\*\*\*Some of the Material in this paper has been repurposed from IT251-1502B-01

 with Professor Noffsinger, and IT351-1601-01 with Professor Williams \*\*\*

Software Construction SWE400-1601B-01

Phase Three Individual Project

Damon Tholson

03-11-2016

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Author** | **Revision** | **Date** | **Sections changed** | **Description** | **Reason** |
| Damon Tholson | 1.1 | 02-24-2016 | Initial Revision | Original Document | Initial revision |
| Damon Tholson | 1.2 | 03-05-2016 | Section 3 API Evaluations  | Phase Two Modification | API Evaluation |
|  |  |  |  |  |  |
| Damon Tholson | 1.3 | 03-11-2016 | API Recommendations | Phase Three Modification | API Recommendations |

**Table of Contents**

1 Project Outline 5

1.1 Description: 5

2 Software Reengineering 5

2.1 Purpose: 5

2.2 Functionality: 5

2.3 General Architecture: 6

2.4 Reengineering Plan: 8

3 API Evaluation 9

3.1 API Designs 9

3.1.1 SCORM APIs 9

3.1.2 Tin Can APIs 10

3.1.3 The Java General APIs 11

3.1.4 Web Services APIs 12

3.1.5 Java EJB APIs 13

4 API Efficiency, Reliability, and Maintainability 14

4.1 SCORM APIs 14

4.1.1 Efficiency Strengths: 14

4.1.2 Efficiency Weaknesses: 14

4.1.3 Maintainability Strengths: 14

4.1.4 Maintainability Weaknesses: 14

4.1.5 Reliability Strengths: 14

4.1.6 Reliability Weaknesses: 14

4.2 Tin Can APIs 15

4.2.1 Efficiency Strengths: 15

4.2.2 Efficiency Weaknesses: 15

4.2.3 Maintainability Strengths: 15

4.2.4 Maintainability Weaknesses: 15

4.2.5 Reliability Strengths: 15

4.2.6 Reliability Weaknesses: 15

4.3 The Java General APIs 16

4.3.1 Efficiency Strengths: 16

4.3.2 Efficiency Weaknesses: 16

4.3.3 Maintainability Strengths: 16

4.3.4 Maintainability Weaknesses: 16

4.3.5 Reliability Strengths: 16

4.3.6 Reliability Weaknesses: 16

4.4 Web Services APIs 17

4.4.1 Efficiency Strengths: 17

4.4.2 Efficiency Weaknesses: 17

4.4.3 Maintainability Strengths: 17

4.4.4 Maintainability Weaknesses: 17

4.4.5 Reliability Strengths: 17

4.4.6 Reliability Weaknesses: 17

4.5 Java EJB APIs 18

4.5.1 Efficiency Strengths: 18

4.5.2 Efficiency Weaknesses: 18

4.5.3 Maintainability Strengths: 18

4.5.4 Maintainability Weaknesses: 18

4.5.5 Reliability Strengths: 18

4.5.6 Reliability Weaknesses: 18

5 Recommendations for Improvement 19

5.1 SCORM API 19

5.1.1 Efficiency Improvements: 19

5.1.2 Maintainability Improvements: 19

5.1.3 Reliability Improvements: 19

5.2 Tin Can API 19

5.2.1 Efficiency Improvements: 19

5.2.2 Maintainability Improvements: 19

5.2.3 Reliability Improvements: 19

5.3 The Java General APIs 20

5.3.1 Efficiency Improvements: 20

5.3.2 Maintainability Improvements: 20

5.3.3 Reliability Improvements: 20

5.4 Web Services APIs 20

5.4.1 Efficiency Improvements: 20

5.4.2 Maintainability Improvements: 20

5.5 Java EJB API 20

5.5.1 Efficiency Improvements: 20

5.5.2 Maintainability Improvements: 20

5.5.3 Reliability Improvements: 20

6 Configuration and Change Management 21

7 Software Construction Tools and Technique 22

8 References: 23

# Project Outline

## Description:

 CourseMill is a Learning Management System (LMS) which allows a corporation to manage and track the results of online training given and received by the company. The CourseMill application utilizes utilises the JavaScript, java, SQL, and html programming languages. It also provides LDAP (Lightweight Directory Access Protocol) connections which integrate with the customer’s configuration to achieve SSO (Single Sign On) capability when generating user accounts.

# Software Reengineering

## Purpose:

The primary purpose of the CourseMill application is to manage, track, and display in a usable format the results and content of online training solutions. To produce online training content housed in CourseMill the desktop authoring tool Lectors Inspire is used. This tool provides a graphical environment which allows developers to create mobile courses once and then store the content in the CourseMill application allowing to be accessed and used by all students and teachers who are authorized.

## Functionality:

The primary features that the application provides are as follows:

1. Advanced Reporting. Information regarding a student’s progress, classroom locations, courses offered, and other pertinent information can be easily created using advanced reporting. Advanced reporting provides drag and drop functionality to allow users to create visual reports with charts and graphs. No SQL programming experience is required as you are able to use a graphical interface to develop SQL queries and commands.
2. Compliance Management. When a student is registered in the CourseMill application they are assigned a curriculum, which contains courses they are required to complete and the sessions which the student can be enrolled in to complete the course. Deadlines for completing the completing the course along with the completion status of the student are included in the course table. Compliance training is also available for instructors which ensures they are properly qualified to teach the courses to which they are assigned.
3. Graphical User Interface. A graphical user interface is provided to allow all users to navigate through the application. Every user is uniquely identified and can only view content for which they are authorized.

## General Architecture:

An online application server is used to house the database and data is exchanged through client server requests. Figure 1 displays the basic Client/Server communications scheme.

Figure 1 Client Server Communications



The application houses data through the use of an SQL database whose tables, columns, and rows can be modified and viewed through SQL queries and commands. The Java API’s used are java server pages which are created through a Java web application. Within the .jsp files the javascript programming language is used to create Database Scripts whose primary purpose is displaying data retrieved from the database in a usable format. Figure 2 is an example of the tables and columns contained within the SQL database.

Figure 2 Interact table and columns

|  |
| --- |
| **interact** |
| CourseID VARCHAR(255) |
| SessionID INT(11) |
| ScoID VARCHAR(255) |
| StudentID VARCHAR(255) |
| INTER\_ID VARCHAR(255) |
| INT\_TIME VARCHAR(13) |
| STUDENT\_RESPONSE VARCHAR(255) |
| RESULT VARCHAR(16) |
| LATENCY VARCHAR(13) |
| RECORD\_TIME DATETIME |
| **Indexes** |
| stud\_crs\_sess\_idx cr\_sco\_int\_rs\_idx |

## Reengineering Plan:

Software re-engineering involves the use of specialized tools and procedures which allow us to break down a program into its original binary code and view the inner details. The reengineering plan will consist of the following phases:

1. Implementation Recovery - This stage involves determining the purpose of the software. By looking at the behavior of a program and how it is built you can predict its rules. As the observation becomes more in-depth and involved the prediction becomes more accurate. The steps involved in in the implementation recovery stage are as follows:
* Browse the existing documentation. This step allows you to learn the original design of the program and its intended purpose (Blaha, 2014)
* Create a modeling tool. This step allows you to determine the development style and the database structure.
1. Design Recovery – During this stage you break down the mechanics of the program. This allows you to break down the source code into a binary format.
2. Analysis Recovery – This stage involves refining the existing model based upon the

observations you have made. The steps involved in analysis recovery are as follows:

* Clarify the design. This step involves removing any design artifacts.
* Eliminate redundancy. This step involves removing duplicate sets of data.
1. Iteration – As is common throughout all software engineering endeavors the design will not be perfect and you will need to backtrack to correct mistakes and oversights

(blaha, 2014).

Some of the benefits of software re-engineering include enhancing features, and fixing bugs, increasing the robustness and security of the design, and helping software testers study viruses and other malicious code. The objectives of the reengineering plan for the CourseMill application will consist of the following:

1. Refactoring the design to provide increases transparency to system administrators. CourseMill stores javascript pages on the application server however currently those scripts cannot be created or viewed by system administrators.
2. Updating the JDBC (Java Database Connection) interface. The current interface produces SQLExceptions on valid SQL queries.
3. Increasing the robustness and security of the design by identifying possible malware and viruses (Richa, 2014).

# API Evaluation

## API Designs

In the following section the common API’s used in the CourseMill application will be reviewed. In the evaluation the completeness, suitability, and overall design quality will be critiqued. Additionally, diagrams will be created which show illustrate the designated API’s design and identify the entry points into the application. And finally there will be an assessment of the API documentation to include the overall clarity, completeness, and ease of use.

### SCORM APIs

The SCORM (Sharable Content Object Reference Model) API provides a set of technical standards for online learning software. These standards guide programmers to write code that plays well with other learning software. These standards set by SCORM also govern the way the LMS (the CourseMill application) and the software learning products communicate. This allows the CourseMill database to accept a wide variety of software learning products which can be uploaded to the database and then used by others within the CourseMill Application.

The following figure 3 SCORM API Interface class diagram shows the integration architecture which identifies the entry points into the application. The first class is the integration layer which is the interface between the SCORM engine and the CourseMill application. This layer facilitates communications and acts as a buffer to keep the systems separate. The integration layer is the only part of the SCORM API which customizable for the specified LMS. In this class all of the methods are defined but not implemented. The DefaultIntegration class in the middle provides all of the default methods which are standard for all implementations. Finally, you have the ClientIntegration classes which allow you to customer design methods for each client though usually only the default methods are used.

Figure 3 SCORM API Interface



The SCORM API documentation reflects the fact that SCORM isn’t truly a set of standards that were written from the ground up. Instead, the API referenced the industry standards that were already in place and solved part of the problem and then added to the standards to create a complete solution. This complete solution essentially told developers the proper way to use the standards that were already in place. SCORM is widely used and very well documented, however it does have a shortage of features which has caused CourseMill to migrate to a more modern API known as the Tin Can API.

### Tin Can APIs

While SCORM is still the most widely used online learning standard, the Tin Can API Which is also known at the Experience or xAPI is quickly replacing it due to the enhanced features and simplified architecture. All of the SCORM content can still be used in a xAPI environment through the SCORM Engine interface architecture which is shown in figure 3. Essentially the xAPI is an evolved SCORM API which is backwards compatible and retains all of the features of its predecessor while adding additional features to include application security, mobile application features, and the ability to transition from a stationary to a mobile platform.

The documentation for the Tin Can API is complete, thorough, and easy to understand. It provides increased encapsulation thereby increasing the simplicity of the standard while increasing the functionality at the same time. As the internet grows from specified desktops and laptops to mobile devices and finally to most electronics (i.e the
IOT “Internet of Things”) cross platform becomes vital to a modern API and it is that feature which makes it suitable for the CourseMill Application.

### The Java General APIs

The java API is a general API which provides a full set of interfaces that utilize the Java programming language. In Java the utilization of Interface objects provides a set of common methods that can be called upon by unrelated classes to perform an action with the intricacies of that action being defined by the class that called upon the interface (Jensen, M. 2009). Programming to an interface allows a developer to utilize the object oriented principle of Abstraction which is when only the essential pieces of logic needed to accomplish an action are analyzed and all else is ignored. Achievement of abstraction occurs when the object oriented principle of Encapsulation is followed. Encapsulation is when an interface object represents, and is only tasked with representing, what an object can do while the details of how it does it are ignored. This compartmentalization of logic greatly simplifies programming methods and allows for an easier understanding of complex programs.

 The Figure 4 class diagram shows a person wishing to enroll in Coursemill who “is” either a student or professor and “has” an address. In java this is known as the inheritance hierarchy which is a concept in object oriented programming where a new class is created by acquiring a parent class’s existing capabilities. Documentation regarding the Java interfaces can be found at oracle.com with Java arguably being the most well documented API in existence. The Java API is well suited for any application and is the most one of the most widely used programming languages.

Figure 4 Java Inheritance Hierarchy Diagram



### Web Services APIs

In addition to general APIs there are also web APIs which are known as web services. Web Services are a form of communication which allows us to share data between different programs on separate computers which may reside on different platforms. They are composed of multiple pieces of customer developed code and allow two or more web applications to communicate with each other. The advantage of web services is that they allow organizations to reuse code built by others which in turn allows them to be used by IT Systems as services provided over the internet.

To allow these different web services to communicate a set of standards which spells out the interaction between these services has been developed. The most common of these standards is known as SOAP (Simple Object Access Protocol). The SOAP API is a standard for making XML requests online, which defines what the request and response in XML looks like and how to share those messages back and forth. The activity diagram shown below in figure 6 illustrates the activities of a SOAP API.

Figure 5 SOAP API Activity Diagram



### 3.1.5 Java EJB APIs

EJB Enterprise JavaBeans which is also known as enterprise bean allows for the accounting of data relationships, error handling, transactions, security, scalability, and connectivity. Common problems can be resolved with common solutions through the use of a framework, and JBE was designed to be that framework. It handles the component deployment to let you focus on writing applications. Each EJB can act on its own it contains an interface, performs services for clients, and tracks its own data. The following figure 6 diagram shows the interaction between the EJB API on the client and server side applications.

Figure 6 Java EJB Diagram



The Advantages of EJBs are that they are a common model for building code, they solve many problems in a common manner, and they provide an easy solution for managing components. The disadvantages are the requirement of an expensive application server, often more than is needed for the application, and there are many more flexible alternatives. An expensive application server is already required for the CourseMill application and the flexibility is more than compensated for by the ease of configuration and thorough documentation.

# API Efficiency, Reliability, and Maintainability

The three attributes that define an API are efficiency, reliability, and maintainability. Efficient APIs allow developers to produce complicated code in less time. Reliable APIs ensure a steady exchange of information between applications. And maintainable APIs allow developers in the maintenance phase of a project to continue to improve the application through updates and patches while maintaining compatibility with other programs. There are factors which both hinder and improve these attributes.

## SCORM APIs

### Efficiency Strengths:

* Once content is published it can play on multiple platforms.
* Well defined packaging model allowing courses to be uploaded using a simple zip file.

### Efficiency Weaknesses:

* Use of JavaScript is a limiting communication technique.
* Dynamic content is not permitted requiring all files used by the LMS to be created ahead of time.

### Maintainability Strengths:

* Publicly defined and direct vocabulary.

### Maintainability Weaknesses:

* SCORM API adapter implements complex rules for reporting errors and tracking data this requires the availability of java technology on the host machine.
* Content must be stored within an LMS.

### Reliability Strengths:

* Well Documented concepts with thorough examples
* Use of the JavaScript architecture means no reliance on plug-ins, Active X controls, or applets which means very few environmental support issues

### Reliability Weaknesses:

* Requires constant internet communications.

## Tin Can APIs

### Efficiency Strengths:

* Provides transparent debugging.
* Expanded functionality through the use of web services.

### Efficiency Weaknesses:

* Not all LMS’s have adopted the specification. Dynamic content is not permitted requiring all files used by the LMS to be created ahead of time.
* There is a learning curve associated with the new standard.

### Maintainability Strengths:

Content can be stored and accessed within the cloud and does not need to reside within the LMS database.

### Maintainability Weaknesses:

* New features increase the complexity and while increased encapsulation helps developers to integrate the new features, it can be a hindrance during troubleshooting.

### Reliability Strengths:

* Created in a software cooperative environment allowing for testing in more environments against more content.
* Supports session less communications.

###  Reliability Weaknesses:

* LMS’s will need to build new reporting capabilities to take advantage of the new features. The compatibility of those capabilities will have to be developed and refined.

## The Java General APIs

### Efficiency Strengths:

* Java programs can be Written once and ran on any machine with the use of the Java Virtual Machine.
* Java can be interpreted at run time by the virtual machine providing flexibility however it can also be compiled at runtime to run directly on the native machine. When compiled directly Java is one of the fastest language platforms.

### Efficiency Weaknesses:

* Java required a designated port.
* Java is noun oriented which causes the language to be more restrictive than others.

### Maintainability Strengths:

* Most commonly used programming language with the most extensive documentation meaning most issues have already been encountered, resolved, and the solutions documented.
* Object oriented design facilitates the creation of reusable code.
* JIT (just in time) compiler and JVM (java virtual machine) interpreter compile and interpret the code alleviating the possibility of version mismatches.

### Maintainability Weaknesses:

* Object oriented methodology has a steep learning curve. Often times code is functional and efficient but lacks clarity.
* Java does not have the ability to automatically self-update meaning Java updates must be applied manually.

### Reliability Strengths:

* Platform independent due to the Java Virtual Machine interpreting the java methods in the runtime environment.

###  Reliability Weaknesses:

* Java hides the relationship between the hardware and the code. Because of this it is difficult to determine the relationship between the code and what the hardware actually does.

## Web Services APIs

### Efficiency Strengths:

* allows organizations to reuse code built by others which in turn allows them to be used by IT Systems as services provided over the internet.
* Provides the use of a common framework.

### Efficiency Weaknesses:

* The use of the XML wrapper causes an increased strain on bandwidth.

### Maintainability Strengths:

* Uses standard HTML forms which are less complex than XML.
* Can be used with any programming model.

### Maintainability Weaknesses:

* High degree of verbosity within the syntax which results in a specification that inherently lacks clarity.
* Relies on XML which is being replaced by URL (Universal Resource Locator) calls.

### Reliability Strengths:

* Easy to operate and use and allows the use of legacy standards.

### Reliability Weaknesses:

* None found

## Java EJB APIs

### Efficiency Strengths:

* allows for the accounting of data relationships, error handling, transactions, security, scalability, and connectivity.
* Handles the component deployment allowing the developer to focus on writing applications.

### Efficiency Weaknesses:

* Lacks Descriptors forcing the developer to specify SQL information.

### Maintainability Strengths:

* Provides a framework where common problems can be resolved with common solutions.

### Maintainability Weaknesses:

* Lacks backwards compatibility with previous versions

### Reliability Strengths:

* Provide a common model for building code which results in robust applications.

### Reliability Weaknesses:

* Methods display inconsistent behaviour. The create method for does not return an error when trying to create an entity with a primary key that already exists (Suleiman, 2004).

# Recommendations for Improvement

## SCORM API

### Efficiency Improvements:

* Updating to the newer Tin Can API will allow for the use of web services over JavaScript though JavaScript can still be used when desired.
* Dynamic content will be permitted with the implementation of the Tin Can API.

### Maintainability Improvements:

* The SCORM API adapter implements complex rules for reporting errors and tracking data this requires the availability of java technology on the host machine.
* Updating to the newer Tin Can API will allow users to access developed content wherever it is stored.

### Reliability Improvements:

* Updating to the Tin Can API will allow session less communication.

## Tin Can API

### Efficiency Improvements:

* The learning curve for adopting a new standard can be mitigated either through a focused effort on training developers or the hiring or developers who already have a familiarity with the standard.

### Maintainability Improvements:

* Training which provides a comparison between the SCORM and Tin Can API will allow developers to identify areas where the new standards are likely to cause issues. This will provide a good starting point for troubleshooting.

###  Reliability Improvements:

* The gradual implementation of the new standard along verified backups which are updated daily will allow developers to more easily recover if an integration issue occurs.

## The Java General APIs

### Efficiency Improvements:

* Java continually improves in efficiency with every release. To maintain maximum efficiency, use the latest versions of the JVM and the JIT (Just in Time) compiler.

### Maintainability Improvements:

* An organization must make a commitment to either hire experienced programmers or provide ongoing training for new hires.
* Because Java does not provide automated updates a documented procedure should be developed which outlines the necessary steps to update the version of java.

### Reliability Improvements:

* comments which describe the overall purpose of the class, the individual purpose of the methods, and how the class is integrated with other classes should be included in every class.

## Web Services APIs

### Efficiency Improvements:

* If bandwidth is a concern, then use of the REST API which wraps data in bare XML elements should be considered.

### Maintainability Improvements:

The REST API is simpler, less verbose and relies on URL calls.

## Java EJB API

### Efficiency Improvements:

* The development of descriptors will alleviate the burden of developing specific SQL information in the source file.

### Maintainability Improvements:

* Adding the feature of backwards compatibility for future releases should be developed.

### Reliability Improvements:

* Method behaviour should be standardized and tested in future releases.

# Configuration and Change Management

## Change Management Process

The requirements of every project will change over time and flexibility is needed to allow room for requirements enhancements and corrections. The goal of the Agile methodology is to manage change rather than prevent it because implementing change coincides with improving the product. To provide this change we will use a framework which addresses the different factors responsible for software project failures. In the following diagram we will show how changes will be introduced based upon their categorization as defective repair or enhancements to a functioning system. All defect repair requests will go through an approval process which determines the cost and time of repair along with the impact to the system. If the impact is of minimal value or not visible to the end user it will likely be listed very low on the priority list



## Configuration Change requirements tools

Verification of the software will be achieved through functional testing by assigned test engineer. Functional testing utilizes the provided input by the user to receive the output of the system. The expected output of the system is then compared to the actual output to determine the quality of the output. Performance testing will ensure the provided hardware is able to handle the assigned workload and not have a catastrophic failure. It will also bring to light issues that would not be detected under normal circumstances ensuring an unacceptable behavior or loss of network service does not occur during operating conditions. Some of the verification and validation techniques employed include simple checks, prototyping, functional test design, and model based verification and validation.

Requirements verification includes checks for the following conditions. If any of these requirements are met a requirement change is needed:

* The product is built right.
* Each step of the process yields the correct product.
* The software requirements are consistent with the specification.

Requirements validation checks for the following conditions:

* The right product is being built.
* The software will satisfy the customer needs.
* The software requirements are consistent with the goals and requirements of the project.

6.3

# Software Construction Tools and Technique

References:

Blaha, Michael (October 19, 2001) Reverse Engineering for the Product Assessment Retrieved on February 20, 2016 from <http://www.informit.com/articles/article.aspx?p=23692&seqNum=6>

Deitel, P., & Deitel, H. (2015). Object-Oriented Programming: Inheritance. In *Java How to*

*Program (Early Objects)* (10th ed., p. 361). Upper Saddle River: Pearson Custom Publishing.

“Jensen, M” (2009). *Program to an interface, not an implementation*. Retrieved from

 http://www.fatagnus.com/program-to-an-interface-not-an-implementation/

Richa (2014). Reverse engineering tutorial: how to reverse engineer any software. Retrieved from: <https://blog.udemy.com/reverse-engineering-tutorial/>

Ostyn, Claude (2003). *A bried introduction to SCORM.* Retrieved from

 <http://scorm.com/wp-content/assets/cookbook/SCORM%201_2%20Overview.htm>

SCORM, E (2014). SCORM Engine Integration Architecture. Retrieved from

 <http://rustici-docs.s3.amazonaws.com/engine/2014.1.x/6-integrationarchitecture.html>

Suleiman, H (2004). A Critique of EJB 3. Retrieved from

 <http://www.theserverside.com/news/1377073/A-Critique-of-EJB-3>

For myself the most compelling part of this was defining the different APIs and differentiating them between general ones and ones with a specific purpose. I fell behind at several points including now and just couldn’t catch up. I still struggle on judging an API because I feel the points of judgement being efficiency, maintainability, and reliability are too broad an area t judge for anyone except for a subject matter expert on that particular API. Also the short length of the class made it difficult for the instructor to provide one on one help which would have helped greatly(technical-vocational-schools.com).

Expanding on that point I feel that this class would have been much better suited to selecting 2 or 3 APIs and examining them in detail. In asking the opinion of the class they were asking people who are not experts to give an expert opinion. For each quality of any of the APIs we looked at there were strength and weaknesses and unless you have years of experience using the API in different situation then the opinion you create is very limited.

I did learn a lot from the discussion boards because the level of experience between the different students is very high and I believe some of the students were developers with years of experience. Even those who were new provide a different perspective that I would have never see if the Discussion board wasn’t there. The most important thing I learned was the savings re-engineering provided over starting a project with a blank slate, you really save a lot of time and resources when you learn from the mistakes of others (Robins, 2010)

References

Robins, D. (2010). *Software Re-Engineering Best Practices.*Retrieved from: <http://www.meetup.com/The-San-Francisco-NET-User-Group/events/12568687/>

Technical-vocational-schools.com. (n.d.). Technical and vocational school guide. Retrieved March 15, 2016, from www.technical-and-vocational-schools.com: <http://www.technical-vocational-schools.com/online_classroom_learning.aspx>

**ey Assignment Draft**

Changes occur rapidly in most software development projects, and it is vital to keep track of these changes and the associated information for correct configuration of the systems and software that are necessary to run the system correctly. These concepts are covered in the topics known as *configuration* and *change management* and will be the focus for this assignment.

For this assignment, you will be examining and documenting the change management process for the project and the requirements for configuration and change management. You will also provide recommendations for appropriate configuration and change management tools.

The project deliverables are as follows:

* Update the Software Reengineering Project document title page with the new date.
* Update the previously completed sections based on instructor feedback.
* Configuration and Change Management
	+ Change management process
		- Examine and document the change management process for the project. Flowcharts or other visual documentation should be included.
		- Establish a list of requirements for configuration and change management for the project.
	+ Configuration and change management tools
		- Using the information and requirements that were developed for the first part of this assignment, provide recommendations for configuration and change management tools for the project.
* Be sure to update your Table of Contents before submission.
* Name the document "yourname\_SWE400\_IP4.doc."
* Submit the document for grading.

**Please submit your assignment.**

**For assistance with your assignment, please use your text, Web resources, and all course materials.**

Weekly tasks or assignments (Individual or Group Projects) will be due by Monday, and late submissions will be assigned a late penalty in accordance with the late penalty policy found in the syllabus. NOTE: All submission posting times are based on midnight Central Time.

**Key Assignment**

Significant work has been completed so far in the project. The parts of the Software Reengineering Project that have been completed include the following:

* Software Reengineering (Week 1)
* API Evaluation (Week 2)
* API Efficiency, Reliability, and Maintainability (Week 3)
* Configuration and Change Management (Week 4)

The final step in completing the Software Reengineering Project is to make the changes to the application planned in other phases of the project. The changes will be made in an effort to improve the application programming interface (API) and strengthen the overall application.

For this assignment, you will use software construction techniques and tools to modify your project application and implement the changes that were planned in Week 3 of the project. You will also submit a description of the changes that were made and the techniques and tools that were used to make the changes. Finally, you will further refine the Software Reengineering Project document to produce the final version. Updates may be based on peer and instructor feedback.

The project deliverables are as follows:

* Update the Software Reengineering Project document title page with the new date.
* Update the previously completed sections based on instructor feedback.
* Software Construction Tools and Techniques
	+ Use at least 1 software construction technique and associated tool to perform the changes to the APIs in the application recommended in the Week 3 part of the project.
	+ Submit the modified application along with a new section of the project document describing the changes that were made and the techniques and tools that were used to make the changes.
	+ This is the last step of the key assignment. It should bring together all aspects of the project in 1 cohesive paper and software solution.
	+ Revise the previous sections so the entire document flows and has a strong introduction and conclusion.
	+ The application files will be submitted as a single ZIP file, including all source code and executable files.
* Software Reengineering Project
	+ Review the entire document for any changes and improvements that you would like to make.
	+ All aspects of the document should present a cohesive report, and each weekly topic should coincide with the other weeks.
	+ Ensure that this final version of the plan is sufficiently detailed to allow the reader to confidently move forward with the same changes to the application that you have recommended.
	+ Any previous instructor feedback should be addressed with appropriate changes.
	+ Be sure to update your Table of Contents before submission.
	+ Name the document "yourname\_SWE400\_IP5.doc."
* Zip the project document and the application files together in a file named "yourname\_SWE400.ZIP," and submit the .zip file for grading.

**Please submit your assignment.**