A Process for Context-Based Technology Evaluation

Grace A. Lewis
Lutz Wrage

June 2005

Integration of Software-Intensive Systems (ISIS) Initiative

Technical Note
CMU/SEI-2005-TN-025

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.

Contents

Abstract vii

1 Context-Based Technology Evaluation 1

1.1 Step 1: Identify Technology Usage Context and Evaluation Goals 2

1.2 Step 2: Plan The Evaluation 4

1.2.1 Form Evaluation Team 4

1.2.2 Identify Stakeholders 4

1.2.3 Select an Approach 5

1.2.4 Estimate Effort and Schedule 5

1.3 Step 3: Develop Model Problems 5

1.3.1 Develop Hypotheses 6

1.3.2 Develop Criteria 6

1.3.3 Design Model Solution 7

1.3.4 Implement and Evaluate Model Solution Against Criteria 9

1.4 Step 4: Analyze Model Problem Results Against Technology Usage Context 9

2 A Case Study in Evaluation of Web Services Technology 11

2.1 Identify Technology Usage Context and Evaluation Goals 11

2.2 Plan The Evaluation 12

2.3 Model Problem Setup 14

2.3.1 Model Problem Hypotheses and Criteria 14

2.3.2 Model Solutions 14

2.3.3 Model Solution Architecture 17

3 Conclusions 21

Bibliography 23

List of Figures

Figure 1: Context-Based Technology Evaluation Method....................................... 2

Figure 2: Model Solution for Hypothesis 1... 8

Figure 3: Model Solution for Hypothesis 2... 8

Figure 4: Platform/Language Application Distribution for Current Applications 15

Figure 5: Component and Connector View for the Model Solution 17

Figure 6: Deployment View for the Model Solution 19

List of Tables

Table 1: Sample Hypotheses and Associated Criteria 6

Table 2: Stakeholders and Responsibilities for Case Study Technology Evaluation 13

Table 3: Hypotheses and Criteria for Case Study Model Problems 14

Abstract

In order to determine a fit between systems and technology, it is necessary to evaluate technologies within the contexts that they will be used. This report describes a process called context-based evaluation that determines the fitness of a technology within a specific context. It includes hands-on experimentation with the technology for a greater understanding of its implications, as well as early competence development of the people conducting the experiments. An integral part of the process is the development of model problems; these are prototypes, situated in a specific context, with the goal of satisfying evaluation criteria.

The focus of this report is on evaluation of software technologies, such as Web services, database systems, or architectural frameworks and development tools. The report also includes a case study of the use of this process for the evaluation of Web service technology.

1 Context-Based Technology Evaluation

In order to determine a fit between systems and technology, it is necessary to evaluate technologies within the context that they will be used. All technologies work well within a specific context and under certain conditions. For example, Web services work well for asynchronous communication over the Internet. In a business environment these conditions are very common. However, this may not be the case in a military tactical command and control environment where high performance and availability requirements prevail.

The process outlined in Figure 1 is a context-based technology evaluation process for determining the fitness of a technology within a specific context. An integral part of the process is the development of model problems, a technique initially codified for building systems from commercial components. This technique prescribes the use of focused hands-on experimentation as a way of reducing the risk associated with component integration [Wallnau 01, Lewis 04a]. The process is also largely based on the PECA process for the evaluation of commercial off-the-shelf (COTS) software products [Comella-Dorda 04].

The focus of this report is on evaluation of software technologies, such as Web services, database systems, or architectural frameworks and development tools. We constrain the scope of the evaluation to technical issues and not the business, legal, and other concerns that are important in their own right but out of scope for this evaluation process. We believe these additional issues become important in the product evaluation and selection stage, but not in the experimentation and exploratory stage for which this process is intended. Here, non-technical concerns are treated as constraints.

The next sections provide detail on each of the activities included in the context-based technology evaluation process.

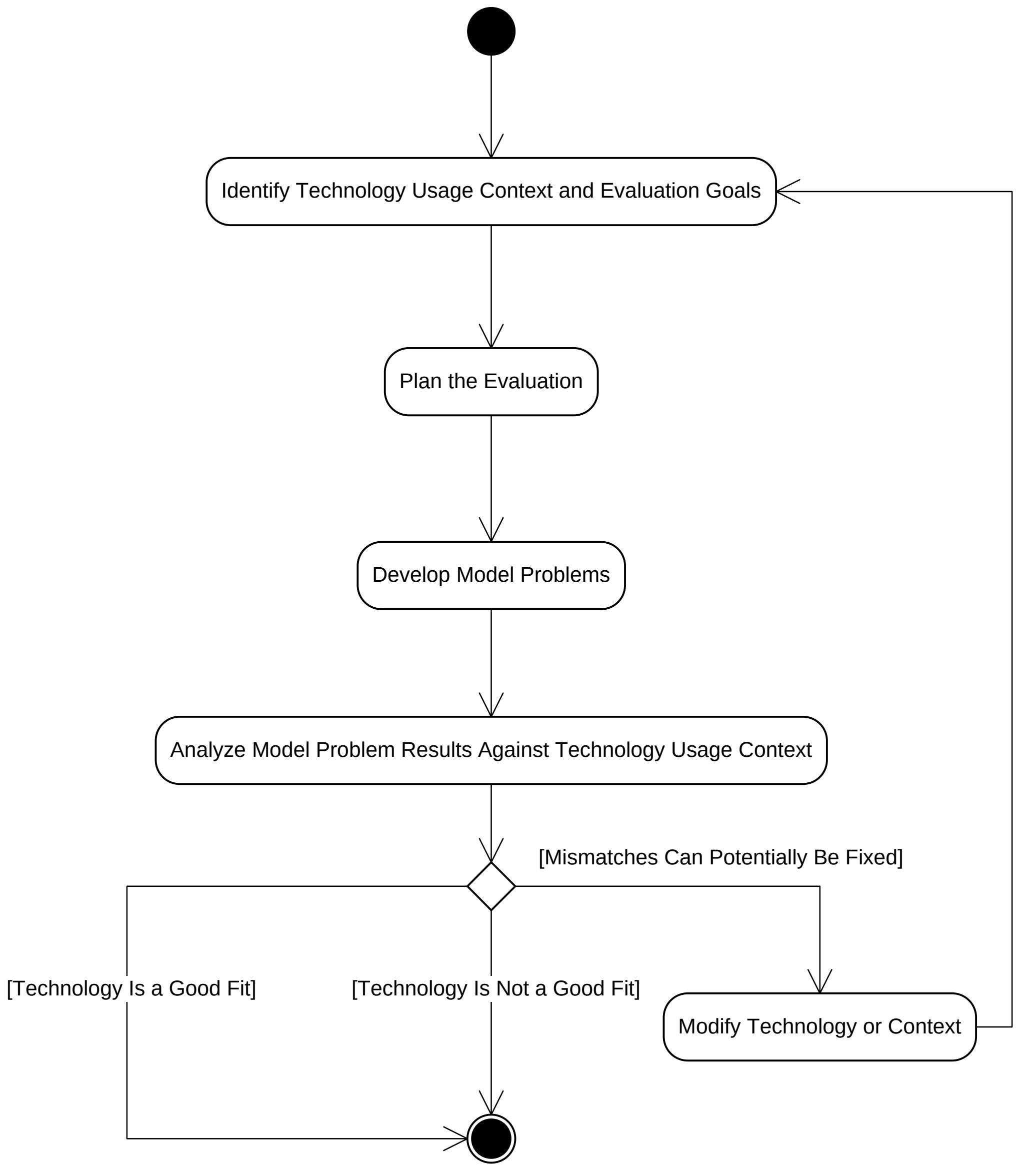
Identify Technology Usage Context and Evaluation Goals
Plan the Evaluation
Develop Model Problems
Analyze Model Problem Results Against Technology Usage Context
[Mismatches Can Potentially Be Fixed]
[Technology Is a Good Fit]
[Technology Is Not a Good Fit]
Modify Technology or Context

Figure 1: Context-Based Technology Evaluation Method

1.1 Step 1: Identify Technology Usage Context and Evaluation Goals

The goal of this step is to determine and document the organization's reasons for conducting the evaluation, what is expected of the evaluation, what the expectations are for the technology, and what constraints, if any, must the evaluators adhere to. The evaluation team will then use the answers to these questions to describe the envisioned state of the organization and its operations if the new technology were implemented.

An organization may have many reasons for undertaking a technology evaluation. It is important that the evaluation team be aware of these reasons, so as to ask and answer the right questions throughout the evaluation effort. Depending on the particular situation, the team will need to address a number of concerns. The following examples illustrate this.

- A bank currently uses a hierarchical database that has been heavily modified over the years. It now wants to evaluate relational database technology to partly replace the legacy database. One major concern is performance.
- A manufacturing company wants to upgrade its communication mechanism with external providers from Electronic Data Interchange (EDI) to Web services. The evaluation is mainly driven by concerns about interoperability with internal and external systems.
- An organization is exploring the introduction of e-mail encryption technology to secure electronic communication. The evaluation is mainly focused on usability, as the majority of their users are non-technical.
- An organization wants to explore future options and possibilities using open-source project management tools. The concern is whether these open-source tools will provide functionality similar to that of the current commercial tool.

It is also important to know what the scope of the evaluation effort is, the budget, and the expected time frame for delivering results. In many evaluations the scope will consist of just one technology to evaluate, but there are also situations where the evaluation concerns a choice between several completing technologies.

A list of expectations for the technology will provide an overall picture of the intended benefits the new technology may bring to the organization. Expectations include both requirements the technology must or should fulfill, and other benefits, such as improved quality of service, that are not necessarily required but can make adopting the technology worthwhile. Requirements may be further subdivided into requirements involving functionality and those that involve quality attributes, such as performance, usability, ease of development, or maintainability. These expectations are revisited in the last step of the process to determine whether they have been met to a sufficient degree to go ahead with adopting the technology.

A system’s overall context may sometimes prove inflexible, thus placing constraints on the technology. Examples include legal requirements or organizational policies. A legal requirement will always place a constraint on technology use, whereas it might be possible to change a policy if so justified by the potential benefits of a new technology. If it is not possible for a technology to meet a constraint, then the technology cannot be used in the organization.

The evaluators will put expectations and constraints together to describe the envisioned state of the organization and its operations should the technology in question be implemented.

In summary, this task will provide answers to some basic questions, including

- What is the environment in which the technology will be used?
- What are the expectations for the technology?

- What are the goals of the evaluation?
- What is the scope of the evaluation?
- What decision authority will the evaluation team have?
- What would be a successful evaluation?
- What constraints must the evaluation team adhere to?
- In what ways is the system context flexible?

1.2 Step 2: Plan The Evaluation

The next step is to plan the evaluation. The details of every evaluation plan will differ based on the scope of the evaluation and the potential impact of the new technology on the business. Evaluating only one technology will require less effort than comparing multiple technologies, and the higher the impact the more detailed the evaluation will need to be. However, the following major activities will probably be part of most evaluation plans:

- Form an evaluation team.
- Identify stakeholders.
- Select an approach.
- Estimate effort and schedule.

1.2.1 Form Evaluation Team

Most importantly, the evaluation team must consist of members who have the technical and domain expertise needed to evaluate the technology in the system context. Domain experts will contribute information about the context in which the technology will need to function. Technical experts will delve into the details of the technology and to create and implement the model problems. Model problems are described in detail in Section 1.3.

It is also important to compose a team with enough authority and credibility to ensure the results are accepted in the organization. The higher the impact of the technology on the organization's business or mission, the more authority and credibility must be manifest in the evaluation team. Talking with supervisors to get commitment from the members of the evaluation team is crucial at this point to assure their dedication to the team.

1.2.2 Identify Stakeholders

The evaluation team must consider the points of view of all stakeholders in the evaluation effort. Different stakeholders will have different goals, expectations, and concerns regarding a new technology, and must be identified early enough to gather their input in the planning phase. Stakeholder input will influence the evaluation goals and scope and have expectations on the technology and the evaluation itself. Examples of stakeholders are end users of the technology, system developers and maintainers, system architects, managers, administrative staff, change agents, and sponsors.

1.2.3 Select an Approach

Based on the technology usage context and evaluation goals, a set of high-level tasks for the evaluation is defined. Any work that must be conducted before the definition of the model problems in the next step, such as surveys, purchases, initial investigations, and conversations with stakeholders, is included. Any reports or briefings that form part of the outcome of the evaluation are also included as tasks.

Even though most technology evaluations will be performed on one technology, in a situation where the team evaluates competing technologies and the goal of the evaluation is to choose one technology for potential implementation, it is also necessary to decide if the team should follow a best-fit or first-fit strategy. In a first-fit approach the evaluation continues until one technology has been identified that meets all expectations and constraints. In contrast, in a best-fit approach the team evaluates all technologies in scope and chooses the technology that meets all constraints and delivers most benefit to the organization. These benefits are measured by the degree to which the technology can fulfill the expectations of the stakeholders. In this case the team must also define how this degree will be measured and the relative weight to be assigned to each expectation. The selected strategy will affect the set of tasks and the length of the evaluation.

1.2.4 Estimate Effort and Schedule

Based on the list of tasks defined in the previous activity and the composition and skill set of the evaluation team, a preliminary plan is put together for the evaluation, indicating the effort required for each task as well as an overall schedule.

1.3 Step 3: Develop Model Problems

Model problems are prototypes, situated in a specific context, with the goal of satisfying evaluation criteria [Wallnau 01]. In the case of technology evaluation, the purpose of using model problems is to conduct situated hands-on experimentation with the technologies, given the context of how the technologies will be used. Scenarios are constructed from the given context to provide a context-based technology fitness evaluation.

The steps for developing model problems are as follows:

1. Develop hypotheses.
2. Develop criteria.
3. Design model solution.
4. Implement and evaluate model solution against criteria.

1.3.1 Develop Hypotheses

The first task is to develop hypotheses. These are derived from the expectations placed on the technology. Hypotheses are claims about the technologies that are to be sustained or refuted. Hypotheses provide the focus and structure that is necessitated by the expense involved in hands-on experimentation.

For Web services technology,[1] for example, valid hypotheses that correspond to a typical Web services context might be

- It is fairly easy for a developer to connect components developed for the same platform using Web services.
- There will be no problems regarding data types if Web services are used to connect components developed for different platforms, specifically J2EE and .NET [Sun 05, Microsoft 05a].

1.3.2 Develop Criteria

Criteria are used to determine if a model solution sustains or refutes a hypothesis. Criteria are derived from requirements and are stated as a clearly measurable statement of capability. Each hypothesis can have one or more criteria, depending on the breadth covered by the hypothesis. Table 1 illustrates valid criteria for the above hypotheses.

Table 1: Sample Hypotheses and Associated Criteria

Hypothesis	Criteria
It is fairly easy for a developer to connect components developed for the same platform using Web services.	1. There are well-documented tools for generating Web Services Definition Language (WSDL) documents. 2. There are SOAP libraries that can be linked into existing applications. 3. If the IDE does not provide facilities for doing so, generating the Web services code requires (1) on the sender side, nothing more than generating an XML document corresponding to the WSDL definition of the Web service and wrapping it as a SOAP message; (2) on the receiver side, nothing more than to extract the contents from the SOAP message and parse the appropriate XML document.

[1] A *service* is a coarse-grained, discoverable, and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model. A Service-Oriented Architecture (SOA) is a collection of services with well-defined interfaces and a shared communications model. Web services are the most common form of SOA, in which service interfaces are described using Web Services Description Language (WSDL), payload is transmitted using Simple Object Access Protocol (SOAP) over Hypertext Transfer Protocol (HTTP), and optionally Universal Description, Discovery and Integration (UDDI) is used as the directory service [Lewis 04b].

There will be no problems regarding data types if Web services are used to connect components developed for different platforms (i.e., J2EE and .NET).	The two applications will exchange a complex data type, a date, and a floating point data type (known to cause problems) with no additional effort other than that indicated in Criterion 3 for the previous hypothesis.

1.3.3 Design Model Solution

The model solution should be very efficient—it only needs to answer the question at hand. The model solution should also be realistic—it is framed within a specific scenario derived from the context. Therefore, a model solution is the simplest set of applications and/or components that are able to answer the questions posed by the hypotheses and associated criteria, within a given context. Designing a model solution will often require preparation work such as acquiring and installing components, performing interviews and surveys, or becoming quickly acquainted with the technology in order to propose a model solution.

Let us say, for example, that the context is a typical human resources arena, the organization's development environment is Java using the Eclipse IDE [Eclipse 05a], and there is knowledge of existing .NET applications that will expose functionality as Web services. The model solutions to evaluate the above hypotheses might be those described with the scenarios below:

Hypothesis 1: It is fairly easy for a developer to connect components developed for the same platform using Web services.

Scenario: An organization has a Human Resources (HR) system that maintains personnel records. Training information is kept in a separate system with which the HR system must interact, using Web services to update and retrieve training information.

Model Solution: A simple J2EE HR application that performs create-retrieve-update-delete (CRUD) operations on personnel basic data (social security number, name, address, salary) is created using the Java Eclipse IDE. Tomcat is the Servlet engine, the J2EE application server is JBoss, and data is stored in an Oracle database [Apache 05a, JBoss 05, Oracle 05]. A second J2EE application that keeps track of basic personnel training information (course name, date taken, location) is created using the same IDE and deployed on the same platform. A Web service with two operations is created in the training application using Eclipse plug-ins from the Web Standard Tools Platform (WST) subproject [Eclipse 05b]. One operation submits new training information to the training application and the other operation returns training information for a given social security number. The HR application has an option for invoking the Web service operations. Figure 2 illustrates this model solution.

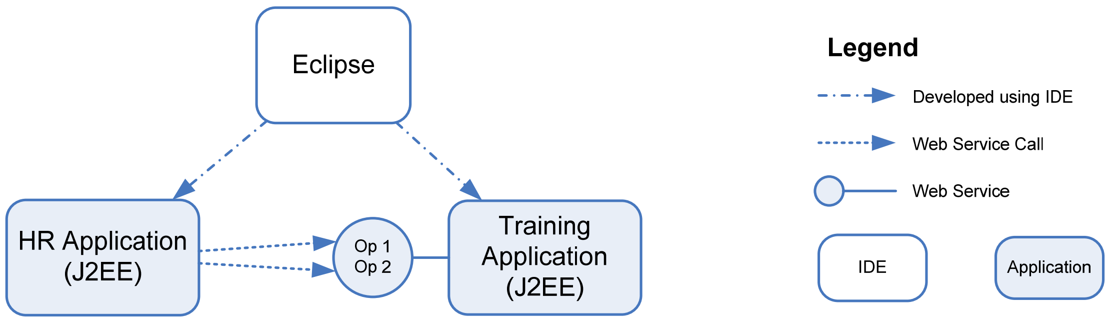

Figure 2: Model Solution for Hypothesis 1

Hypothesis 2: There will be no problems regarding data types if Web services are used to connect components developed for different platforms (i.e., J2EE and .NET).

Scenario: The HR system, a J2EE application, interacts with the payroll system, a .NET application, using Web services to report new employees and salary changes. The payroll system interacts with the HR system, also using Web services to indicate that the salary change has been processed.

Model Solution: A payroll application is generated using Visual Studio .NET [Microsoft 05b]. The payroll application has a basic user interface that displays employee information and payroll-related events for that employee. A Web service with two operations is added to the payroll application: one to receive new employee records (a complex data type) and one to receive salary changes (social security number, a floating point type representing a percentage, and a date by which they should take effect). The HR application has an option for invoking the Web service operations. A Web service with an operation that receives salary change notifications (social security number, date processed, and a floating point representing the amount of the new salary) is added to the HR application. A simple user interface to record salary changes and view notifications is also added to the HR application. The payroll application has an option for processing the salary change and invoking the notification Web service. Figure 3 illustrates this model solution.

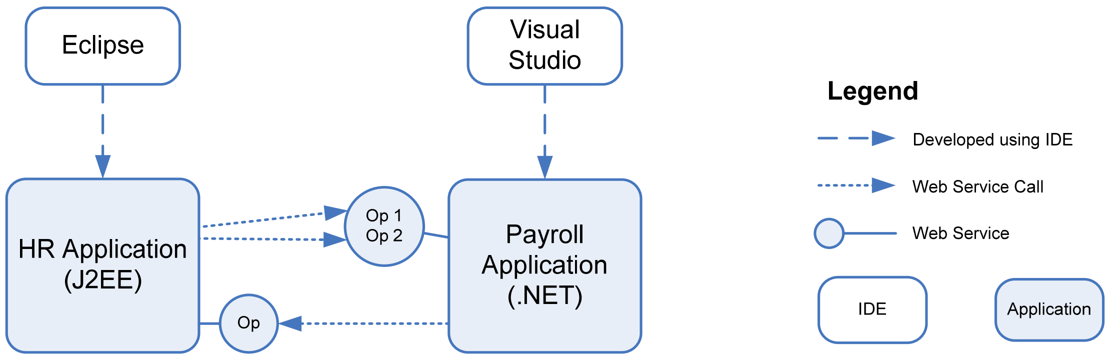

Figure 3: Model Solution for Hypothesis 2

Given that there is a better idea of details of model problems at this point, it is a good idea to revise the overall plan for the evaluation that was put together in the previous step.

1.3.4 Implement and Evaluate Model Solution Against Criteria

In this activity, the model solution is implemented, run, and observed, until there is enough information to sustain or refute the set of hypotheses. Criteria sometimes require refining because they are too vague, making it difficult to decide whether the hypothesis is sustained or refuted. For example, Criterion 1 for Hypothesis 1 can be refined to indicate that not only should there be tools to generate WSDL documents, but also tools to generate code from WSDL documents.

A simple but effective way to capture experience and knowledge is to keep notes about product installation, problems encountered, interim results, deployment environment, and any information that might help future projects, should the technology be selected for use in the organization. The simplest mechanism to capture this information is through an informal blog that can be shared among all people participating in the model problem process.

1.4 Step 4: Analyze Model Problem Results Against Technology Usage Context

The last step in the process is to make a final determination with respect to the fitness of the technology for the context in which it is intended to be used. Model problems provide increased negotiation power and informed decision making because of the direct hands-on experimentation with the technologies.

Based on the results of the evaluation there should be enough information to determine if

- the technology is a good fit with requirements
- the technology is not a good fit with requirements
- the technology has some mismatches that could potentially be solved by modifying the context or modifying the technology itself

If the model solution provides information that sustains the set of hypotheses, and assuming sufficient confidence in the results, the technology can be declared a good fit with the requirements. Given that the model solution is a simplification of the real solution, it should be determined whether the results can scale so that they are valid in the larger context as well.

If the model solution provides information that refutes the set of hypotheses, and there is no room for negotiation, or very strict constraints that are not met, it can be stated that the technology is not a good fit with the requirements. In this case, it may be possible to find a different technology that will satisfy the requirements. The process should then be repeated, reusing the previously produced information and results as much as possible.

Most often, the model solution will provide information that sustains some of the hypotheses and refutes others. In this case of mismatch, it is useful to determine if either the context or the technology itself can be modified. Some examples of context modification are

- Reset technology expectations.
- Relax constraints.
- Negotiate technology requirements with stakeholders.

Sometimes it is also possible to modify the technology. Some examples follow:

- In the case of open source technologies, an option would be to modify the technology, making sure that the modifications are included in the main release line of the technology. This of course requires access to source code; the cost-benefit of this option must also be evaluated. The risk of making modifications that are not fed back into the open source project is an inability to take advantages of new releases, unless the same modifications are made to every new release.
- Investigate complementary technologies that in combination would satisfy the hypotheses.
- Negotiate the possibility of performing modifications with the vendor, open source community, or standards organization. In the case of the first two, it is important to make sure that modifications are included in the main release line of the technology. This is usually easier with small vendors or small projects. In the case of standards organizations, there is usually a formal process to put forth modifications that can involve a lengthy approval time.

2 A Case Study in Evaluation of Web Services Technology

The following case study is an example of how the context-based technology evaluation process could be used in the evaluation of Web services in an organization that provides business intelligence and data mining services to its registered users. The case study does not include the actual technology evaluation.

2.1 Identify Technology Usage Context and Evaluation Goals

Technology Usage Context

An organization is migrating from a federated suite of legacy applications and data stores, used for providing business intelligence data mining services, to a solution based on Web services. The plan is for most of the “as-is” systems to migrate via their adaptation to Web services. However, some of the older legacy systems will have to be replaced in order to fit with Web services, or eliminated, if no longer needed.

The current systems support approximately 2,000 simultaneous users. Typical queries are serviced with response times of a few (2-10) seconds, and some complex queries take as long as two minutes. These response times are acceptable by current users. Query results are provided in a variety of pre-defined and user-specified formats, such as database tables, spreadsheets, or advanced data visualization including high-resolution images.

Access to data is determined by business area. For example, users from customer relations management should not have access to long-term strategic company forecast data.

There is a requirement that the new Web-services-based system support single “sign-on” for users, rather than the current jumble of separate logins/passwords for different systems.

The organization—while new to Web services—has been in the software development business long enough to realize the significant gap that often exists between technologies touted as mature versus those that are mature in reality. The organization’s main concerns are the feasibility of adapting its current systems, maintaining (or improving) current response time, allowing image transfer, and providing single sign-on.

Goals of the Evaluation

The goals will be to determine how Web services use will affect the

- percentage of systems that can be adapted
- ability to maintain or improve current response time
- feasibility of transmitting images as part of messages
- possibility of having single sign-on

Scope of the Evaluation

The evaluation will be limited to the feasibility of using Web services as a modernization option for the set of legacy systems, limiting the evaluation to the four goals listed above. Other aspects of Web services are out of scope for the present evaluation.

Expectations for the Technology

- The use of Web services will ease the eventual replacement of legacy applications due to a common interface and will provide a single point of entry and user interface for all applications.
- Single sign-on will reduce the amount of work necessary for current users when they need data from more than one application.

Constraints

- Since this is a feasibility study, there is no budget for the purchase of new software. All software should be freeware, open source, or already licensed for the organization.
- Java-based products are preferred given the current IT mandate to move towards Java as the development and production environment. Eclipse is the current Integrated Development Environment (IDE) tool.

2.2 Plan The Evaluation

Evaluation Team

The evaluation team will be composed of

- a project lead
- a member from the core information technology (IT) group
- two members from the maintenance group
- a member from the development group

Each member of the evaluation team is expected to contribute 50% of her or his time for two months. Direct supervisors will be requested to sign a commitment form to assure continued participation on the team.

Stakeholders

Stakeholders and their responsibilities are shown in Table 2. All stakeholders will be informed of the goals and dates of the evaluation, as well as possible requests for interviews and/or availability of personnel for receiving input.

Table 2: Stakeholders and Responsibilities for Case Study Technology Evaluation

Stakeholder	Technology requirements	Sponsorship, administration	Technical information	Negotiation	System inventory
IT Manager		X		X	
Software Development Manager		X			
Business area representatives	X			X	
Current system developers			X		
Current system maintainers			X		
IT personnel			X		X

Approach

A system inventory will be performed before the evaluation starts, in collaboration with IT, to determine the number of systems in use and some basic characteristics regarding their potential for adaptation to Web services.

An Internet search will be conducted to select free/open-source software to satisfy the model solution. Some of the software elements below will be necessary for more than one platform, depending on the results of the systems inventory and the selected deployment configuration.

- HTTP Server
- SOAP engine
- Tool to convert from a WSDL document to code, to handle Web service requests in selected programming languages
- SOAP libraries

Model problems will be developed to address the goals of the evaluation.

2.3 Model Problem Setup

2.3.1 Model Problem Hypotheses and Criteria

Hypotheses and criteria for the model problems are shown in Table 3.

Table 3: Hypotheses and Criteria for Case Study Model Problems

Hypothesis	Criteria
75% or more of existing applications can be adapted to Web services.	1. There are SOAP libraries that can be easily linked into 75% of existing applications. 2. If not, there are clear mappings between native data types and XML Schema data types that will allow for manual serialization and deserialization of SOAP messages.
Response time will not be degraded due to the use of Web services.	Response times using Web services will be within the same order of magnitude from current response times for representative applications.
Images can be transmitted as part of SOAP messages.	An image can be requested using Web services and viewed on a client application.
It is possible to have single sign-on using Web services.	A user is able to login once and obtain information from three different Web services residing on three different machines.

2.3.2 Model Solutions

Hypothesis 1: 75% or more of existing applications can be adapted to Web services.

Preparation Work: A system inventory was conducted, gathering the following information for all applications: business area, function, name, technology (data storage, platform, programming language), and development type (commercial or in-house). The distribution of platforms/programming languages is shown in Figure 4.

Scenario: A user from one of the business areas needs data from nine different sources. A client application, using Web services, is able to obtain the data from the data sources that are located on different applications on different machines.

Model Solution: An application that is representative of each of the groups in Figure 4 is targeted for adaptation to Web services. The simplest operation available from the application should be selected for this model solution. If there is no such simple operation, a simple application is developed by one of the application maintainers that simply returns a set of data containing representative data types (integer, floating point, date, string). A Java client application is developed to connect to each of the Web services and display the results. These results are manually compared to the results from the current requests to the applications. Tools are acquired to help in the task, if available. Time is tracked for the adaptation and client development tasks. Interviews with maintainers should help determine whether the effort can in fact be generalized to the rest of the applications in each group. The architecture for the solution is presented in Section 2.3.3.

Hypothesis 2: Response time will not be degraded due to the use of Web services.

Scenario: Same as for Hypothesis 1.

Model Solution: Using the same representative applications as for Hypothesis 1, response time is measured for the current requests to those applications. The client application developed for Hypothesis 1 is instrumented to show response times for each of the requests. To make the results comparable, all times are collected during similar network traffic periods. Time permitting, the different deployment options presented in Section 2.3.3 should be tested to see how performance is affected.

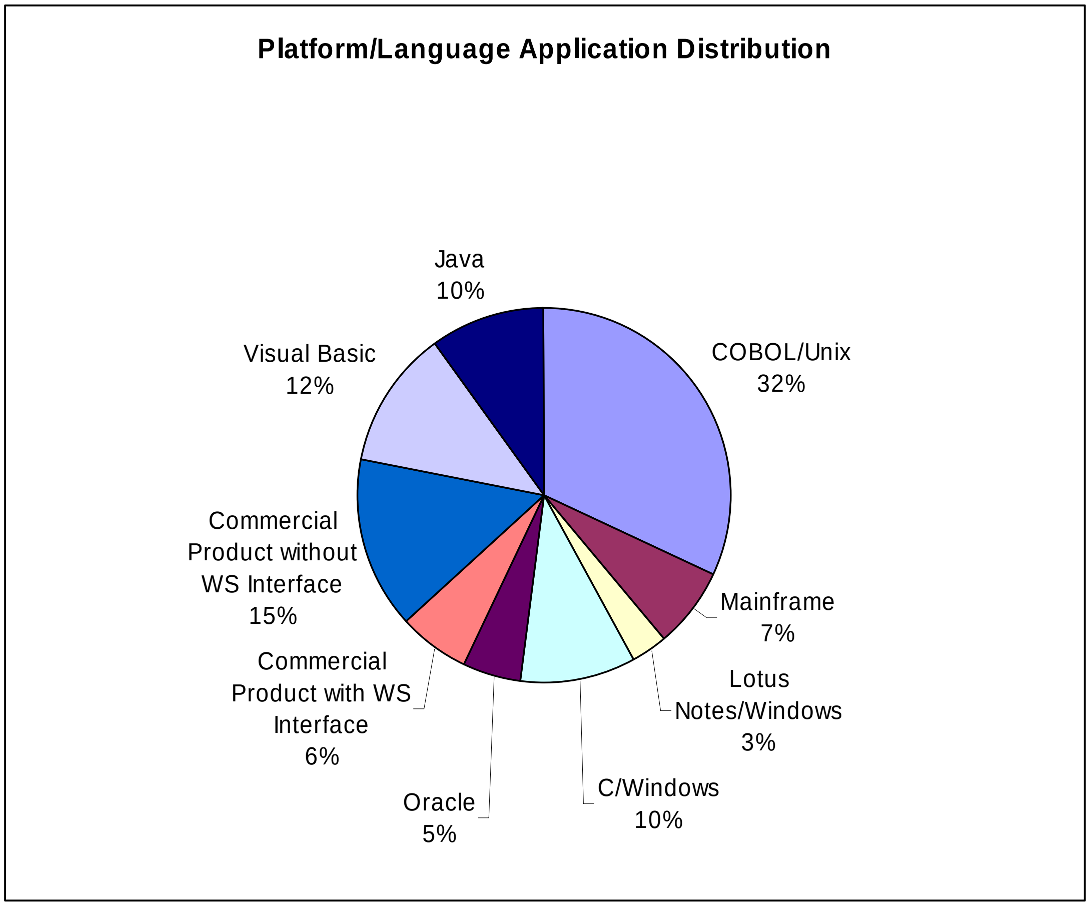

Figure 4: Platform/Language Application Distribution for Current Applications

Hypothesis 3: Images can be transmitted as part of SOAP messages.

Scenario: A user from one of the business areas requests data that is returned as an image.

Model Solution: The representative Java application is extended with an operation that returns a high-resolution image. The size of this returned image is comparable to those returned by the current legacy applications. The new operation is also made available as a Web service where the binary image data is added to a SOAP message as a binary attachment. The client application is extended with an option that invokes this new operation. The rationale for using the Java application is the availability of tools that make the task easier and faster (e.g., Apache Axis) [Apache 05b]. Apache Axis provides preliminary support for the *SOAP with Attachments* specification that allows SOAP messages to carry attachments, including images [W3C 00]. The team has also considered the newer Message Transmission Optimization Mechanism (MTOM) specification but has decided not to pursue this technology because the specification is still very new (January 2005) and therefore implementations are not yet available [W3C 05].

To support additional performance measurements (Hypothesis 2) it may be necessary to select images of different sizes as return values.

Hypothesis 4: It is possible to have single sign-on using Web services.

After initial discussions, the evaluation team realizes that a thorough evaluation of single sign-on technologies would involve sufficient complexity to justify a separate technology evaluation effort. The team has identified three potential specifications pertaining to single sign-on using Web services: Security Assertion Markup Language (SAML), WS-Security, and Web Services Security (based on WS-Security) [OASIS 05, IBM 02, OASIS 04]. However, preliminary research shows that none of these directly provides single sign-on; rather, they enable technologies that may be used in a complete single sign-on solution. Given the different options and technologies available, it would be impossible to analyze these in the given time frame.

Instead, the team decides to evaluate an interim solution based on MS Windows authentication. This solution should be feasible, since all users use MS Windows on their desktop computers, and the proposed architecture for the model solution has all HTTP Servers and SOAP engines running on Microsoft Windows on the local network. The stakeholders agree with this interim solution and are aware that, depending on the model problem results, it may be necessary to perform a separate evaluation for single sign-on technologies.

Scenario: The scenario is the same as for Hypotheses 1 and 2. Users authenticated to the Windows operating system on their workstation do not need to log in to the individual applications.

Model Solution: The Java client is extended to tie into Windows NT LAN Manager (NTLM) authentication of the user's workstations which all run MS Windows [Microsoft 05c]. The Web servers hosting the Web services are configured to restrict access to services based on authentication tokens provided by the client with the service requests. Depending on the legacy applications' authorization policy, each Web service will be modified to either access the application with full rights to all data and functions or impersonate individual users. In the latter case the implementation of the Web service is extended with a new component that authenticates Web service users with the legacy applications. The new authentication component will contain copies of the password databases of individual legacy applications.

2.3.3 Model Solution Architecture

A combined view of the component and connector view for the model solution is presented in Figure 5. The component and connector view shows a Java Client Application in the center. This application invokes the Web service operations from the different representative applications and displays results of the operations on the screen. It is instrumented to show response times from each of the legacy applications along with the results. The Web service that connects to the Java application contains two operations: one that exchanges basic data types as plain text and one that returns an image as part of the message.

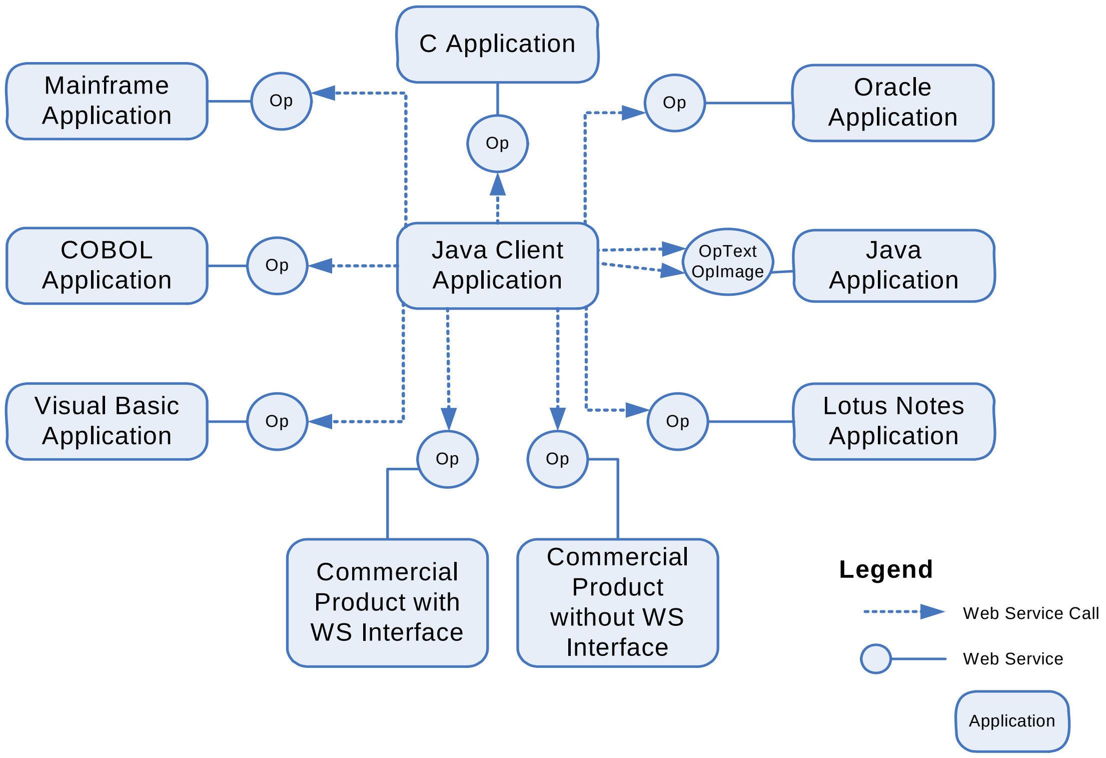

Figure 5: Component and Connector View for the Model Solution

Figure 6 shows potential deployment options for the model solution. The selection of the deployment option for each application will depend on the availability of tools and software for the legacy platform, as well as performance.

- Option 1 is to deploy in the way indicated by the C, Visual Basic, and COBOL applications, as well as the commercial products with or without a Web service interface. In this case, a machine is set up as an HTTP server. The HTTP server contains the SOAP Engine that redirects operations to the proper applications. Each application has a Web service adapter resident on the same machine with the application. This adapter is needed to make functions of the legacy application available to the Web service running in the SOAP Engine.[2] The adapter converts the call from the Web service into calls to the application and then receives and sends results back to the Web service. The advantage of this approach is the unification of all interfaces to legacy applications. The disadvantage is that performance might degrade given that the HTTP server acts as a “router” for all Web service calls. Using one central component also reduces reliability because it introduces a potential single point of failure. For the single sign-on scenario the additional access control component would be deployed as a servlet on the central HTTP server node.

[2] A nice depiction of how this works can be found at http://www.soapuser.com/basics4.html.

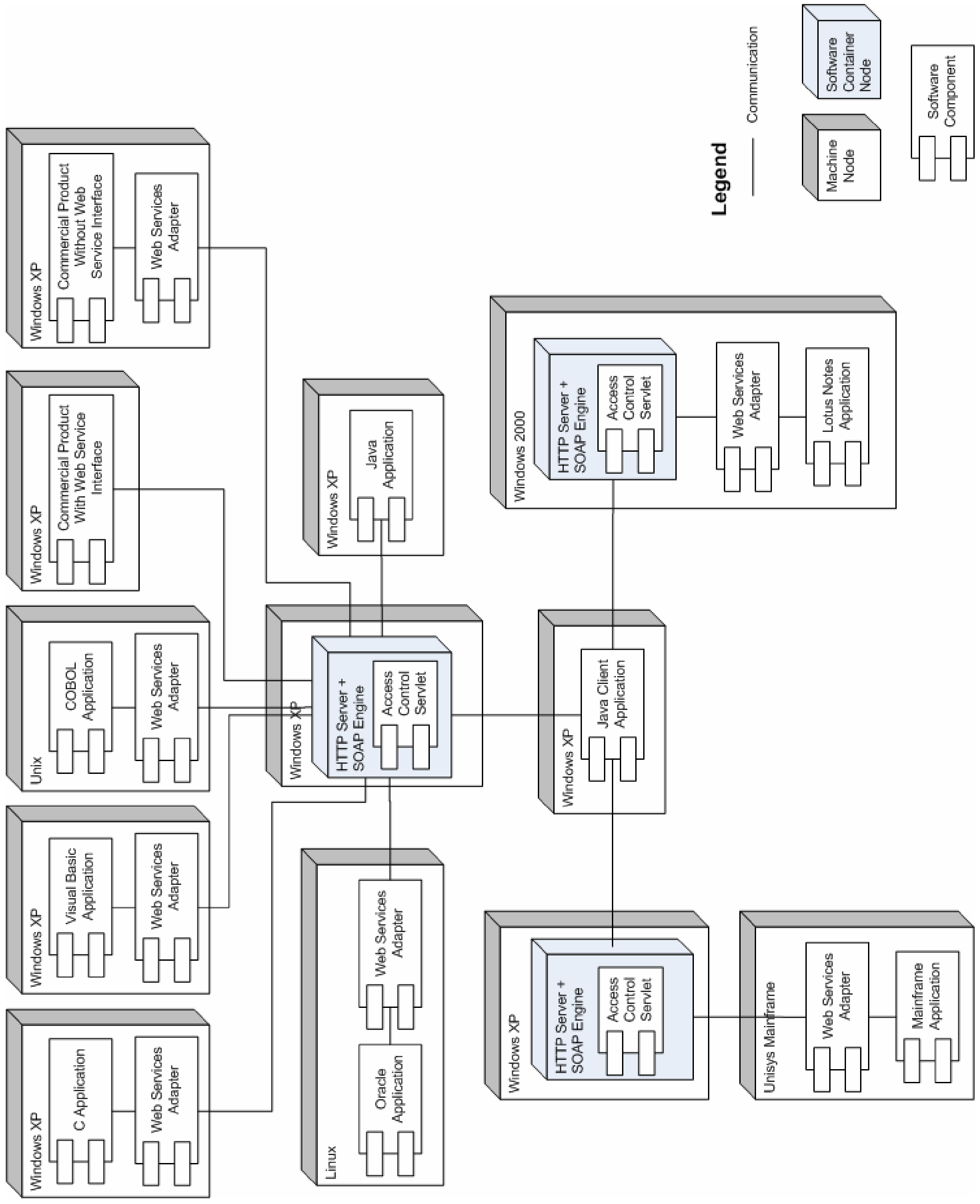

Figure 6: Deployment View for the Model Solution

A variant of this option is to have several HTTP servers, depending on application usage patterns. Figure 6 shows an additional HTTP server for the mainframe application. This would indicate that the mainframe application is accessed frequently enough via Web services to warrant its own HTTP server. An extreme variant would have one HTTP server per application. The advantage of this variant is the load sharing between HTTP servers. The disadvantage is the need for additional nodes to run the HTTP servers. In this variant, access control components run as servlets in each HTTP server.

- Option 2 is to deploy in the way indicated by the Lotus Notes application. In this case, the application resides on the same machine as the Web services adapter and the HTTP server. The selection of this option depends on the availability of both Web service libraries and tools and HTTP servers for the legacy platform. The advantage of this approach is that the load on the HTTP server is spread across its multiple instances. The disadvantage is the additional load on the legacy application host machine. With this option, an instance of the single sign-on access control component would run on all nodes that run an HTTP server.
- Option 3 is represented by the commercial product with Web service interface and the Java application. In this case there is no need for a Web services adapter because the product is already enabled to be used in a Web service context or the application is enabled to receive remote calls. The advantage is that adapters would be unnecessary. However, the caveat is that applications or products must already be enabled to be integrated in this way, and this is probably only feasible in newer products or applications.

3 Conclusions

Technology is moving at a faster pace than people can keep up with. Because of this, certain technologies or “buzzwords” become popular and are quickly embraced by practitioners and managers alike. Examples are client/server in the early 90s, e-*anything* in the late 90s, and now service-oriented architectures and Web services. The problem with embracing technologies based on their popularity is that they are often inserted into organizations—or mandated—without any formal evaluation as to their suitability to the organizations’ environments.

Context-based technology evaluation allows for better informed decisions about the fitness of a technology within a particular context. The use of model problems within the evaluation process provides an organized and efficient way to “play” with technologies before they are inserted into organizations’ systems.

As demonstrated by the case study, context-based technology evaluation requires extensive planning, effort, and research. But too frequently the alternative is embracing a technology that must later be abandoned because of mismatches between its capabilities and the expectations for its capabilities. It’s almost certain that the cost of such an unfortunate effort will exceed and justify the cost of a context-based evaluation.

Bibliography

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

[Apache 05a]	The Apache Jakarta Project. *Apache Jakarta Tomcat.* http://jakarta.apache.org/tomcat/ (2005).
[Apache 05b]	The Apache Software Foundations. *WebServices - Axis*. http://ws.apache.org/axis/ (2005).
[Comella-Dorda 04]	Comella-Dorda, S; Dean, J; Lewis, G; Morris, E; Oberndorf, P; & Harper, E. *A Process for COTS Software Product Evaluation* (CMU/SEI-2003-TR-017) Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, July 2004. http://www.sei.cmu.edu/publications/documents/03.reports /03tr017.html.
[Eclipse 05a]	Eclipse.org. http://www.eclipse.org/ (2005).
[Eclipse 05b]	*Eclipse Web Tools Platform (WTP) Project*. Web Standard Tools Subproject. http://www.eclipse.org/webtools/ (2005).
[IBM 02]	IBM Corporation. *Web Services Security (WS-Security) Specification.* April 2002. http://www-106.ibm.com /developerworks/library/ws-secure/.
[JBoss 05]	JBoss. *JBoss Application Server.* http://www.jboss.org/products/jbossas (2005).
[Lewis 04a]	Lewis, G. *An Approach to Analysis and Design for COTS-Based Systems*. Proceedings of the 4th International Conference on COTS-Based Software Systems. Bilbao, Spain, February 7-11, 2005. Lecture Notes in Computer Science, *3412*, New York, NY: Springer, February 2005.
[Lewis 04b]	Lewis, G. and Wrage, L. *Approaches to Constructive Interoperability* (CMU/SEI-2004-TR-020). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, January 2005. http://www.sei.cmu.edu/publications/documents /04.reports /04tr020.html.

[Microsoft 05a] Microsoft Corporation. *Microsoft .NET* homepage. http://www.microsoft.com/net/ (2005).

[Microsoft 05b] Microsoft Corporation. *Visual Studio: Product Information for Visual Studio .NET 2003.* http://msdn.microsoft.com/vstudio/productinfo/ (2005).

[Microsoft 05c] Microsoft Corporation. *Microsoft NTLM*. http://msdn.microsoft.com/library/default.asp?url=/library /en-us/secauthn/security/microsoft_ntlm.asp (2005).

[OASIS 04] Organization for the Advancement of Structured Information Standards. *Web Services Security v1.0 (WS-Security 2004).* March 2004. http://www.oasis-open.org/specs/index.php#wssv1.0.

[OASIS 05] Organization for the Advancement of Structured Information Standards. *Security Assertion Markup Language (SAML) v2.0*. March 2005. http://www.oasis-open.org/specs/index.php#samlv2.0.

[Oracle 05] Oracle Corporation. *Oracle Database.* http://www.oracle.com/database/ (2005).

[Sun 05] Sun Microsystems. *Java 2 Platform, Enterprise Edition (J2EE).* http://java.sun.com/j2ee/ (2005).

[W3C 00] World Wide Web Consortium. *SOAP Messages with Attachments*. http://www.w3.org/TR/SOAP-attachments (2000).

[W3C 05] World Wide Web Consortium. *SOAP Message Transmission Optimization Mechanism.* http://www.w3.org/TR/soap12-mtom (2005).

[Wallnau 01] Wallnau, Kurt; Hissam, Scott; & Seacord, Robert. *Building Systems from Commercial Components*. New York, NY: Addison-Wesley, 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 2005	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE A Process for Context-Based Technology Evaluation		5. FUNDING NUMBERS FA8721-05-C-0003
6. AUTHOR(S) Grace A. Lewis, Lutz Wrage		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2005-TN-025
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

In order to determine a fit between systems and technology, it is necessary to evaluate technologies within the contexts that they will be used. This report describes a process called context-based evaluation that determines the fitness of a technology within a specific context. It includes hands-on experimentation with the technology for a greater understanding of its implications, as well as early competence development of the people conducting the experiments. An integral part of the process is the development of model problems; these are prototypes, situated in a specific context, with the goal of satisfying evaluation criteria.

The focus of this report is on evaluation of software technologies, such as Web services, database systems, or architectural frameworks and development tools. The report also includes a case study of the use of this process for the evaluation of Web service technology.

14. SUBJECT TERMS context-based technology evaluation, model problem			15. NUMBER OF PAGES 35
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL