL compiler to generate

   C++ interface definitions and GUID declarations (CConn.h)

3. CCoCConn.vcproj

   This is the main project file for VC++ projects generated using an Application Wizard. It contains information about the version of Visual C++ that generated the file, and information about the platforms, configurations, and project features selected with the Application Wizard.

4. CConn.idl

   This file contains the IDL definitions of the type library, the interfaces and co-classes defined in the project.

This file will be processed by the MIDL compiler to generate the following:

C++ interface definitions and GUID declarations	(CConn.h)
GUID definitions	(CConn_i.c)
A type library	(CConn.tlb)
Marshaling code	(CConn_p.c and dlldata.c)

5. CConn.h

   This file contains the C++ interface definitions and GUID declarations of the items defined in CConn.idl. It will be regenerated by MIDL during compilation.

6. CConn.cpp

   This file contains the object map and the implementation of your DLL's exports.

7. CConn.rc

   This is a listing of all of the Microsoft Windows resources that the program uses.

8. CConn.def

   This module-definition file provides the linker with information about the exports required by the DLL. It contains exports for

   DllGetClassObject
   DllCanUnloadNow
   GetProxyDllInfo
   DllRegisterServer
   DllUnregisterServer

   Other standard files

9. StdAfx.h, StdAfx.cpp

   These files are used to build a precompiled header (PCH) file named CConn.pch and a precompiled types file named StdAfx.obj.

10. Resource.h

    This is the standard header file that defines resource IDs: Proxy/stub DLL project and module definition file.

11. CConnps.vcproj

    This file is the project file for building a proxy/stub DLL if necessary.

    The IDL file in the main project must contain at least one interface and you must first compile the IDL file before building the proxy/stub DLL. This process generates dlldata.c, CConn_i.c and CConn_p.c , which are required to build the proxy/stub DLL.

12. CConnps.def

    This module definition file provides the linker with information about the exports required by the proxy/stub.

Other notes:

The MFC Support option builds the Microsoft Foundation Class libraries into your skeleton application, making MFC classes, objects and functions available to you.

Issues

1. If the client process is not killed properly, the Java component will still deliver the data to the client application. This state prevents the client application from re-establishing the lost connection to properly terminate the data stream.

Appendix E Trading Bond System COCOTS Estimation Details

The approach followed is strictly based on the COCOTS estimation model proposed by Christopher M. Abts and Barry W. Boehm [Abst 00]. The standard COCOTS calibration tables are used for the calibrated parameter values for each cost driver in the model. The corresponding parameter value for each driver is fed into the spreadsheet tool—COCOTS calculator.

Assumptions

1. The values of very high, low, and so forth, have been determined based on a heuristic approach rather than on previous data collection.
2. The KSLOC is assumed to be based on the programming experience of the team with the prior knowledge of the domain addressed here. The SLOC for developing a glue code for integrating the C++ and Java Components using JNI is found to be approximately 1000 SLOC [1 KSLOC].

   The component that is the glue code for the integrating C++ and Java is assumed to be developed using JNI. We realize that the excerpts taken from the article on Junc++ion (http://www.codemesh.com/en/CodemeshWhitepaper.pdf) demonstrate that JNI requires a huge number of lines of code.

   > "If the programmer were trying to write an application to display a Java Swing dialog box from C++ and store the user's input in C++ using JNI to communicate between C++ and Java, about 200 lines of JNI code would be required."

3. Since there are no real-world customers, there is a very minimal requirement change for this integration scenario and hence the BRAK % is assumed to be 1.
4. The Normal Labor Cost here refers to the Software Engineers in any company that will be involved in the development.

Constraints

Currently, we have one option for C++ and Java components. Also, the Trading Bond System here addressed is restricted to the legacy integration of components

Cost Drivers Selection

The following table presents the values selected and the reasons for their selection for the various cost drivers of the COCOTS estimation model.

Category	Cost Drivers	Value	Why?
Integration Personal Drivers			
1	COTS Integrator Experience with Product	VL	Two months of experience with the products
2	COTS Integrator Personnel Capability	L	Two months of experience with the domain
3	Integrator Experience with COTS Integration Processes	L	Organizational level [Professional Development Center] process for COTS integration is not defined.
4	Integrator Personnel Continuity	N	There will be a rotation of people every year in the Professional Development Center, as it is an educational environment.
COTS Component Drivers			
1	COTS Product Maturity	H	The product has high time on market.
2	COTS Supplier Product Extension Willingness	L	The products we consider here are standard C++ and Java components available on the net; hence the number and nature of changes are very minimal.
3	COTS Product Interface Complexity	N	Since most of the APIs of the components are well defined, consistently applied, and clear, they can easily be used to interface with the glue code.
4	COTS Supplier Product Support	H	The level of available support is high; a detailed explanation of the components to be integrated is available.
5	COTS Supplier Provided Training and Documentation	N	Nominal documentation is provided for the scenario considered here.
6	COTS Product Volatility	L	Only one release is expected.

Application/System Drivers			
1	Constraints on System/Subsystem Reliability	N	This is not a mission-critical system; there are backup servers to recover the lost data.
2	Application Interface Complexity.	L	Use of standard communication mechanisms such as APIs reduces the application interface complexity.
3	Constraints on System/Subsystem Technical Performance	VH	The analytic engine handles the real-time market data and feeds it to trader's desktop.
4	System Portability	VH	The traders' desktops might be running on different operating systems.
Nonlinear Scale Factor			
1	Application Architectural Engineering	L	Simple paper Analysis of the architecture of the system will be done for the currently selected scenario.

COTS Integrator Calculator

	CMU-West														
	COTS Integration Cost Calculator V2.0														
	(COCOTS Glue Code Sub-model)														
		Cali-bration Tables:													
		Driver													
	Rating	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
		ACIEP	**ACIPC**	**AXICP**	**APCON**	**ACPMT**	**ACSEW**	**APCPX**	**ACPPS**	**ACPTD**	**APVOL**	**ACREL**	**AACPX**	**ACPER**	**ASPRT**
	VL	**1.34**	1.60		1.58	1.45				1.20	0.71				
	L	1.16	**1.27**	**1.12**	1.26	1.20	**1.07**	0.82	1.14	1.09	**0.84**	0.88	**0.84**		
	N	1.00	1.00	1.00	**1.00**	1.00	1.00	**1.00**	1.00	**1.00**	1.00	**1.00**	1.00	1.00	1.00
	H	0.86	0.79	0.89	0.80	**0.83**	0.94	1.22	**0.88**	0.91	1.19	1.14	1.19	1.11	1.07
	VH	0.75	0.62	0.79	0.63	0.69	0.88	1.48	0.77	0.84	1.33	1.30	1.42	**1.22**	**1.14**
			Linear Scaling Factor						**Nonlinear Scaling Factor**						
			A						**AAREN**						
			12.0						**VL**	**L**	**N**	**H**	**VH**		

									4.00	**3.00**	2.00	1.00	0.00		
		Inputs:													
		Driver													
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
		ACIEP	**ACIPC**	**AXICP**	**APCON**	**ACPMT**	**ACSEW**	**APCPX**	**ACPPS**	**ACPTD**	**APVOL**	**ACREL**	**AACPX**	**ACPER**	**ASPRT**
	Value	**1.34**	**1.27**	**1.12**	**1.00**	**0.83**	**1.07**	**1.00**	**0.88**	**1.00**	**0.84**	**1.00**	**0.84**	**1.22**	**1.14**
			Scaling Factor					**Sizing**				**Labor Cost**			
			AAREN					**KSLOC**	**BRAK (%)**			**Rate ($/Pr-mt)**			
		Value		**3.00**				1.00	1			7000.00			
Outputs:															
	ESIZE	**B**	**EM_{1-14}**		**Estimated Effort (Prs-mts)**						**Estimated Cost ($)**				
	1.01	1.00	1.46		17.63						$123,407				

Appendix F Project Details

This appendix contains the detailed work breakdown structure (WBS) and project details.

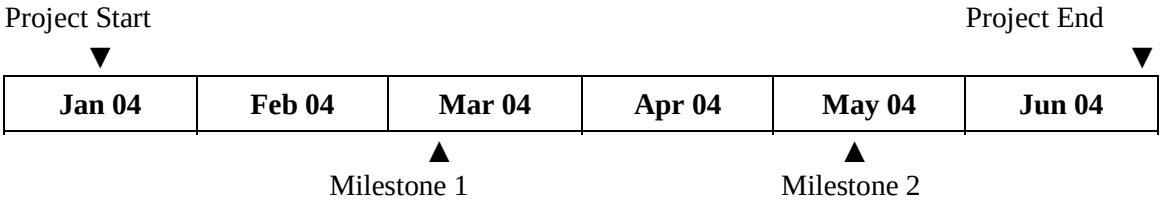
Estimated Effort hours: 3x10x24 = 720 person hours

No. of Team members = three
No. of hours per week per team member = 10 hours
No. of months = six (equivalent to 24 weeks)

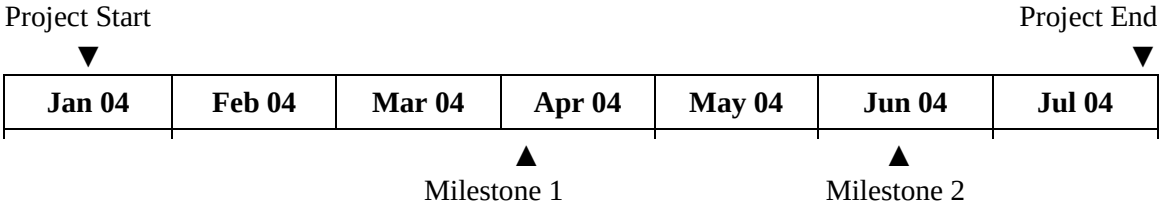
Project Planned Start Date: Fri 1/23/04
Project Planned Finish Date: Wed 6/23/04

The overall schedule of the planned project is given in detail in WBS. Here we illustrate with a simple timeline the overall schedule of the project.

Planned Schedule



Actual Schedule



- Project Start denotes the actual project start date of the rapid integration tools project.
- Milestone 1 implies the completion of Task1 - This included identification of tools and coming up with a classification scheme for them.
- Milestone 2 denotes the completion of Task2 - This includes identifying the model problem and getting hands on experience in evaluating the tools which would help in solving the model problem.

- Project End denotes the submission of the evaluation results in the form of a technical report and expressing the process of evaluation as a framework that can be extended to any model problem.

The line in the actual schedule denotes where we were when we were writing this technical report.

Similarly, the estimated effort into the project also increased from 720 person hours to 840 person hours.

Estimation:

The above WBS is based on the rapid integration tools document provided by the SEI before the start of the project. The project is divided into three tasks which have deliverables associated with each of them. The milestones are based completely on the three tasks. Each Task was allocated two months out of the total six months for the project.

Actual Progress:

However, as shown in the actual progress timeline, Task 1 took almost three months for completion, Task 2 took another three months to complete, and Task 3 is currently underway at the time of writing this report.

The primary reasons for schedule slippage are multiple commitments of team members on other projects, and the fewer number of hours allocated for the elective.

Table 8: Milestones and Schedule of the Project

Milestones	**Expected Date**	**Revised Date of Submission**	**Actual Date of Submission**
Task 1 - Survey and classify the tools- • List of Rapid Integration Tools	2/5/2004	6/7/2004	5/18/2004
• Classification Scheme	2/17/2004	6/09/2004	6/5/2004
• Classified List of Tools	3/02/2004	6/10/2004	6/5/2004
Task 2 - Evaluate the tools using a model problem • Preliminary Evaluation Scheme	3/19/2004	6/16/2004	
• Model Problem Definition	4/26/2004	6/16/2004	5/18/2004

• Reports detailing evaluation of tools' applicability to the model problem	5/11/2004	6/24/2004	5/18/2004
Task 3 - Develop and document general evaluation criteria • Documented evaluation criteria for rapid integration tools	6/1/2004	5/28/2004	

Glossary of Technical Terms

Term	Description
EAI	"Acronym for enterprise application integration. EAI is the unrestricted sharing of data and business processes throughout the networked applications or data sources in an organization. Early software programs in areas such as inventory control, human resources, sales automation and database management were designed to run independently, with no interaction between the systems. They were custom built in the technology of the day for a specific need being addressed and were often proprietary systems. As enterprises grow and recognize the need for their information and applications to have the ability to be transferred across and shared between systems, companies are investing in EAI in order to streamline processes and keep all the elements of the enterprise interconnected. There are four major categories of EAI: 1. Database linking: databases share information and duplicate information as needed. 2. Application linking: the enterprise shares business processes and data between two or more applications. 3. Data warehousing: data is extracted from a variety of data sources and channeled into a specific database for analysis. 4. Common virtual system: the pinnacle of EAI; all aspects of enterprise computing are tied together so that they appear as a unified application." http://www.webopedia.com/TERM/E/EAI.html
B2Bi	**Business-to-Business Integration**
Legacy System	"A computer system or application program which continues to be used because of the cost of replacing or redesigning it and often despite its poor competitiveness and compatibility with modern equivalents. The implication is that the system is large, monolithic and difficult to modify" http://computing-dictionary.thefreedictionary.com/Legacy%20system
Adapters	"Adapters and Connectors are pieces of software that are used in the integration of component-based applications and serve as a "wrapper" that mediates access to an application that was not developed with integration in mind, including legacy applications"

	http://eai.ittoolbox.com/nav/t.asp?t=346&p=347&h1=346&h2=347
Service-Oriented Integration	"Service-Oriented Integration (SOI) leverages open standards, loose coupling, and dynamic description and discovery capabilities of Web Services to reduce the complexity, cost, and risk of integration." http://www.zapthink.com/cluster.html?id=soi
Web Services	"Web Services refers to the technologies that allow for making connections. Services are what you connect together using Web Services. A service is the endpoint of a connection. Also, a service has some type of underlying computer system that supports the connection offered. The combination of services—internal and external to an organization—make up a service-oriented architecture." http://www.service-architecture.com/web-services/articles/web_services_definition.html
ALE	Stands for Application Embedding and Linking. "ALE allows behaviors between components and applications to be linked on a single-screen. Users are able to drill within applications, as well as from one application to another, without changing focus. ALE overcomes the limitations of HTML-based Web applications where any embedded link typically brings up a new page with no contextual link between the various Web pages." http://www.dharbor.com/products/psc_feat.html
JMX	"Java Management Extensions (JMX) technology provides the tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into those of the future " http://java.sun.com/products/JavaManagement/
CICS	"Short for Customer Information Control System, a TP monitor from IBM that was originally developed to provide transaction processing for IBM mainframes. It controls the interaction between applications and users and lets programmers develop screen displays without detailed knowledge of the terminals being used." http://www.webopedia.com/TERM/C/CICS.html
SOAP	"Short for Simple Object Access Protocol, a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before

	sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP.” http://www.webopedia.com/TERM/S/SOAP.html
IIOP	“Short for Internet Inter-ORB Protocol, a protocol developed by the Object Management Group (OMG) to implement CORBA solutions over the World Wide Web. IIOP enables browsers and servers to exchange integers, arrays, and more complex objects, unlike HTTP, which only supports transmission of text.” http://www.webopedia.com/TERM/I/IIOP.html
WSDL	“Short for Web Services Description Language, an XML-formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages. WSDL is an integral part of UDDI, an XML-based worldwide business registry. WSDL is the language that UDDI uses. WSDL was developed jointly by Microsoft and IBM.” http://www.webopedia.com/TERM/W/WSDL.html
LDAP	“Short for Lightweight Directory Access Protocol, a set of protocols for accessing information directories. LDAP is based on the standards contained within the X.500 standard, but is significantly simpler. And unlike X.500, LDAP supports TCP/IP, which is necessary for any type of Internet access. Because it's a simpler version of X.500, LDAP is sometimes called X.500-lite.” http://www.webopedia.com/TERM/L/LDAP.html
End-to-end Encryption	“The encryption of information at its origin and decryption at its intended destination without any intermediate decryption.” http://www.its.bldrdoc.gov/fs-1037/dir-014/_2016.htm
XML	“Short for Extensible Markup Language, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.” http://www.webopedia.com/TERM/X/XML.html
Microsoft CLR	“The Microsoft CLR Debugger is intended as an interim tool for debugging applications written and compiled for the common language runtime.”

	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cptutorials/html/the_net_sdk_debugger.asp
XSLT	"Short for Extensible Style Language Transformation, the language used in XSL style sheets to transform XML documents into other XML documents. An XSL processor reads the XML document and follows the instructions in the XSL style sheet, then it outputs a new XML document or XML-document fragment. This is extremely useful in e-commerce, where the same data need to be converted into different representations of XML. Not all companies use the exact same programs, applications and computer systems." http://www.webopedia.com/TERM/X/XSLT.html
SMTP	"Short for Simple Mail Transfer Protocol, a protocol for sending e-mail messages between servers. Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another; the messages can then be retrieved with an e-mail client using either POP or IMAP. In addition, SMTP is generally used to send messages from a mail client to a mail server. This is why you need to specify both the POP or IMAP server and the SMTP server when you configure your e-mail application." http://www.webopedia.com/TERM/S/SMTP.html
HTTP	"Short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page." http://www.webopedia.com/TERM/H/HTTP.html
PKI	"Short for public key infrastructure, a system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction. PKIs are currently evolving and there is no single PKI nor even a single agreed-upon standard for setting up a PKI. However, nearly everyone agrees that reliable PKIs are necessary before electronic commerce can become widespread." http://www.webopedia.com/TERM/P/PKI.html
J2EE	"Short for Java 2 Platform Enterprise Edition. J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applica-

	tions.” http://www.webopedia.com/TERM/J/J2EE.html
JSP	“Short for Java Server Page. A server-side technology, Java Server Pages are an extension to the Java servlet technology that was developed by Sun. JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements -- the actual design and display of the page—to help make the HTML more functional (i.e., dynamic database queries).” http://www.webopedia.com/TERM/J/JSP.html
JCA	“The J2EE Connector architecture provides a Java technology solution to the problem of connectivity between the many application servers and today's enterprise information systems (EIS).” http://java.sun.com/j2ee/connector/overview.html
EJB	“Enterprise JavaBeans (EJB) is a Java API developed by Sun Microsystems that defines component architecture for multi-tier client/server systems. EJB systems allow developers to focus on the actual business architecture of the model, rather than worry about endless amounts of programming and coding needed to connect all the working parts. This task is left to EJB server vendors. Developers just design (or purchase) the needed EJB components and arrange them on the server.” http://www.webopedia.com/TERM/E/Enterprise_JavaBeans.html
JAAS	“The Java Authentication and Authorization Service (JAAS) is a set of APIs that enable services to authenticate and enforce access controls upon users. It implements a Java technology version of the standard Pluggable Authentication Module (PAM) framework, and supports user-based authorization.” http://java.sun.com/products/jaas/
Aspect Oriented Programming	“Aspect-oriented programming (AOP) is a new programming technique that allows programmers to modularize crosscutting concerns (behavior that cuts across the typical divisions of responsibility, such as logging). AOP introduces aspects, which encapsulate behaviors that affect multiple classes into reusable modules.” http://www-106.ibm.com/developerworks/java/library/j-aspectj/

References/Bibliography

URLs are valid as of the publication date of this document.

[Abst 00] Abst, C.; Boehm, B.; & Clark, B. *COCOTS: A Software Integration Cost Model*. Los Angeles, CA: USC Center for Software Engineering 2000. http://sunset.usc.edu/publications/TECHRPTS /2000 /usccse2000-501/usccse2000-501.pdf

[Brownsword 04] Brownsword, L.; Carney, D. J.; Fisher, D.; Lewis, G.; Meyers, C.; Morris, E. J.; Place, R. H.; Smith, J.; & Wrage, L. *Current Perspectives on Interoperability* (CMU/SEI-2004-TR-009, ADA421613). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents /04.reports/04tr009.html

[IONA 02] ONA Technologies PLC. *IONA's Orbix E2A Enterprise Integration Technology*. Waltham, MA: IONA Technologies PLC, 2002.

[Jaccheri 02] Jaccheri, L. & Torchiano, M. "Classifying COTS Products," 246-255. *Software Quality – ECSQ 2002. Quality Connection – 7th European Conference on Software Quality*. Helsinki, Finland, June 9-13, 2002. Proceedings, *Lecture Notes in Computer Science*, 2349. Berlin, Germany: Springer-Verlag, 2002.

[Johnson 88] Johnson, A. M. & Malek, M. "Survey of Software Tools for Evaluating Reliability, Availability, and Serviceability." *Computing Surveys*, *20*, 4, (Dec. 1988) 227-269.

[Kasunic 03] Kasunic, Mark. "Measuring Systems Interoperability." *Conference on the Acquisition of Software-Intensive-Systems*. Arlington, VA, January 28-30, 2003. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University 2003.

[Ncube 02] Ncube, C. & Dean, J. "Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components," 176-187. *Proceedings of the First International Conference of COTS-Based Software Systems, ICCBSS 2002.* Orlando, FL, February 4-6, 2002. Berlin, Germany: Springer-Verlag, 2002.

[Powell 97] Powell, A., et al. "Evaluating Tools to Support Component Based Software Engineering, 80-89. *Proceedings of the Fifth International Symposium on Assessment of Software Tools and Technologies*. Pittsburgh, PA, June 2-5, 1997. Los Alamitos, CA: IEEE Computer Society Press, 1997.

[Sai 04] Vijay, S. COTS *Acquisition Evaluation Process: Preacher's Practice* (CMU/SEI-2004-TN-001, ADA421675). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents /04.reports/04tn001.html

[Schmidt 02] Schmidt, Marty. J. *Business Case Essentials: A Guide to Structure and Content.* West Sussex, UK: Matrix Solutions Ltd, 2002. http://www.eitoolkit.com/tools/initiation /Business_Case_Essentials.pdf

[Simon 03] Simon, Jonathan. "Case Study: Bond Trading System." Chapter 13, Hohpe, Gregor & Woolf, Bobby. Enterprise Integration Patterns. Boston, MA: Addison-Wesley, 2004.

[Vu 02] Vu, John. "The eCommerce Capability Model (eCCM): A Framework To Implement eCommerce Successfully."(tutorial). *SEPG 2000: Software Engineering Process Group Conference*, Phoenix, AZ, February 18-21, 2002.

[Wallnau 91] Wallnau, K. & Feiler, P. *Tool Integration and Environment Architectures* (CMU/SEI-91-TR-011, ADA237810). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1991. http://www.sei.cmu.edu/publications/documents /91.reports /91.tr.011.html

[Wallnau 02] Wallnau, K.; Hissam, S.; & Seacord, R. *Building Systems from Commercial Components.* Boston, MA: Addison Wesley, 2002.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	December 2004	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Rapid Integration for Rapid Application Development	F19628-00-C-0003

6. AUTHOR(S)

Amit Midha, Ravindra Singh, Lakshmi Pratha Hari

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2004-TR-023

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2004-023

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report investigates the rapid integration tools available in the current market. These tools aid in the rapid integration of software systems and components. The research centers on a model problem that requires such a tool to address legacy integration challenges. The report presents a generic evaluation framework for identifying and evaluating rapid integration tools and an evaluation of three identified tools. This evaluation engaged selected evaluation criteria based on the demands of the model problem. A process reference is also included; this forms the guidelines for identification and evaluation of the tools with respect to other model problems.

14. SUBJECT TERMS	15. NUMBER OF PAGES
rapid integration, integration tools, rapid application development	107

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL