Web Mashups in the Supply Chain

Matthew J. Zeets

Submitted to the graduate degree program in Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Thesis Committee:

_____
Dr. Victor Frost: Chairperson

_____
Dr. Gary Minden

_____
Dr. Daniel Deavours

_____
Date Defended

The Thesis Committee for Matthew J. Zeets certifies
That this is the approved version of the following thesis:

Web Mashups in the Supply Chain

Thesis Committee:

_____
Dr. Victor Frost: Chairperson

_____
Dr. Gary Minden

_____
Dr. Daniel Deavours

_____
Date Approved

Acknowledgements

I would like to thank Dr. Victor Frost for everything he has done to help me along in attaining this milestone in my life. He has always known what to do to keep me motivated and on the right track throughout the entire process. I truly would not have been able to be where I am at now without his guidance along the way. I would also like to thank Dr. Gary Minden and Dr. Daniel Deavours for taking the time to be on my committee. They are very intelligent, motivated individuals whose input means a lot to me. Next I would like to thank Dr. Man Kong for being there all the way through my undergraduate and graduate career at KU, always giving me great advice in school and in life. Lastly I would like to thank my family- Joe, Deanne, Danielle, Michael, Janene, James, Kelly, Connor, Ed, and Shirley, who believed in me even when I did not believe in myself and have always given me the strength to be the best I can be.

Abstract

Many supply chains include multiple parties with the goods exchanged several times from the time of their production until they are sold. The liability issues that arise from goods changing hands several times along the way, as well as the exporting and importing regulations that have to be considered, add considerable complexity to the system. Furthermore weather, traffic, and market fluctuations make supply chains less reliable. Many of these issues in supply chain management could be solved or their adverse effects lessened by having information more readily shared among parties.

Using Web 2.0 technologies such as a Service-Oriented Architecture, web mashups, blogs, wikis, and social networking sites could be used to facilitate sharing information between parties in a supply chain. This paper focuses on analyzing the potential use of web mashups by enterprises in the supply chain industry. Web mashups, the supply chain, and the security implications of using web mashups in the supply chain are analyzed to determine if it would be worthwhile to use web mashups in the supply chain industry.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the cost involved in transporting goods all over the world, enterprises are always looking for ways to improve supply chain management. Several parties [forming a supply chain] have to work together to get goods from where they are manufactured to where they are sold. Having many parties working together leads to inefficiencies. These inefficiencies could be solved if the parties involved and the responsibilities of each party stayed the same. However market fluctuations cause supply chains to change often. Furthermore, there are security issues in supply chains including theft of goods and smuggling illegal goods, which lead to losses of money and reliability. Making information more readily available to appropriate parties involved in a supply chain can help decrease inefficiencies and increase security.

There are a number of Web 2.0 techniques to help make information easier to access between parties including Service-Oriented Architecture (SOA) [12], web mashups, blogs, wikis, and social-networking sites. Web 2.0 techniques make

information easier to share between parties by allowing users to contribute to a web service in addition to using information provided by it. Some Web 2.0 tools like blogs, wikis, and social-networking sites are simple in how they make information available to other parties. Blogs post information on a regular basis and provide feeds that allow the information to be aggregated and filtered easily. Wikis provide a central source for many parties to post important information and to find information others have posted. Social-networking sites allow for easy communication between parties in the same field and as the name suggests, make it easier to network with new parties. Service-Oriented Architecture (SOA) and web mashups take more work to create and to get working but offer ways to alleviate issues with blogs, wikis, and social-networking sites. Both SOA and mashups combine two or more existing services together in a way to make a new service. There are differences and similarities between how SOA and mashups go about doing this which will be discussed more in the body of this paper. SOA has already been used by an initiative called Electronic-Freight Management [2] to test in the supply chain industry, so SOA will be analyzed and compared with mashups to help determine the worth of mashups in the supply chain industry.

This thesis will focus on analyzing the use of mashups in the supply chain industry and deciding if they can benefit the participating stakeholders. This thesis will determine the suitability of mashups used in the supply chain by answering a number of questions that are necessary to know if mashups would be worth the effort of implementing in the supply chain. These questions are still set in a theoretical setting, so the answers will not likely be proven by deductive logic. However, the

questions will be answered with evidence showing that either there is a scenario in which mashups would not work in the supply chain or presents examples of how mashups would likely be used that is not disproven by any of the criteria being looked at. Questions addressed here to decide mashups' worth to stakeholders are:

- Will mashups work in an enterprise setting?
- What advantages does using mashups in the supply chain offer?
- What are the main costs of implementing mashups in a supply chain?
- Can mashups be implemented in a supply chain in a secure way?
- Will enterprises share information openly?

The contribution of this paper will be to show evidence that mashups would either work or fail in the supply chain industry. By answering the questions set out above, the opportunities presented by mashups and the largest reasons mashups would not work will be discussed. If there is evidence showing scenarios that offer opportunities to the supply chain industry that are not available without mashups and there is no evidence supporting the problems that could occur with mashups, then it is sufficient to say there is enough reason to attempt to use mashups in the supply chain industry. If there is evidence that no scenarios would offer opportunity or if there is evidence supporting the problems with mashups being too difficult to overcome, then it is sufficient to say there is enough reason to avoid using mashups in the supply chain industry.

The remainder of this paper will be broken up into four chapters: the next chapter is devoted to providing the background information for mashups, a chapter devoted to

reviewing supply chains and discussion of the problems they will present for mashups, a security chapter presents the trust concerns mashups introduce and the ways to handle those issues, and an analysis chapter that ties all information gathered together and determine if mashups are worth pursuing by enterprises in the supply chain industry.

Chapter 2

Mashups

Mashups, or more specifically web mashups, have grown much in popularity in recent years. This is because mashups are tools to help users make new services that have not previously been possible. More specifically a mashup is combining two services or data sources (almost always web data sources) into a new service. Usually a data source comes through an Application Programming Interface, or API, which is an interface that tells the user how data is presented and can be obtained from the application. If all current web services and data sources are available to be "mashed" together, then it becomes apparent just how many new services are possible and what a powerful tool mashups can be.

2.1 Mashup Example

An example will help make the idea of mashups less abstract. The most common example of a mashup uses Google Maps with some other data source to form a way to view data in terms of location. More concretely, you can imagine how useful it would be to take a site that provides times and locations of sporting events then

overlays this information onto a Google Map. While just providing the location and time of the event is helpful on its own, adding the map provides additional value by leveraging a new data source. Furthermore if the data provided by both sources is in an easy-to-use format, with new mashup tools emerging, it is relatively easy to "mash" these two services together. In figure 2.1, below, Google Maps would be Service A and the site that provides sporting event information would be service B. Service C would be the mashup created from combining these two sources.

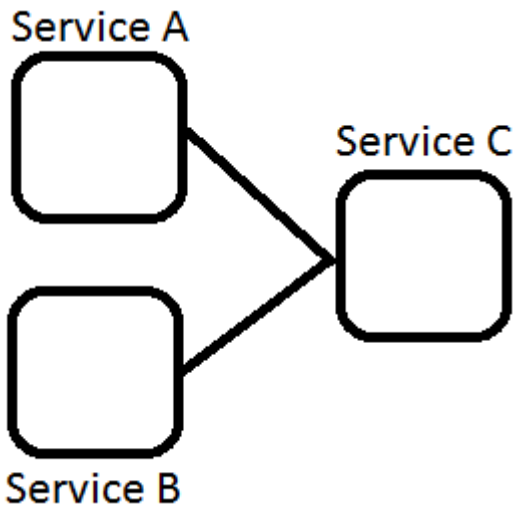

Figure 2.1- Simple Mashup Diagram

2.2 Mashup Data Sources

So far the data sources discussed in creating mashups have just been referred to as services or data sources. The term data sources can be broken down more to understand what sources mashups are made from. Data sources used in mashups can actually be split into three fields: feeds, widgets, and services.

2.3 Feeds, Widgets, and Services

Feeds are XML data streams that are usually in standard formats like RSS [15] and Atom [16]. These are a very simple way of transferring the data stream itself. Widgets are data feeds that usually encapsulate data within something else like a user interface. Where a feed is just useful for transferring data, a widget can be useful as its own stand alone tool and being a data source is more of a secondary use of it. A service is a specialized application or information provided through a web source that fills a certain need. Services many times have data that is provided or created and these are data sources that can be used by mashups. Many mashups are also services and can be used as a data source for another mashup.

2.4 New Information Streams

The idea of creating new information streams needs to be discussed to show why mashups are such an important resource for enterprises. In this paper creating new information streams will be defined as the useful knowledge an entity discovers by sharing data from different sources. An example of creating a new information stream is the knowledge discovered by combining inventory levels from two different branches in the same enterprise. While these branches might normally operate independently, the inventory levels could be shared to help the branches ship inventory to the branch in need. This would allow the branches to keep their work in progress minimized while still being able to keep up with customer demands. Work in progress is an important idea to companies that want to minimize money tied up in inventory; this idea is feasible with the right mashup environment. The information

stream in this scenario allows one branch to know it can help another branch out by shipping it some of its spare inventory. This information would not have been acquired without the knowledge discovered by combining the data sources.

2.5 Types of Mashups

2.5.1 Consumer versus Enterprise Mashups

There is a distinction between mashups that are used by consumers, as is with the Google Maps used before, and those used in the business world. These mashups are referred to as consumer mashups and enterprise mashups respectively, and the distinction is important. This paper will focus on the problems that have arisen with enterprise mashups since they will be the same ones that are used when applying mashups to the field of supply chain management, so it is important to distinguish the advantages and disadvantages that come with enterprise mashups and focus less on the consumer mashups.

2.5.1.1 Consumer Mashups

Consumer mashups are usually the simpler of the two because they mostly focus on a specific want or need that requires bringing two data sources together. Because of this, a mashup with a very specific purpose is created. Consumer mashups are the side of the mashup split that has grown very quickly, leaving enterprise mashups still mostly in the development phase. www.ProgrammableWeb.com is a useful site to find data sources and consumer mashups. In September of 2009 there were more than 1450 APIs and 4300 mashups available on the site. This is an extensive tool for

the public to find a mashup that performs the desired service or to find the APIs necessary to create a new mashup if the service has not yet been created.

2.5.1.2 Enterprise Mashups

With enterprise mashups, the idea is to utilize a more complex system of internal and sometimes external data sources. These data sources can all be mashed together so that sharing of information is easy and helps every part of the enterprise run as efficiently as possible. While having the 1450 APIs available on www.ProgrammableWeb.com leaves the "masher" with many options, there are too many APIs that are not relevant to an enterprise's needs.

Ideally, an enterprise has a common platform that stores pertinent data sources and provides users with an easy to use interface to mash them together, so that anyone in the company can help with creating new information streams and keep it from getting bottlenecked in the IT department. There is much information to be gained by mashing data sources together, but IT departments do not have the resources to do it all on their own [7]. A well set up enterprise mashup platform, whether it be internally developed or an outside source like IBM Mashup Center [3], is a very important aspect to making enterprise mashups work. IBM Mashup Center and the differences between consumer and enterprise mashups will be discussed in more depth later.

2.5.2 Presentation Layer, Data, and Process Mashups

Defining different types of mashups can be approached in many different ways. While splitting them up into consumer and enterprise mashups characterizes them by

need or reason for existence, there needs to be a way to accurately describe the broad functionality of each mashup. These broad functionalities can be divided into three types: presentation layer, data, and process mashups [9].

2.5.2.1 Presentation Layer Mashups

Presentation layer mashups are simple mashups that present data from different sources "in a unified view." An example of this is iGoogle since this allows the user to provide many data sources that fit his/her needs and put them in the same window. The Google Maps example used earlier is another example of this. You can combine map data with other information, such as sporting event data, as was used in the example before, and create a new way to view the two sources together that is more helpful than either one separately. A presentation layer mashup makes it easier to view data from many sources at the same time, but does not provide any new information stream.

2.5.2.2 Data Mashups

A data mashup combines data together and presents it as a new data source. An example would be using a clock service with a scheduling service and an SMS messaging service to send an SMS message to a phone when certain events on the calendar are about to start. In this way it is using all previous services but it is using them in a new useful way. Mashups would allow services that did not previously have the capability to schedule events or send messages to be able to do both. This example shows how creating new services that would have previously taken much

expertise can be done with much less if the services are set up in a way that a mashup platform can use all of them.

Sometimes data mashups will provide a new information stream, but other times they will just present previous knowledge in a new way. Most data mashups fit within the consumer mashup area, and sometimes are referred to as situational mashups. There are many situations where there is a need or desire for a new service, such as the one mentioned above, that can be created if information is pulled from two or more sources. A mashup is created to solve the situation by satisfying this need or desire. This is what data and situational mashups do and in this way they are closely related to each other.

2.5.2.3 Process Mashups

A process mashup is the most complicated type of mashup. It is almost solely used in the enterprise mashup realm and is when two or more data sources are combined to govern a business process itself. An example of this is if shipment data is combined with weather, traffic, and truck location data to provide a way to affect the whole shipping process. Using this information, shipments might be handled differently depending on weather, traffic, or other factors, resulting in increased efficiency.

The reason most Google Maps examples do not qualify as process mashups is because while they normally provide a new useful way of looking at data and even allow the user to see data in a way that would not be possible outside of a mashup,

they do not affect any process. No decisions can be made from the combination of this data that could not be made before.

The inventory example used before is also an example of a process mashup. The data gained from this would affect how much inventory is produced at each plant and how the production levels can be maximized. The information gained to make these processes more efficient is the motivating factor behind developing process mashups.

2.6 Current Mashup Tools

For mashups to work there has to be an interface set up and a way to easily pull data from the different services involved. Several mashup platforms exist to make this process much easier. A notable consumer mashup platform is Yahoo! Pipes [17]. There were several other prominent consumer mashup platforms including Microsoft's Popfly[18], Google's Mashup Editor [20], and Dapper [21]. Google's Mashup Editor went out of service in early 2009 followed by Popfly later in the same year. It is speculated that it was not profitable enough for these companies to maintain their mashup platforms. Dapper also switched from being another well-rounded mashup platform to specializing in helping integrate ads onto web sites. The significance of these former platform producers is to show that there was not as much interest as was expected or that interest waned quickly or it was not profitable for these companies to maintain their consumer mashup tools.

While the previous platforms are tailored for consumer mashups, there are some business solutions that are made almost exclusively for enterprise mashups. In addition to IBM's Mashup Center mentioned before, JackBe [19] is another enterprise

mashup platform. The enterprise mashup platform services have been more successful than the consumer mashup platform services. One reason that would explain this is that there is more money at stake for enterprise mashups than with consumer mashups. It could also be that enterprise mashups have lagged behind and they will turn out to be unprofitable by the majority as well. The differences between consumer mashups and enterprise mashups will be discussed in more depth to help understand the reasons how enterprise mashups can best add value.

2.6.1 Consumer Mashup Platforms

Platforms for consumer mashups differ from those for enterprise mashups substantially. For consumer mashups a critical attribute is having as many data sources and mashups readily available and for it to be inexpensive or free. Since the target audience for consumer mashups is fairly adept at using mashup tools, the interfaces are not always simple, but made to be more flexible and to support a wide range of data sources. Consumer mashup platforms are made expecting to be used in making mostly presentation layer and data mashups.

2.6.2 Enterprise Mashup Platforms

For enterprise mashups, users are willing to pay the extra money to have an environment that is personalized for the enterprise. Furthermore enterprise mashup platforms attempt to create an environment where anyone in the enterprise is capable of making a mashup that fits their situational need. This ease of use leaves the mashup platform less flexible many times in that the data sources have to be more strictly formatted. This means there are fewer data sources to draw from but those

available are easier to use. Lastly and most importantly for our purposes these enterprise specific mashups are built to take the best parts of the consumer mashup world and make them work with the security concerns of enterprises [7]. Enterprise mashup platforms are made knowing many presentation layer and data mashups are still common, but must also be able to support the creation of process mashups.

2.6.3 IBM Mashup Center

The IBM Mashup Center [3] is an example of a well-established enterprise mashup platform. It promotes a large repository of enterprise data sources with its IBM WebSpheresMash software and IBM Lotus Widget Factory software. Both of these tools help users make their data sources into standard output feeds such as RSS, XML, or Atom feeds. The IBM Mashup Center only uses these feeds, so if some outside data sources need to be combined with internal, enterprise feeds, they either already have to be in this format or they have to be converted into one of these formats to be used. However there are not typically that many sources an enterprise would want to use outside of its own data sources and the standardized feeds make it much easier to create new mashups, allowing almost any employee at the enterprise to create a mashup if they have a situational need.

2.7 Mashup Advantages

There are several advantages to using mashups other than the ones discussed above- creating new services and information streams from existing data sources. While it is possible to make this new service without reusing existing services, it is much more efficient to use these previous services. Much in the way that object-

oriented programming encouraged easier reuse of code by modularizing every action and object created, mashups encourage the reuse of services by providing a framework to build on, namely platforms, such as the ones mentioned above, that facilitate mashing services together. Reuse of services helps in several ways as the object-oriented paradigm taught us. By creating a central service that gets used by all, the most obvious advantage is that people that come later do not have to spend time developing services that have already been created. Google Maps is reused so much in mashups because it is a very useful tool that fits in well with many applications, it has been developed in a way to tie in easily with other services, and it is a huge project way beyond any single person being able to develop on their own.

Another important aspect of reusing code is that it allows the developer to fix code in just one place. In the case of Google Maps, it would mean Google would be the only one fixing problems whether they be bug fixes, updates because of cities and roads changing, or upgrading to new standards. For enterprises this would mean letting each department focus on what it does best, then providing data used by other departments in an API. This would make it easier for departments to use other departments' work instead of recreating it. Service reuse also fosters competition and the best product possible by providing many more choices.

Another advantage to using mashups is that "they complement existing investments in IT because they can access and use existing applications in new ways" [3]. Service-oriented architecture (SOA) [12] is a great example of this. The SOA paradigm has been used much more in industry in the past couple years and is set up perfectly to work with mashups. SOA and mashups both seek to find existing

services and reuse them. The main difference is that SOA is still more rigid with the uses. An SOA is generally set up for a specific purpose and all services that are made are meant to be used toward that purpose. While services in SOA that are meant to be used together are loosely coupled, so that they can be used by more than one application, mashup services are generally even more loosely coupled. This allows a number of services to be used together that were not originally designed to be used together.

Figures 2.2 and 2.3, below, help show the differences between SOA and mashups. In both figures, each service is represented by a box. In figure 2.2, which depicts the SOA scenario, the services are combined in series, one from each group. For SOA to work correctly, generally one service from each functionality will need to be used, making it more structured which is useful in making it predictable and repeatable but not as flexible for making any service desired.

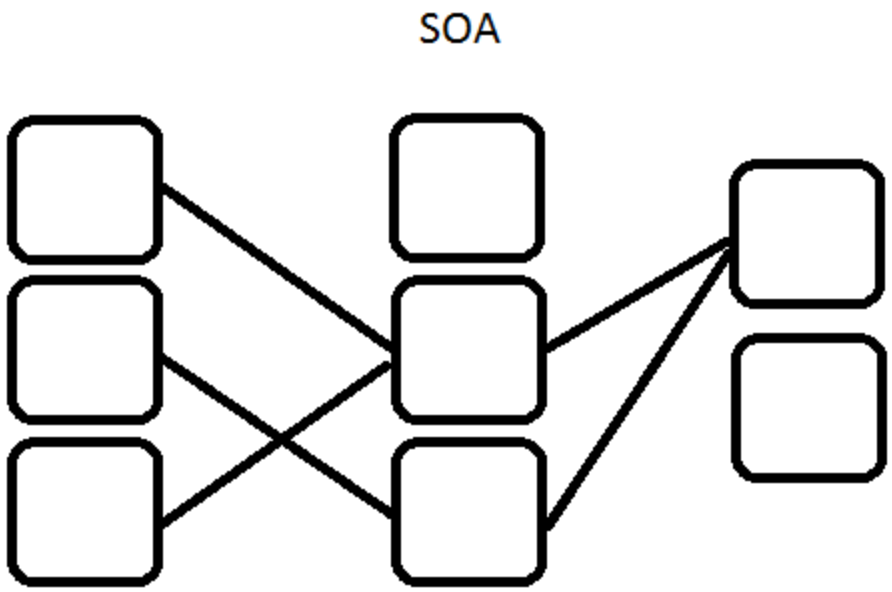

Figure 2.2- Service-Oriented Architecture

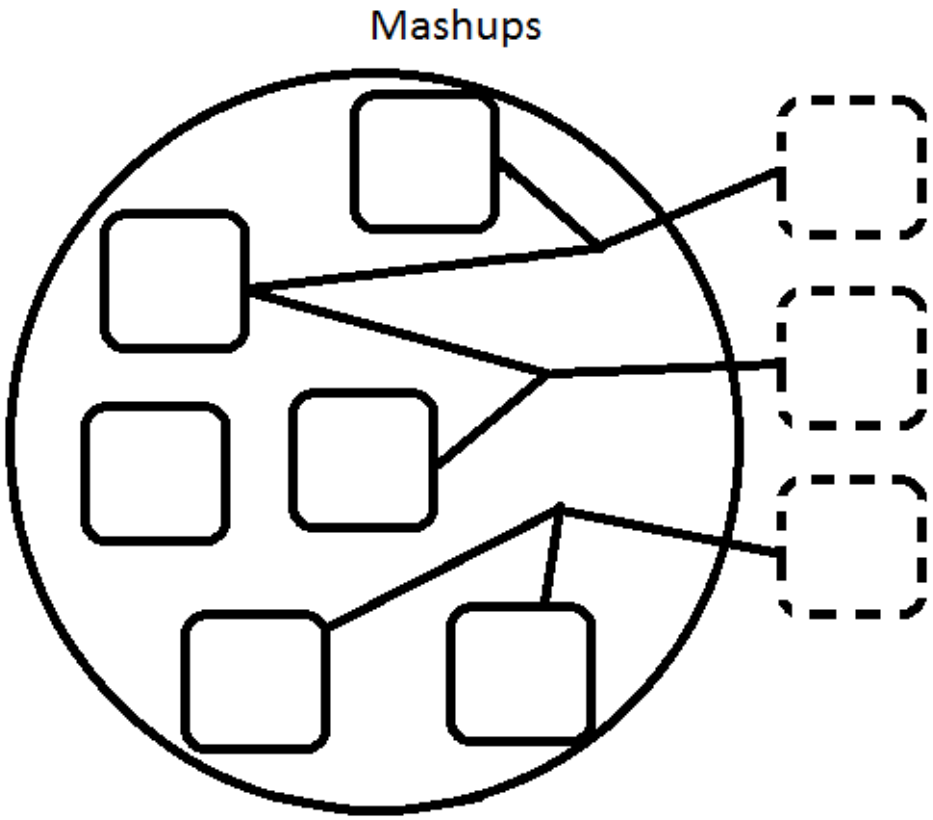

Figure 2.3- Mashup Architecture

In Figure 2.3 the solid boxes are random services provided on the web and the dotted boxes are mashups because they are new services created by combining two or more other services. The solid box services can have any functionality and can be used by any mashup. All these services are within a circle to show they are the pool of all available services, where the mashups are outside the circle. Sometimes mashups created are setup in a way that they are services themselves. There is little to no structure with mashups to contrast with how SOA generally operates. The differences between mashups and SOA will be gone into in more depth later on.

There are other similar applications such as portals and information management solutions that also fit in well with mashups. The Electronic-Freight Management (EFM) [2] initiative is another example of an application that appears to fit very well with mashups in the transportation industry. This is an initiative that also focuses on creating new information streams and attempts to cut out lack of knowledge as a reason for extra costs. The EFM initiative will be discussed more in this paper as it may become part of the supply chain management field and how they will work with mashups.

2.8 Summary

While there are certainly advantages to using mashups, there are disadvantages and some problems that have to be solved before they will be able to be used widespread in industry. Three issues that must be addressed are data source availability, security issues, and the amount of work to switch to a new methodology [7]. The problem with data source availability comes from certain APIs and data sources that are only available for consumer use, but have "licenses excluding their use for commercial purposes." This issue is not that large because enterprises could always develop the service internally or buy the service. The security issue is very important in enterprise mashups because of the nature of openly sharing information. Inherently mashups are meant to make data open, but for many enterprise uses, certain data and mashups need to be made private or only for certain uses. The vulnerabilities that need to be discussed by introducing mashups into the supply chain will have its own chapter devoted to it. The last issue is that it will be a matter of the costs of setting up enterprise mashups and the training and maintenance that goes

along with it. To word it more concisely, will mashups provide enough profit opportunities to exceed the costs? This issue will also be discussed later in this paper.

Chapter 3

Supply Chain Management

Maintaining a supply chain is a much more complex matter than it may seem. The simplest supply chain involves a single product directly transported between a buyer and a seller, within a single country, over a single mode of transportation, and involving no complex legal issues. Very few supply chains are this simple and most are considerably more complex. The issues that make supply chains more complex can be split into two categories for our purposes: those that are common and predictable and those that are unpredictable. While both create extra work, the later have to be solved quickly since nothing can be specifically set up for them beforehand because of the unpredictable nature.

3.1 Predictable versus Unpredictable Problems

The common, predictable problems mostly deal with complexness in the supply chain itself because of the size of an operation or distance an item needs to be shipped. Just the transportation component of a supply chain is normally comprised

of "shippers, forwarders, line-haul operators, local transporters, drayage operators, and brokers" [2]. To break it down even more, shippers can "represent a broad range of companies, including forwarders, manufacturers, and distributors" [2]. This shows how many different types of entities can be involved along a supply chain. The mode of transportation can be much more complicated as well since there are logistical problems in changing between truck, rail, and ships. If the shipment is international, then some export/import licenses and quota issues must be accounted for [5]. The more complex legal issues usually involve what party is liable at each stage of the shipping process to account for lost or stolen goods. All of these issues that have to be handled to make a supply chain work are just the common ones that come up in almost every supply chain. Figure 3.1, below, shows a common route that goods travel in a supply chain. Each arrow depicts when the goods and liability change hands. The port authorities are usually involved when importing and exporting issues are handled.

Figure 3.1- Typical Supply Chain

Outside of the problems that an expert in supply chain management has to handle often, there are going to be some problems that come up much less frequently or cannot be solved by the same systems. These problems are understandably tough to define, but can still be seen from some common occurrences. The weather is a problem that would arise often, but does not have the same solution every time. Depending on what type of weather is occurring, what type of good is being shipped, and many other factors the problem would be solved differently each time. There are many other issues that are similar to weather in that they require real-time data to solve. With these sorts of problems, it is important to make sure that enough information is available to make the best decision to keep the shipment arriving safely and as quickly as possible for as little money as possible.

3.2 Physical Security of Transportation

Common security concerns in most supply chains include both the theft of the goods being shipped and using the containers to smuggle goods. In either case the containers are kept locked as often as possible and are typically only opened to load, unload, or check the inventory at border crossings or similar security checkpoints. Container security is a subject not taken lightly but the number of ways a container can be broken into and the number of containers that must be watched at all times makes container security difficult.

The three main modes of transportation are road, rail, and waterway which includes both inland waterway and maritime travel. Large vessels used to transport containers by maritime shippers are very secure once loaded because the containers

are stacked so close that the containers below deck or on the inside of the stacks are blocked off completely. Figure 3.2 [30], below, shows how close containers are loaded together on a full container vessel and how hard they would be to access in transit. Furthermore consolidating so many containers means they can be watched at all times by a fewer number of people. It is much more difficult to watch all the containers transported by rail or road at the same time because they are not transported in as large of bulk. All three modes of transportation offer the advantage of being difficult to tamper with while in transit. Because it is difficult to break into a one while it is in transit, containers are least secure at rail yards, road stops, border crossings, and shipping/loading interchange terminal facilities, and ports [5].

Figure 3.2- Full Container Vessel [30]

Mashups offer some support in securing containers by making information more available, thus tracking containers easier. However mashups also introduce new

vulnerabilities by having more information available. If not properly protected, confidential information about the contents, location, or schedule could be leaked to people who would use the information maliciously. The specifics of ensuring only the correct individuals see confidential data provided through mashups will be looked at further in the next chapter.

3.3 Web 2.0 in the Supply Chain

It should now be clear how many factors go into decisions made in supply chain management and how helpful sharing information among different parties could be. There are many opportunities to make every day decisions in the supply chain more efficient and cheaper by combining information from all the different aspects of the supply chain. This is not a novel idea however. The Electronic Freight Management (EFM) initiative [2], mentioned previously, is working to help the supply chain management field cut costs and be more efficient throughout its processes. This initiative will need to be looked at more in depth to understand where it adds value and where mashups can still improve the process.

3.3.1 Electronic Freight Management (EFM)

The EFM initiative is one that will be closely related to mashups in the supply chain management field. EFM is focused on improving "operating efficiency, safety, and security of goods movement." [2] Improving these three areas should be the focus of incorporating mashups into the supply chain management field as well. Aside from their goals, EFM shares some other aspects in common with mashups. One similarity between EFM and mashups is that they solve problems by making

information more transparent between parties. "Potential benefits of using Web services technologies include improved shipment visibility throughout the entire supply chain, a reduction of redundant data entry, improved tracking, simplified interfaces with government authorities, and enhanced security." [2] EFM breaks the typical steps taken to transport goods in a supply chain into services. These services are loosely coupled and can be used by many parties, allowing EFM to be scaled to very large supply chains. The repeatability of the process is where most of the advantages mentioned before come from. Mashups could provide many of the same advantages, but there are still differences between the two. Finding the differences between SOA and mashups will help determine where each can add the most value to the supply chain management field.

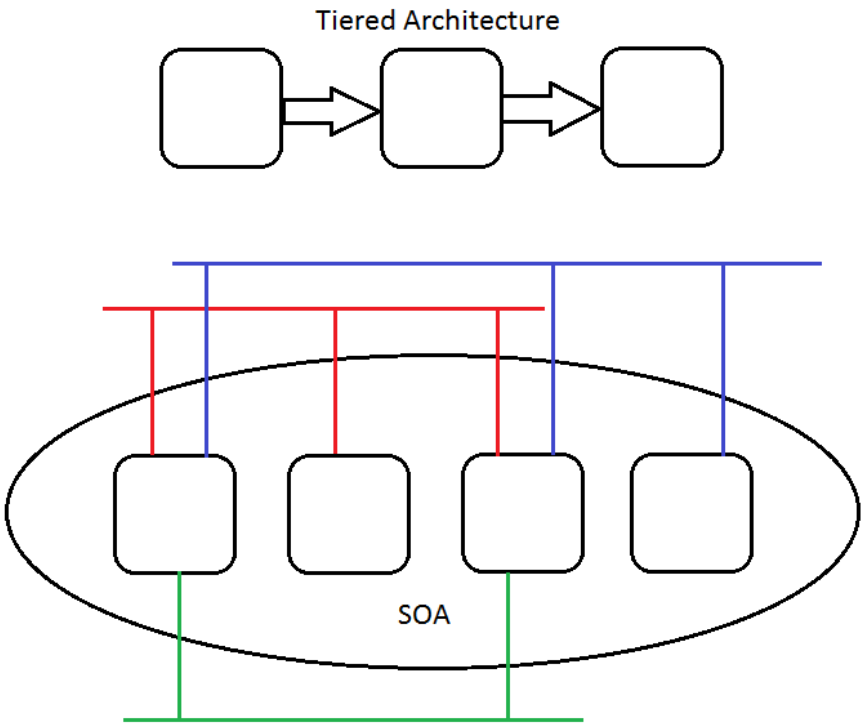

Figure 3.3- Tiered Architecture versus SOA

Figure 3.3, above, shows that when using a tiered architecture, services are used sequentially and are not loosely coupled enough to be used out of order. With SOA, different services can be combined to perform the same result. The idea behind a tiered architecture is to make a single service that provides the functionality desired. SOA makes several useful services that are loosely coupled to work well with each other, so they can be reused more easily. It is the loose coupling of the services that allows EFM to easily scale up and be used by many different enterprises.

3.3.2 Service Oriented Architecture (SOA)

There are several important characteristics of SOA that make it a useful tool for the supply chain management field. For one, SOA provides "loosely coupled services", so that they can work together even if they are on different platforms or made from different technologies. SOA also allows services to be easily changed and to be reused many times in the future because of the good level of abstraction it provides. Lastly, while current data interchange protocols such as the Electronic Data Interchange (EDI) [22] are not real-time, SOA provides real-time data that can help businesses make adjustments when something does not go according to plan [2].

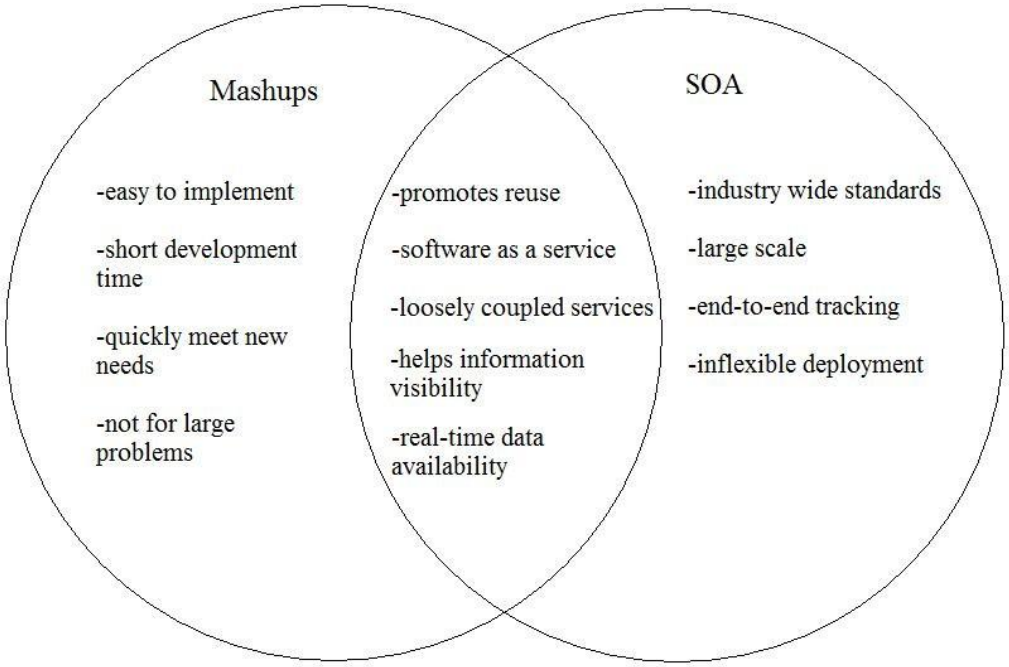

Figure 3.4- Mashups versus SOA

Figure 3.4, above, shows SOA and mashups have many of the same goals, yet they differ in many ways in how they go about achieving those goals. As stated above, SOA helps to make services loosely coupled which allows different entities to work together easier, and it provides real-time data. The downside to these large benefits is that it takes well-defined standards and setup for this to work. It also requires majority of parties involved along the supply chain to be using SOA for it to be really effective. Mashups are more of a lightweight tool that should be able to fit varying needs better, but may not be able to solve the larger, more complicated problems that SOA can. SOA is better fit for helping with larger well-defined problems, but mashups are well suited to help in some areas that EFM cannot. Mashups can provide real-time data better and can be developed quicker to solve smaller problem than the way EFM uses SOA. Instead of being a whole architecture

that is already set up from the start, mashups would provide businesses with more information to help them meet problems that come up during shipment.

This is where predictable and unpredictable problems come back in. SOA and therefore EFM are very good at handling the predictable problems that come up on a regular basis. EFM helps standardize the process which makes fewer problems come up in the first place and SOA makes it easier to solve the common issues that still do come up. Mashups are a good fit to solve the unpredictable problems. Since mashups have a shorter development time and can quickly solve problems, mashups are a much better fit to solve the unpredictable problems that come up in supply chain management. It must also be mentioned that mashups could not replace SOA in the EFM initiative. Mashups are meant to be lightweight applications that just could not handle the scope of the problems as large as SOA is meant to solve.

3.4 Mashups in the Supply Chain

The next thing to consider is how mashups should be implemented in the supply chain industry. There are going to be situations in which mashups would be used within a single part of the supply chain such as with a single shipper that is using mashups to make its own small part of the whole process more efficient. This part would be done much the way was talked about in the first chapter. By using one of many of the mashup platforms already available out there, mashups can be easily made to help solve business problems within a single enterprise.

Another application for mashups is larger scale and trying to solve the problems that come up between multiple parts of the supply chain. Mashups in these cases will

work best in conjunction with SOA in the EFM initiative. Mashups could be used as a layer on top of the SOA setup. SOA would do all the work of providing information from multiple parts of the chain in a secure, standardized manner, and the mashups would be able to stay lightweight. The key to this would be that mashups would have to be used to mash up data between companies only using data provided by EFM through their SOA setup. If there is no SOA already in place to take advantage of, more work will have to be put into making a central repository of services to use, but mashups still have value even with the extra effort.

Mashups have value in the supply chain management field mostly solving small, real-time data problems that SOA is not well suited for. SOA is another web service tool used by EFM that is particularly good at solving common problems in the supply chain. Mashups would be more likely to help out with problems that do not come up often, which means it is also more difficult to predict their true value. Mashups were also determined to be much less of an undertaking in terms of effort to implement than SOA. One concern that comes up specifically with integrating mashups into a supply chain is the large number of firms that have to work together in order to make them successful on a larger scale. This will be looked at more in depth later. Another issue is ensuring that mashups do not cause more security concerns than they solve. Trying to integrate mashups into the supply chain management field with minimal security risks introduced is the next topic that will be covered.

Chapter 4

Security

This chapter will provide an overview of common knowledge in the field of information security. The common security concepts used will be applied to use with mashups and in the supply chain industry. Lastly the problems with security as it pertains to mashups will be discussed.

4.1 Security Overview

This section will provide a review on a number of security concepts, threats, types of attacks, the attackers, and safeguards as they relate to the fields of computer security and data security and assurance.

4.1.1 Security Concepts

Before going forward some more basic concepts of data security and assurance must be reviewed. There are several concepts that have to be verified before data can truly be considered secure- confidentiality, integrity, availability, authenticity, and

accountability. Understanding these five concepts and verifying that mashups can address them will help ensure their security.

Confidentiality is broken up into two parts- there is privacy and data confidentiality. Privacy refers to one party's control over what information is provided and which other parties can view and disclose that information. Data confidentiality is assurance that the privacy rules are followed and that confidential information is not shared with or disclosed by unauthorized parties.

Integrity is also broken up into two parts- data integrity and system integrity. Data integrity refers to the assurance that data can only be operated on in the expected manner. System integrity refers to the assurance that the system performs its intended functions in the manner expected. Compromising either one of these integrity assurances leaves the whole transaction unsecure.

Availability refers to the assurance that a service is provided within an expected amount of time and that it is accessible by all verified parties. An attack that targets the availability component of data security and assurance is called a denial of service (DoS) attack. This attack disrupts services by destroying the timing of transactions or denying them completely.

Authentication is an assurance that both the consumer and the service provider are who they say they are during any transaction. While not one of the five main concepts, authorization is very closely related with authentication. Authorization is granting permission to a party for a service or system resource and ensuring parties that are not granted permission cannot use that service or system resource. The

reason authentication and authorization work so closely together is because after proving that a party is authorized for a service or system resource, you must also prove that party is who they say they are. In all current data transfer security mechanisms, both authenticity and authorization must be trusted for it to be a viable option. Many mechanisms that have either failed or become obsolete were because they sufficiently protected authenticity but not authorization or vice versa.

The idea of accountability was introduced assuming that the other security concepts will not always be perfect. With every security measure there is going to be tradeoffs. Many times to have some functionality, some aspect of the security cannot be perfectly protected. Accountability is the assurance that if something does not go as expected the source of the problem can be determined.

4.1.2 Attackers

An attacker, or threat agent, is defined as an individual or group of individuals that are intentionally trying to take advantage of a threat. Motivation for attackers can range from the thrill and self-acknowledgement of beating a system to stealing information or damaging an entity.

All attacks that attackers use can be split up into two types- passive and active attacks. A passive attack is one that will not affect system resources; the purpose of passive attacks is usually to attempt to gain knowledge or access to data without directly affecting system resources. An active attack attempts to affect system resources in a negative way.

Attackers can be broken up into internal attackers or insiders versus external attackers or outsiders. Attackers can also be broken up into three groups: hacker, criminal enterprise, both of which are external attackers, and internal threat.

The hacker usually breaks in through small vulnerabilities found in a system and sometimes will try to harm the entity, but many times the hacker is doing it more for the thrill or status of beating the system. Hackers that do not purposely cause damage are referred to as benign intruders and those wishing to cause damage are referred to as malign intruders.

The criminal enterprise is usually comprised of many actors working together to beat a system. Criminal enterprises are more dangerous because they can beat more complicated systems because they can share knowledge with each other. Furthermore, because they are more organized and act faster, they are tougher to catch. Also because criminal enterprises are more organized, they are usually malign intruders wishing to gain something for their effort or make an entity look bad.

When the internal threat penetrates a system's defenses, the attack does not always cause as much damage, but the internal threat is the hardest to detect and prevent. Some level of trust has to be given to insiders in order for them to do work, so when that trust is taken advantage of, it is difficult to prevent. Insiders are motivated by a feeling of entitlement or desire for justice.

4.1.3 Threats

When modifying the way any process is done, there is almost always going to be vulnerabilities introduced. New vulnerabilities must be identified and either secured

or accounted for as talked about with the accountability security concept. One way to do this is identify common threats then see which the new system is vulnerable to. There are four types of threats, or referred to as threat consequences by RFC 2828[14], which are unauthorized disclosure, deception, disruption, and usurpation. There are a number of attacks associated with each threat that will be discussed as well.

4.1.3.1 Unauthorized Disclosure

Unauthorized disclosure is when data is accessed by a party that is not authorized to access it making it a threat to the confidentiality concept. The four attacks that result in unauthorized disclosure are exposure, interception, inference, and intrusion.

The exposure attack can be performed by an insider intentionally releasing confidential information or it can be caused by an unintentional error that allows the release of confidential information. If it is caused by an error, it can be introduced by a human, hardware, or software. A very common example of this is one party accidentally releasing another party's confidential information.

Interception is when an outside party is able to access confidential data over a connection between two authorized parties. One common example of when this happens is on LAN networks because anyone can see packets between any two sources on the LAN. Another common example is how email and some other forms of data transfer are not secure in that they are not encrypted before sending them. If packets from one of these forms can be obtained, then it is a threat to confidentiality.

Another attack that results in unauthorized disclosure is inference, which is when an unauthorized party gains access to confidential information indirectly. This is done by using information that separately would not create a problem but combined can be used to break into a system. Traffic analysis, or analyzing information like packet size between hosts or time between transmissions, can give enough information away when analyzed over time to be a threat. Another good example is a technique in analyzing databases called tracking in which several queries that are all sufficiently protected on their own allow the attacker to infer information from combining what is learned from them collectively.

Intrusion is the last attack related to unauthorized disclosure and is when an attacker gets past the security measures to access confidential information. Since password protection is such a common practice for a security measure, beating a password system is a very common way for intrusion to occur. This can happen by a number of ways, whether it is too weak of a password that is beaten by brute force, based on a common phrase, or a number of other means.

Since unauthorized disclosure is a threat to releasing information to unauthorized parties, mashups will be vulnerable. Data sources that have confidential information will have to be protected. Several methods such as access control security models and API keys will be discussed later in this paper to analyze how well they would protect the data sources against attacks to the unauthorized disclosure threat.

4.1.3.2 Deception

Deception is when a party believes that data received is from a legitimate, trusted source when it is not. This is a threat to the integrity concept as it can affect both system integrity and data integrity. Usually when deception occurs it is because of vulnerabilities to the authentication security measures. Deception can be caused by three attacks- masquerade, falsification, and repudiation.

Masquerade is an attack resulting in deception that involves an unauthorized party pretending to be an authorized party and gaining access to a system or performing malicious acts in this way. While this attack overlaps with intrusion in that the attacker can gain access by way of beating a password in both, intrusion deals with data and thus confidentiality being compromised whereas masquerade deals with actions taken while on the system compromising integrity. An example of this is with a Trojan horse which mimics a valid, often well-known program and gets the user to run it which results in the malware gaining control of system resources.

Falsification is replacing valid data on a system with false data to deceive authorized parties. An example of this is if a person gained access to the IRS database and changed information in it to deceive the IRS about the person's true financial situation.

Repudiation is when one party refutes actions that it took in order to deceive authorized parties. It is a simple attack that can be hard to argue with since it puts all pressure of proof on the authorized party. A simple example is if a party denies receiving or sending data. You cannot make them use a signature, so there is little

way to prove that the deceiving party sent something. It is even tougher to prove they received it if you do not have access to their system.

Repudiation is the least likely of the attacks targeting deception that mashups will be affected by. While repudiation is still an attack that must be addressed, it is typically an attack that targets transactions that must be verified. These are usually routine transactions that will not be set up using mashups, but rather a more formal process like a Service Oriented Architecture as discussed in the Electronic Freight Management section earlier in this paper.

Mashups would be vulnerable to attacks resulting in the deception threat because of both the data sources they use and the services they create. Both the data sources and the services themselves would have to be protected. Changing data or altering a service's functions would be more severe than stealing confidential information as with unauthorized disclosure.

4.1.3.3 Disruption

Disruption is when system services are interrupted, slowed, or stopped to a level that impedes the system's ability to perform as expected. It is a threat to the availability and integrity concepts. There are three attacks that cause disruption which are incapacitation, corruption, and obstruction.

Incapacitation is an attack that leads to disruption in which a service is disabled by either an attack on the software or damage to the hardware that keeps the system from performing its intended functionality. This makes it an attack on the availability concept. A common example of incapacitation happening is when different forms of

malware are used to disable the software of a service. An example of an incapacitation attack on the hardware of a system is if the servers hosting a service were destroyed or damaged.

Corruption is when a system is kept from working properly by modifying system services or data. While similar to incapacitation, this attack focuses on modifying system functions or data, where incapacitation focuses on disabling their use altogether. When service use is denied or hindered, it is an attack on availability; when a service is changed, it is an attack on integrity. An example of corruption is if a party gains access to a system and changes the functionality of one or more services.

Obstruction is the last of the attacks causing disruption; it is when the communications of a system are kept from operating properly, overloaded with information, or disabled altogether. This is another threat to availability. A broad example of this attack is the Denial-of-Service (DoS) attack. A common way to deny a service is to overload servers or a whole network with more requests for a service than it can handle.

The security in attacks targeting disruption has to be considered because of the threat it poses to both services and data sources. Both services and data sources can be disrupted and cause processes that rely on them to be disrupted as well. For some applications, disruption will affect the integrity of the data. More often, disruption will affect the availability of the services and data sources. The interruption of

availability of services is the main component that needs to be looked at because of disruption.

4.1.3.4 Usurpation

Usurpation is when an unauthorized party gains control of system services. This is a threat to the integrity concept as it relates to system integrity. There are only two attacks that are considered a cause of usurpation which are misappropriation and misuse.

Misappropriation is one of two attacks that target the threat of usurpation. This occurs when a party seizes control of a service or whole system and uses it for means other than it was intended. An example would be if a hacker gained control of a system and used its processor to carry out an attack on another system. This would result in the system resources being used for a purpose other than what they were intended and would be considered a threat to integrity.

Misuse is the other attack that leads to usurpation and is when a system is broken into and the system security is disabled. This attack is intended to bypass the security in some manner and make it easier to do whatever the attacker wants to the system. Any attack, including many types of malware, that beats the security of a system is considered a misuse attack. Misuse is a threat to the integrity concept as well.

Attacks that result in usurpation would target the new service created since by definition if they targeted the data sources used they would be attacks targeting unauthorized disclosure. For much the same reason that mashups can consider any of these attacks less than others, it is because the services created are not as important as

protecting confidential data sources the enterprise might be providing. These attacks are dangerous to mashups in the supply chain mostly if the data sources are not well enough abstracted from the services they create. This means services created must be protected in the same way that the data sources used to create it are.

4.1.4 Safeguards

Two important terms to know when designing or considering network or information security are Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). Both are broad terms that attempt to protect a system in different ways.

4.1.4.1 Intrusion Detection Systems

Intrusion detection systems monitor different parts of a system, such as network traffic or system events, to discover attempts at accessing or altering unauthorized system resources. The idea behind intrusion detection is that successful attacks will happen, but another safeguard is to have ways to detect if an attack has happened and restrict damage at that point by limiting resources or alerting authorities. An IDS is an example of an application of the accountability security concept since it is a measure that helps track and detect attacks instead of actually prevent them.

Intrusion detection systems themselves can be broken up further into host-based IDS's and network-based IDS's. A host-based IDS is designed to analyze the events occurring on each host, whereas a network-based IDS monitors network traffic to try to identify possible attacks. Both forms of IDS have their advantages and have certain attacks they are better at detecting.

Intrusion detection systems have three components: sensors, analyzers, and user interface. The sensor is the part that collects the data and sends it to the analyzer(s) to be analyzed. Sensors can collect their data from network packets, system log files, and anything else that holds useful information. The analyzer takes the data from the sensor and determines if an intrusion has occurred. Trying to minimize false negatives and false positives is the goal of most analyzers. The user interface is the way that the IDS conveys to a user the information found from sensing and analyzing the system.

4.1.4.2 Intrusion Prevention Systems

Intrusion prevention systems monitor parts of a system and attempt to detect a possible attack before one ever occurs. Intrusion prevention systems can work in two ways. One way is that they act much like an IDS in that they analyze a host or network for packets that could possibly be an attack, except that an IPS can block the packets and keep them from ever affecting the system in the first place. The other way is that a firewall acts as an IPS and detects attacks before they ever get inside the system. Intrusion prevention systems can also be broken up into host-based and network-based and work in much the same way that intrusion detection systems do.

4.1.4.3 Safeguards Used with Mashups

Intrusion detection systems and intrusion prevention systems are both important to protecting mashups in the supply chain. Since it is better to prevent attacks altogether, intrusion prevention systems are preferred, but it is unrealistic to think that all attacks can be prevented. Intrusion detection systems offer a good way to protect

against gaps in the security such as mistakes made by employees and new vulnerabilities that show up.

4.2 Security as it Pertains to Web Mashups

Analyzing the changes that are made to a system by introducing mashups will help point out where vulnerabilities are introduced. In the case of mashups, one element that is subject to change is how much data is shared. The differing amounts of data shared can be analyzed by looking at presentation layer, data, and process mashups separately.

4.2.1 Vulnerabilities Introduced by Presentation Layer Mashups

Presentation layer mashups should have the least security concerns because they are the most straightforward. These mashups will usually be displaying data in a different manner, so the only real concern is restricting the data that is provided. If only public data is being used, then there should be no concerns at all. If data is being provided from one or more parties within the supply chain, then APIs will need to be set up for the mashups to use. This means trusting one or more people to determine what data can be provided by the APIs. There is more of a security concern if the APIs will be made into public data, but there are still some problems even if the APIs will be internal to the parties involved. Ranging from public to only being accessible by the party that created an API, the scope of APIs will be defined in this paper as all the parties the data is available to be used by. Because presentation layer mashups only present data already used in a different way, DoS attacks would not affect the mashups anymore than they would the data sources looked at without mashups.

4.2.2 Vulnerabilities Introduced by Data Mashups

For data mashups there will be a few more vulnerabilities simply because they are more complex than presentation layer mashups. With data mashups, not only do the APIs need to be considered, but the new data source that is created needs to have its own scope as well. This is the only time that data mashups would be different from presentation layer mashups. The rest of it would be done much in the same way. There would still need to be one or more people that are trusted to mark APIs with a certain scope based on who should be able to use them.

4.2.3 Vulnerabilities Introduced by Process Mashups

The process mashup is once again the most complicated but is also the most important for the purposes of figuring out how mashups will work in the supply chain industry. While not always true, it would be much more common for multiple parties to be involved over the whole supply chain with a process mashup than with the two other types of mashups. If there are more parties involved, the scope will many times be more complicated.

For instance, if there is a single party involved, the scope is basically broken down to being public or private data (private being internal to that party). If there are multiple parties, then it has to be decided which APIs are available to each party. To help illustrate this point further, refer to Figure 4.1, below. Mashup 1 is created by combining data sources 2 and 3. Party A might not want Party C to use Data Source 2. This would mean Party C also cannot use any data provided by Mashup 1. Complexities that come up like this will need to be handled in some way.

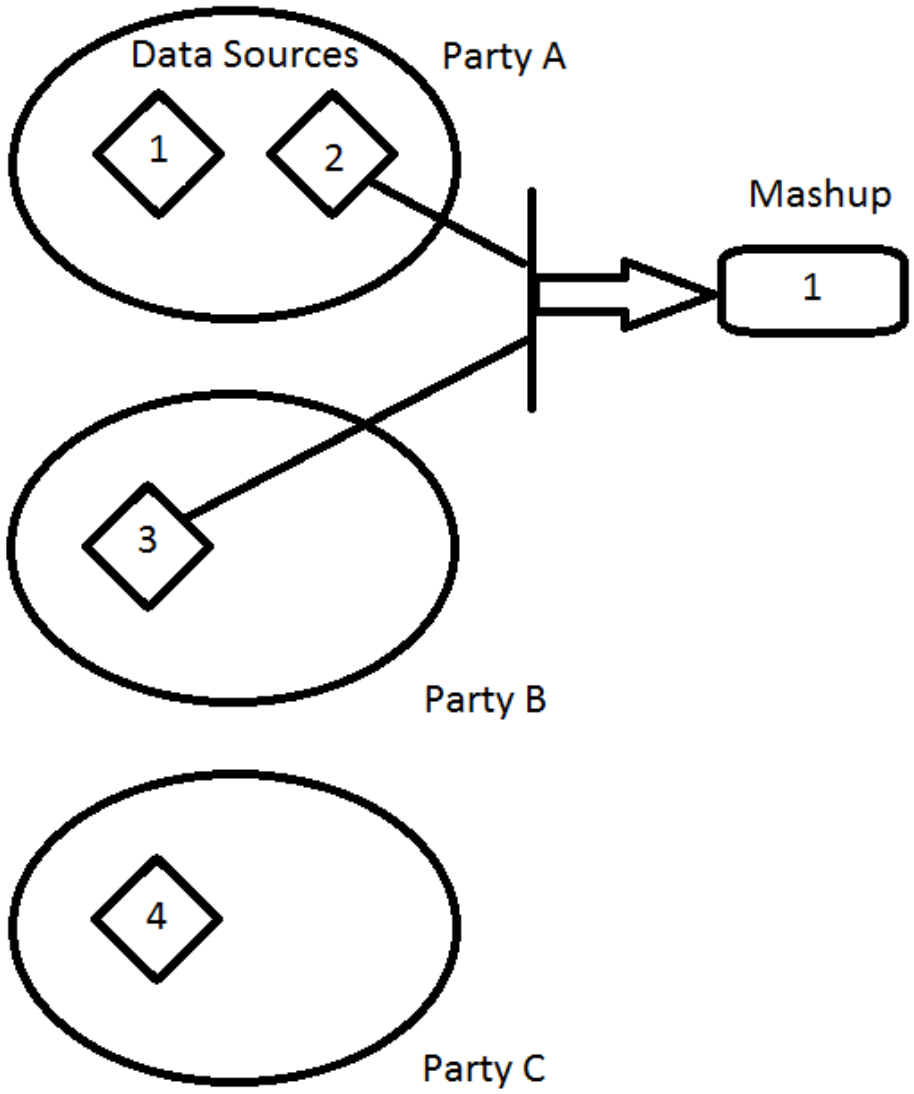

Figure 4.1- Conflict of Interest Mashup

4.2.4 Security Models and Access Control

There are a number of security models designed to help with the security issues that come up when sharing data among multiple parties as would happen with mashups. The Bell-La Padula, Biba Integrity, Clark-Wilson Integrity, and Chinese Wall Models were all designed to formalize access control to help protect confidentiality and integrity issues [13]. The idea of access control is to limit access

to parties that require access. Each of the models specifies how to determine which parties require access in a different way and thus each is good for a different situation.

4.2.4.1 Bell-La Padula Model

The Bell-La Padula Model or BLP model is built around a hierarchy of security levels. Each subject and object is assigned to one of these security levels. An object is classified as at this level of security and a subject has clearance at this security level. An example Stallings and Brown [13] use to show this hierarchy of security levels is the security classification scheme the U.S. military uses:

Top Secret > Secret > Confidential > Restricted > Unclassified

The next important point to understand about the BLP model is that it defines four different access modes. The four modes are read, append, write, and execute. Read implies that the subject has only read access to the object. Append refers to the subject only having write access to the object. Write refers to the subject having both the properties of read and append, so both read and write access to the object. Execute means the subject has neither read nor write access to the object, but it does have access to execute the object.

The parts of the BLP model described up to this point are all terms and definitions of the BLP model, but do not actually provide any protection to confidentiality, which is what the BLP model specializes in. There are three rules that are followed that help provide confidentiality when the subjects and objects are classified into the correct security classes. The three rules are the simple security property (ss-

property), the star property (*-property), and the discretionary security property (ds-property).

The ss-property follows the no-read-up rule which means that a subject can only read objects at the same security level or lower. The *-property follows the no-write-down rule which means that a subject can only append to objects at the same level or greater. The ds-property allows for discretionary access. The specifics of it are not important for our purposes, but it allows for the owner of an object to specify access to that object to different subjects, so long as the first two properties are not violated.

4.2.4.2 Potential Use of Bell-La Padula Model in Mashups

The BLP model is one of the simpler models, so the ideas of its security levels, access controls, and rules to preserve the confidentiality of all objects in a system are all important. The object would be a data source that could be used by a mashup. The subject would be whichever enterprise or individual wanted to use it. So this setup would be very good at keeping things confidential if the security level and scope were uniform across several enterprises.

The limitation of this model in the supply chain is that it can only set an enterprise at a certain security clearance and it is the same for all data sources. The BLP model is built to work within a single enterprise and does not have a formal model for multiple enterprises as would happen often in a supply chain. Therefore, the BLP model could still be used by a single enterprise, but the model does not work well across multiple enterprises. The example depicted by Figure 3.1 and described above shows this well by showing how Data Source 2 would need two different security

levels to allow Party B to use it but still keep Party B from accessing Data Source 3. The BLP model is not set up well for that, so it would not be the best model for mashups.

4.2.4.3 Biba and Clark-Wilson Integrity Models in Mashups

While the BLP model is designed to protect the confidentiality concept, the Biba Integrity model protects the integrity concept. It has four access modes as well: modify, observe, execute, and invoke. The specifics of the access modes are not important for our purposes, but they work much the same way that they do in the BLP model. Another similarity between the BLP and Biba models is that they each have three rules that specify which subjects can access different objects and which modes they can access them by. The rules are similar to the BLP model and can be considered the same for our purposes as well. The Biba Integrity model would have the same problem as the BLP model however; it is not set up well for multiple enterprises and the varying security levels for a single object.

The Clark-Wilson model (CWM) also focuses on protecting the integrity concept instead of the confidentiality. The CWM differs from the Biba and BLP models in several ways. It defines two data items, constrained and unconstrained; two procedures, integrity verification and transformation; and two policies, well-formed transactions and separation of duty among users. Despite the differences, the main problem with the CWM being used in conjunction with mashups is the same as with the Biba and BLP models- the CWM does not work well with varying security levels for a single object.

4.2.4.4 Chinese Wall Model in Mashups

The Chinese Wall model is very different from the other models mentioned and lends itself to mashups in the supply chain industry much better. The Chinese Wall model was actually designed for commercial applications, especially financial and legal industries. It protects both the confidentiality and the integrity of the system when implemented correctly. The subjects are done the same way and there are still access rules, but the objects are done differently.

Objects are a single type of information, with datasets and conflict of interest classes the other two. All objects can still be considered data sources for our purposes, but under the Chinese Wall model every object is part of a dataset. Each dataset represents all objects that belong to the same enterprise or company. Every dataset is part of a conflict of interest class which keep enterprises of the same industry in the same conflict of interest class.

This hierarchical structure helps keep the two rules used in the Chinese Wall model working correctly. They were named the simple security rule and the star property to show that they work the same as the BLP model. They differ slightly because they have to consider the datasets and conflict of interest classes now, but the idea is the same. They are set up in a way to protect both integrity and confidentiality. Figure 4.2, below, shows the hierarchical structure of the Chinese Wall Model. If a party read from Shipper 1, then that party could no longer read from Shipper 2 because they are in the same conflict of interest class. If a party wants

write access to object E, then that party cannot have read or write access to any other object to protect the party from accidentally spreading other objects' data.

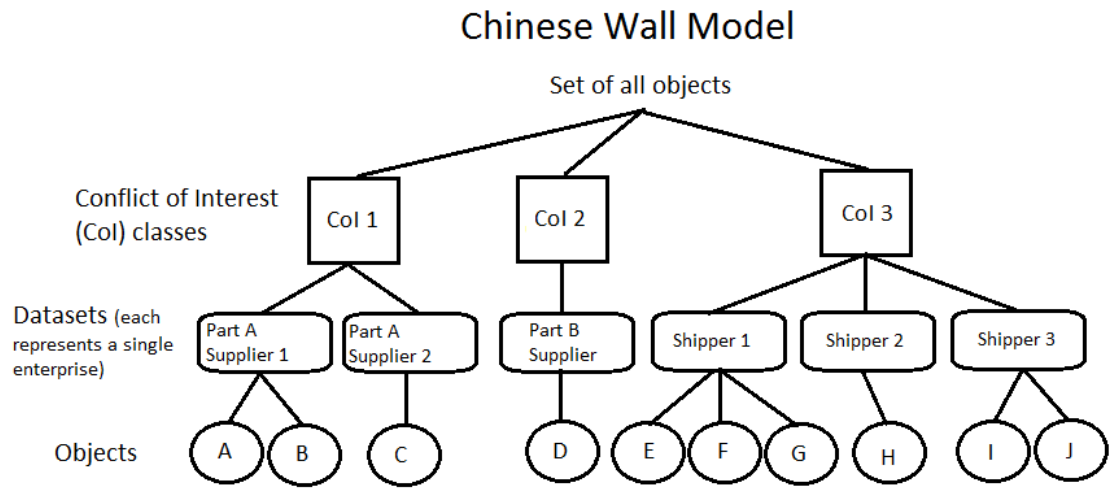

Figure 4.2- Chinese Wall Model

The Chinese Wall model fits very well with the needs of mashups in the supply chain. If there are several enterprises, they can be separated into different conflict of interest classes to help secure them. The model still does not work perfectly since a single enterprise may have some data sources it wants as more secure than other ones and the conflict of interest classes do not handle that aspect. The Chinese Wall model has all objects of a single enterprise underneath the same dataset. The model would have to be modified some to work with the supply chain industry, but it is something to start with.

4.2.5 Current Mashup Security Plans

This section will look at a formal plan that was developed to protect privacy in government mashups. Then API keys will be reviewed to see if they offer the security necessary for mashups in the enterprise setting.

4.2.5.1 Model for Privacy Protection in Government Mashups

One instance of a security plan that is related directly to data protection in mashups is a plan set forth to protect privacy in government mashups [6]. The plan includes Regulatory Privacy Policies, Personal Privacy Policies, and Mashup Privacy Policies. All of these policies are "uniformly specified based on five aspects of the data and mashup service: DataType (DT), Linking Parameters (LP), Operations (OP), ProviderType (PT), and MashupPurpose (MP)." The Personal Privacy Policies would probably not be nearly as relevant to the supply chain industry since they are meant to deal with the security involving the individuals using the mashups, but the idea of privacy policies in general with the five aspects can be used. The Regulatory Privacy Policies and the Mashup Privacy Policies would still be relevant to the supply chain.

The model for privacy protection in government mashups is mostly a template for how to set up and use privacy policies to help keep track of the scope of all the data and figure out which parties are authorized to use the data. However it does not go into the specifics on making sure the users are authenticated. One mechanism that has been used to help authenticate users of mashups and other Web 2.0 services is the API key [10].

4.2.5.2 API Keys

An API key is a means for security that service providers sometimes require. It can be used in different ways: as a way to limit users by only giving out a number of API keys or even as a password of sorts by making and tracking a unique key for each user. Either way the API key is used it helps restrict usage of a service to fewer people. This helps both with intrusion prevention by adding another means of security intruders have to beat before they break into a system and with figuring who an intruder could be when a system is compromised by limiting the search space of possible attackers.

Now to discuss the way that API keys are used; first the consumer must establish an API key and consumer secret upon registration. In some cases the API keys will be distributed to all parties that the service will be available to and the consumer secret will be established upon consumer registration. After registration, the user can use the hash algorithm set forth in the service provider documentation to hash together the API key and the secret key to create a MAC (Message Authentication Code). This MAC along with the parameters necessary is sent in an API request to the service provider. The service provider sees the API key, looks up the corresponding secret key, and uses the same hash algorithm to generate a new MAC. If the new MAC matches the one sent, then the message is both authenticated and has proven its integrity to be upheld within a reasonable doubt. The service provider can then give a response to the consumer and continue with the API usage.

4.2.5.2.1 API Keys with Mashups

Mashups could use API keys as a means of security for the API data sources. The API keys would be distributed to any party that the data source owner deems trustworthy of using the data source. While these keys were made for APIs, the same idea could be provided to other non-API data sources like feeds and widgets. It would be a "data source key" that allows you to subscribe to a feed, widget, or service.

4.2.5.2.2 API Keys with the Supply Chain

The idea of an API key would fit well in the supply chain industry because it addresses the issue of allowing some parties access to data while denying others access. One problem with using API keys in the supply chain industry is that they lack the security level tiers of the access control security models that were reviewed. One advantage to the security level tiers is that each object is set to a security level and each subject gets a security clearance, so the access level decisions only have to be made once for each object and subject. With API keys, the access decision would have to be made once for every object/subject combination. Table 4.1, below, displays how many decisions would have to be made for security level tiers and API keys based on the number of objects (N) and subjects (M).

Table 4.1- Number of Decisions for Security Level Tiers versus API Keys

	Security Level Tiers	**API Keys**
Number of Decisions	N+M	N*M

So API keys seem to have the disadvantage of taking more decisions for larger values of N and M. This is not too important for small values of N and M, but for larger values of N and M and when a committee has to meet for every decision, it could be very costly. However API keys have the advantage of being able to specify more specifically which objects can be used by which subjects. One of the main problems with the access control models is that they did not have a way to handle this. Furthermore API keys could be used in a way that they would work like an access control model. This would be done by giving out keys for different security levels. They would still be used like API keys but all enterprises with the same security clearance would have the same key and the keys would verify security clearance instead of the enterprise accessing the data source. So in this way API keys are more versatile and could be used for whichever way deemed the best.

4.2.5.2.3 API Key vulnerabilities and Standards

There are a number of vulnerabilities that Stephen Farrell discusses in his assessment of API keys [10]. He reviews the problems with the MAC algorithms used often with API keys, how Secure Socket Layer (SSL) and Transport Layer Security (TLS) are used in a not completely secure manner, and how API keys cannot be used entirely securely with open source software. Because of all of these vulnerabilities Farrell gives his own set of suggestions for using API keys and provides a set of standard specifications used for API keys.

The standards that Farrell mentions are the OAuth Standards Specifications [23]. Following these standards can help greatly reduce the vulnerabilities and the impact

of an attack when using API keys. The standards include practices that would help avoid the scenarios Farrell discussed as well as some more obvious ones that he did not.

4.3 Conclusion

This chapter reviewed the common terms and practices used with computer security and data security and assurance. This chapter also revealed the main vulnerabilities that are presented by using mashups in the supply chain, then discussed techniques already in practice to protect against these vulnerabilities. The main security issue to mashups in the supply chain is ensuring that only the correct parties can view confidential data. The Chinese Wall Model was determined to be the best security model to help ensure only the correct parties are authorized to see confidential data in a supply chain. API keys were determined to be a way to authenticate that users are who they say they are in a supply chain. Now that the background information of mashups, supply chains, and data security have all been discussed, the next chapter will discuss all of what has been discovered in these topics combined.

Chapter 5

Mashups in the Supply Chain Analysis

Now that mashups, supply chain management, and security concepts have all been reviewed separately, they will be analyzed together to see if mashups are worth pursuing in the supply chain industry. First some terminology and examples of how mashups might be used will be discussed to ensure the discussion of the worth of mashups and how they will be used is clear. Next each of the questions that was set out in the introduction of this paper will be answered to help determine if mashups are viable in supply chain applications. The questions that need to be answered are:

- Will mashups work in an enterprise setting?
- What advantages does using mashups in the supply chain offer?
- What are the main costs of implementing mashups?
- Can mashups be implemented in a supply chain in a secure way?
- Will enterprises share information openly?

After answering these questions, use can determine if mashups will be worth using in a supply chain setting. Thus the final part of this chapter will be a conclusion deciding if mashups will be worthwhile.

5.1 Background Information

One important distinction that must be made is the difference between mashup developers and mashup users. A mashup developer is a person that programs a mashup. Programming a mashup entails combining the information from two or more sources, usually in their XML formats, to make a new service. Developers can make the XML formats themselves and combine the data sources using their own tools or mashup platforms such as those provided by IBM [3] and JackBe [19] can be used to help combine data sources. A mashup user is a person that uses the mashup that was constructed by the developer. The developers can also sometimes be the users, but in other cases there are some people that develop the services but do not use them.

A second important point to discuss is the media over which the services created will be used. Users can utilize mashups on any media that they are developed for. Services will be desktop applications, applications for mobile devices, or in some cases will be available for both. The type of media that the services are used on is important to distinguish because desktop applications and mobile devices will have different security implications, so they must be looked at separately.

Now some examples of developers, users, and the services that could be created in a supply chain will be discussed. Presenting typical situations of how mashups

might be used in the supply chain will facilitate the discussion of the worth of mashups by having specific cases to refer back to. One of the scenarios that will be looked at is an application for mobile devices that is developed by people in the IT department and used by truck drivers. The second scenario presented is a desktop application that is both developed and used by logistics experts.

In the first scenario an IT department creates a mashup served to help truck drivers better avoid traffic and weather. The application is intended for use on mobile devices to allow truck drivers to access this service wherever they have cell phone service. The idea would be to create a service that uses another service for directions such as Google Maps [31] or Mapquest [32] and combine that with traffic and weather feeds from large cities and highways along the way. The data gathered could be presented to the truck driver to avoid the poor weather or high traffic areas or if it were a more sophisticated service, it could come up with a new route and present that to the truck driver.

For the second scenario there is an idea proposed in the supply chain industry that would use mashups to help make shipping more efficient we will call collaborative cargo management. This scenario came about because there are many small and medium sized enterprises that sometimes ship containers that are less than full simply because they do not have enough goods to ship. Collaborative cargo management, at its very simplest, is taking two or more of these less than full containers and combining them to ship them both at a smaller cost for both parties involved. While there are already forwarding enterprises that consolidate shipments into fewer containers, having this information available to be used in mashups would allow

many more enterprises to save money by doing the work themselves instead of hiring these consolidation companies. In this scenario, if enough small and medium sized enterprises provide shipment data, an outside company would not have to be hired. Any employee that knows about outgoing shipments could develop a mashup that finds other enterprises shipping along the same route and contact them about consolidating shipments. The mashup created could be developed by the same person that would be using it or be developed and used by separate people. This scenario shows why the question of 'will enterprises share information openly?' needs to be answered. Without enterprises sharing at least some information openly, this scenario would not be possible.

5.2 Mashups in an Enterprise Setting

As discussed in the first chapter, mashups have been popular on the consumer side, but there is not nearly as much recorded about mashups being successful in an enterprise setting yet. Deciding if mashups would work in an enterprise setting was one of the questions this paper set out to answer. This chapter will decide whether mashups will work in an enterprise setting or not by looking at the different types of mashups and applying the scenarios described above.

5.2.1 Types of Mashups

The varying types of mashups were all discussed in chapter one, but they were only reviewed in a general sense. Applying what has been discussed about supply chains and the mashup security to the different types of mashups is important to understand if mashups will work in an enterprise setting. While consumer versus

enterprise mashups were reviewed before, only enterprise mashups need to be looked at for the purposes here. Presentation layer, data, and process mashups will all be compared since they can all be used in the enterprise setting.

The presentation layer mashups offer some advantage in the enterprise setting. They could be used by individuals to help display shipping, inventory, or weather data based on different locations. Different departments or even individuals could customize the data they want presented. As discussed in the security chapter, presentation layer mashups would be very safe because they rarely use new data sources. With presentation mashups the value is not gained by using new data sources, but by presenting those already used in a different way. Because of this, the few times that new data sources are used, they should be verified, but many times, no extra work would be required.

Data or situational mashups are very useful to the supply chain industry because they work well with real-time data. There is much data in a supply chain that is changing between every shipment. If the differences in shipments could be looked at quicker, more optimizations could be made to the process. In theory, data mashups provide this opportunity by providing near real-time shipping, inventory, cargo, and weather data in an easy to use format that would allow individuals at all levels of the enterprise to make a service if they see a need. One concern is if the mashup platforms available make it easy enough for the developers to make new mashups. If the platforms do not make it easy enough for many people to develop new mashups, then most of the workload still falls on the IT department. One of the advantages for

mashups is that it allows many people to develop them and this advantage would be nullified if the mashup platforms are difficult to use.

Data/situational mashups would have to consider security more, but it would still be possible to use them securely within the context of a supply chain by verifying the integrity of data sources. First the data sources used to create the mashup would have to be considered. With data/situational mashups, the data source would be determined by the need instead of simply presenting data already provided. This means more data sources would need to be verified to be safe before being used. The data source created would probably not need to be considered much more than with presentation layer mashups. The main use of data mashups with the supply chain industry would be for cheap, one-time use sort of mashups that would not be creating a new data source. For the mashups that are creating a new data source, the data provided would certainly have to be reviewed to ensure it would not be sharing confidential data.

Process mashups are the most complex and can help a business the most if done correctly because they can affect an entire business process which offers much more opportunity for optimization. The problem with process mashups is that they are not usually as lightweight as mashups are designed to be. If the mashup is actually changing a business process, it would not be a lightweight component that can be made quickly and thrown away because it was inexpensive. For a business process, the service needs to perform very accurately and consistently and cannot be thrown away as easily. For this reason the data provided by the mashup should be reviewed and everything should be double-checked by the IT department. Process mashups

can still be used, but they come with more overhead, so this should be considered when deciding on the value of using process mashups.

Process mashups would have many of the same security concerns as data/situational mashups from the data sources they use. However process mashups would create a new data source more often since that is one of the main advantages for the extra effort spent in making a process mashup. As discussed in the security chapter, the data source created would have to be reviewed by an individual or group of individuals to ensure it is not providing confidential data to the wrong parties. Below, Table 5.1 is provided to show how useful each type of mashup would be in the supply chain and the security implications.

Table 5.1- Types of Mashups Uses and Concerns

	Presentation Layer	**Data/Situational**	**Process**
Usefulness in supply chain	Moderate	High	Moderate
Data source security concerns	Very few	Many	Many
Data provided security concerns	Very few	Few	Many

5.2.2 Mashup Scenarios in the Enterprise

Now both of the possible scenarios that were presented earlier will be analyzed to see what type of mashups they would use which will help determine if they would work in an enterprise setting. If either scenario would work in an enterprise setting, then there is at least one example of how mashups could help in the supply chain

industry, so they should still be scrutinized further. If neither scenario would work in an enterprise setting, then the possibility that mashups will not work in an enterprise setting needs to be explored further.

The first scenario that should be looked at in an enterprise setting is the one with the truck drivers that use mobile applications to help avoid traffic and bad weather. In this scenario, the mashup used would likely be classified as a data or situational mashup. The mashup created would provide a new service by combining several different data sources but would not be changing an entire business process, so it would not be considered a process mashup. With this scenario, enterprises would not need to share data. Any shipping enterprise could use this idea and make it as simple or as complex as they chose. There scenario seems a good fit for an enterprise setting.

In the second scenario, there is a mashup created to help parties consolidate less than full containers to save money. This mashup would change the way the entire shipping process works and would thus be considered a process mashup. There would be many parties involved for this to work correctly, making it likely to be implemented in several enterprises at once which would mean possible problems with enterprises not being willing to openly share this data. Also because it is a process mashup, it will have more security concerns that need to be looked at later. These possible problems that need to be looked at more do not affect the fact that this scenario describes a mashup that is well suited for an enterprise environment.

Both scenarios seem that they would work well in an enterprise setting. The concerns that mashups might not be as well suited for enterprises as they are for consumer use seem to be unfounded by these scenarios. The advantages mashups provide will be looked at again to determine if they will still be useful in the supply chain industry.

5.3 Advantages of Mashups in the Supply Chain

Some general benefits mashups provide have already been reviewed in the first chapter. Mashups also provide some benefits that would be unique to the supply chain industry. Some of the general mashup benefits previously discussed were reuse of services, central repository to maintain and fix code, and use of existing systems such as SOA. Some of the advantages to using mashups that are specific to the supply chain industry are allowing many parties to more easily share data and making situational needs able to be quickly accommodated.

With most supply chains there are going to be many parties involved along the way. If mashups were widespread in the industry, it would allow each party to provide data in publicly available services if it is information that is safe to have open to anyone. All parties could benefit from having more access to each other's public knowledge as shown with the collaborative cargo management scenario. However if few parties are involved or little data is made public, it does not offer much advantage to the remaining parties. For some enterprises, the data necessary to perform collaborative cargo management might be considered too sensitive to share with other parties. Also if the data is not provided in the same formats across the industry, it

would not be as easy or quick to use, so not nearly as useful. Collaborative cargo management does not require every party along a supply chain to be involved. Any enterprise that provides the necessary information to make collaborative cargo management work could benefit from mashups. The main point to draw from mashups being used to facilitate collaborative cargo management is that while mashups become even more useful if many parties provide data openly, there is still benefit in using them with only a few parties involved.

The supply chain offers many situations that could use real-time data to help enterprises quickly solve problems. Mashups fit this role well because they can be set up to provide real-time data and they are meant to make quick solutions that are not necessarily robust but can be developed very quickly and used by less technical people. The main issue that still faces mashups in this context is that while they have been advertised as being suitable for less technical employees to use, there have been few platforms that have managed to make it easy enough for anyone to develop. Most mashups will still be developed by people with expertise and sometimes used by less technical employees, which is the case with the mobile application used by truck drivers scenario. The application could be quickly constructed from a directions service, traffic feeds, and weather feeds by IT employees and used by truck drivers without the technical background.

Table 5.2- Potential Benefits of Mashups in Supply Chain

	Data Accessible to Many Parties	**Solve Problems with Real-Time Data Quickly**
How much Benefit?	Large benefit	Large benefit
Potential Problems	Limited number of parties providing data; data provided in different formats	Mashup platforms might only be useful for very technical users
Overall Conclusion	Benefit increases with more parties providing data	There should be multiple applications for using real-time data to improve shipping efficiency

Table 5.2, above, shows that there is reason to pursue mashups both for the data they make available to many parties and their use with solving problems with real-time data. For these reasons and the benefits listed for mashups in the first chapter, the costs presented by using mashups in this way should be looked into further to see if mashups are still worthwhile.

5.4 Costs of Mashups in the Supply Chain

There are going to be several costs when implementing mashups that have not been looked at in depth. These include the startup cost of infrastructure, maintenance costs, including mashup training, service repository maintenance, and data source verification.

Majority of the initial costs will be with constructing the infrastructure; there needs to be some software or other service directory that makes it easy to post services to be used by others and use services posted by others. The costs associated

with the infrastructure will either be paying for an existing, supported mashup platform or internally developing a mashup platform or some other means of publishing available services (such as SOA). If there is already a means for posting services such as SOA already in place, there is no cost at all to develop it. It should be noted that commercial products offer more advantage for a larger enterprise. The cost is often a set amount regardless of how many services are used. Using a commercial mashup platform also offers a standardized way of using mashups that would not be as common with internally developed ways. Lastly a commercial mashup platform's cost is going to be more predictable because the price for all the different features is already set out beforehand and there is usually support provided. The cost of platform setup could seem large but in terms of how much money could be saved by scenarios like the collaborative cargo management example, the cost would be very small.

The first maintenance cost that will be looked at is providing mashup training programs. The purpose of spending for training is to ensure that a system that is invested in is actually used. The training would likely need to be provided only for mashup developers. Mashups will typically be lightweight applications that are fairly easy to use such as the weather and traffic applications made for mobile devices described before, so the need to offer training for mashup users is not as great as for the developers. One advantage to using mashups is that it allows individuals at every level in an enterprise to see opportunities for new helpful services and create these from existing services. This can only happen if employees know about the services available and how to use them in a mashup environment to create a new service.

Providing both the awareness and the best practices of using mashups would make them much more successful. It should be noted that for smaller enterprises that do not plan on using mashups for more than one or two purposes such as the scenarios described in this chapter, the expenses for training would be lower than for large enterprises that have many uses for mashups. Training could be a significant ongoing expense, but the rewards for it outweigh the costs.

The reasoning behind service repository maintenance would come mostly from the "ad hoc" mentality with mashups. Because mashups would be made by a wide range of individuals, there could be duplicate services, services that are not ever used, and services that do not work the way they were intended. There would need to be an individual or group of individuals put in charge of maintaining the repository and keeping it from getting unnecessarily large. This maintenance could be done by periodically going through the available services and removing duplicate and unused services. Another way would be to have a strict set of guidelines before adding services to the repository. These simple solutions make service maintenance a problem that needs to be considered, but if handled proactively should not be that large of a cost.

Because enterprise mashups are built on being able to access data sources inside and outside of the enterprise, the situation is more complex. This means the data can be provided in many more types of data formats than if it was all internal data sources that could be standardized. Furthermore outside data sources cannot always be trusted in that they might provide inaccurate data. If all data sources have to be checked for validity it takes away most of the automation and the number of data

sources that can be used. Furthermore if the data sources have to be verified, this would be another maintenance cost. The collaborative cargo management scenario illustrates this point well. Verifying that the data necessary to collaborate with another enterprise is valid would be very helpful but not absolutely essential. If a malicious user were to successfully imitate a legitimate shipping enterprise, the data provided could cause the deceived enterprise to waste time. However security measures could be set up to ensure that the deceived enterprise does not provide confidential information to the wrong source. This will be discussed more in the security section, but for the purposes of maintenance, it is not absolutely essential other data sources be verified. For this reason the resources and overall cost used for data validation would likely be low.

Table 5.3- Mashup Cost Analysis

	Type of Cost	**Important Notes**
Mashup Platform Development	Initial	A necessary cost, but not that large compared to possible costs cut by mashups. Some enterprises already have a good mashup platform infrastructure in place such as those with SOA already in place
Mashup Training	Initial/Maintenance	Training is very important, so it is well worth the expense. Also training will typically be a higher cost, the larger the enterprise
Service Repository Maintenance	Maintenance	This cost should usually be low if repository maintenance is watched closely
Data Source Verification	Maintenance	This cost is fairly low because outside data source verification is not nearly as important as securing inside data sources which will be discussed more in the next section

In Table 5.3, above, the four main costs are presented along with if they will be an initial cost only or if they represent a recurring maintenance cost. The main part to pull from the table is that each cost can be low or high, but it would change the effectiveness of the mashups. Each individual scenario for mashups needs to be looked at to see where more money should be spent. If only internal services need to be used, then there is no reason to pay for data source verification. Such decisions need to be made for every situation mashups are used in.

5.5 Mashup Security in the Supply Chain

This section will set out to answer the question of 'Can mashups be implemented in a supply chain in a secure way?' Mashup security has been reviewed in a previous chapter as well as how some security principles have been applied to other problems. This section will review the vulnerabilities that would come up by using mashups in a supply chain setting and decide if they pose too much of a threat to the supply chain to make mashups worth using. The best way to do this is to present the main vulnerabilities while reviewing the two scenarios set out at the start of this chapter.

The first scenario was the application for mobile devices used by truck drivers. In this scenario, the application being used would only be providing mapping, routing, weather, and traffic data. None of these data sources would be considered sensitive information, so there would be no need to worry about leaking confidential data. There could be a slight worry about sending routing data over the mobile device, but if the application ran purely on the phone and only used communication to retrieve

data such as the weather and traffic, it could be used safely. However this scenario does point out the fact that there could be sensitive data that needs to be transferred to an application run from a mobile device. Assuming the mobile device is a cellular phone that uses one of the main carriers to send the data, this method would be secure enough for any services with information that would be talked about in a phone call or email. Data sent over cellular phones is not perfectly protected, but it would be as protected as any business calls or emails made over business phones, so the services should be limited to information that can be sent over this network.

The weather and traffic mashup scenario also simplifies the technology in that it would be data that all would be kept within a single enterprise. Even assuming that there is confidential data that needs to be transferred to the cell phone, there would only be need for password capabilities e.g., in the form of an API key or perhaps a password entered into the phone the first time it is used. These will be discussed in more depth shortly, but the important conclusion to draw from this example is that there really would not be any more security needed beyond a password or API key if only one enterprise is involved. It is when multiple enterprises share data that the situation gets more complicated.

The last possible security vulnerability shown by this scenario is the physical security of the cell phone itself. If that cell phone is built to access an application after a password has been entered the first time the phone is used. The solution of course is to use a password that has to be entered after every use. The tradeoff is that with the passwords entered a single time into the phones, the truck drivers did not even have to be trusted with the passwords. The passwords could have been entered

into phones before they were distributed to the truck drivers. If the password is entered with every use, then the trust is put in the truck drivers. Either the truck drivers or the physical security of the phones has to be trusted at some point. Either one of these vulnerabilities is a reasonable amount of risk for an enterprise to take on in the supply chain. At some point employees or their actions to keep their phones safe must be trusted for any business to work. It is prudent to know this is a vulnerability, but it does not make sense to not use mashups because of that small of a problem.

The second scenario was the collaborative cargo management scenario that included involvement from two or more enterprises and would likely just be a desktop application developed and used by logistics experts in charge of the shipping of products. This scenario has all of the same security vulnerabilities of the first problem, but also must deal with the security of having both authorization and authentication of who can use the data provided.

The authorization of using the data provided for the collaborative cargo management scenario refers to deciding on which other enterprises can access the data. The two ways of doing this is that either every enterprise needs to decide for themselves which other enterprises should be allowed to see their data or a trusted third party would have to act as an authority to help authorize what data can be seen by each enterprise. In either case, one of the security models referred to in the security chapter needs to be used to help separate enterprises with conflicts of interest. The Chinese Wall Model seemed to fit well with the supply chain industry because of the use of conflict of interest classes. Enterprises that would have a

conflict of interest by seeing each other's data could simply be put in the same conflict of interest class, so they would be protected from seeing each other's data.

If a trusted third party were used to keep the conflicts of interest separate, it would simplify it for each separate enterprise having to do this individually. The problem is that there needs to be justification for starting a trusted third party. It is likely that this would have to be done by each individual enterprise to start and a third party authority might be something that could come along in the future if collaborative cargo management ever got large enough.

Now that the authorization has been handled, the authentication needs to be considered. After each enterprise decides who should be able to see its data, it needs a way of authorizing that each source is who they say they are. This is where API keys come back into play. As discussed in the security chapter, API keys are similar to passwords in that they are coupled with a user name to provide authorization to a user. The difference is that they work in the context of APIs better than a password because they are designed to be put right into the URL when making an API call. If API keys were generated and carefully distributed to trusted parties, this would provide authorization of the API calls within a reasonable doubt. The enterprises that the API keys were distributed to would still need to be trusted, but that is going to be the case with any collaboration that is done in the industry.

Both scenarios have been reviewed for all security vulnerabilities that might be present and everything that has been considered has a solution that should be sufficient. While it is possible that there are some situations that have not been

reviewed would not be safe, the two scenarios that have been looked at and many similar scenarios can be implemented safely.

5.6 Openly Sharing Information in the Supply Chain

The last question that needs to be answered to decide mashups worth in the supply chain is 'Will enterprises share information openly?' This question is hard to answer because it is a question subjectively decided by each enterprise. The main reason for an enterprise to decide it is not worth sharing information openly is the security aspect. As was just reviewed in the previous section, the security of mashups should be sufficient not only to keep malicious users from seeing the information provided by the mashups but also to separate enterprises that have conflicts of interest. The safeguards put in place should be enough for many enterprises to openly share information. Furthermore with API keys data would not have to be put out for just anyone to see. Even an enterprise that needs its data very secure can maintain its own set of API keys and can thus limit the users of a service or data source to whoever that enterprise sees fit.

However there will always be some enterprises that do not trust any security measures at all. Sometimes the data that must be provided to make a mashup work will be too valuable to an enterprise to risk. Even in these cases there are still mashups that these enterprises can use such as the traffic and weather mashup that could be used by truckers without having to provide any confidential data at all. The enterprises that consider it too much of a risk to openly provide information in mashups would not be able to participate in mashups like the one in the collaborative

cargo management scenario. However this would not keep other enterprises from still using a collaborative cargo management mashup to cut costs. It does not take involvement from all parties for that scenario to work. In theory even two small enterprises could cut costs by providing information to each other. The scenario of course works better with more enterprises involved but it is not a requirement. For these reasons even if there are some enterprises that will not share information openly there is still plenty of value in using mashups in the supply chain industry.

5.7 Conclusion

This chapter discussed the background information by going over terminology that will be used and presenting two scenarios that mashups might be used in the supply chain. Then all of the questions that were presented at the beginning of this paper were discussed in depth and looked at in terms of the scenarios presented. The results of what was found by answering each of these questions by applying it to the two specific scenarios is what the next chapter will discuss.

Chapter 6

Conclusion

Many aspects of mashups, supply chains, and the security of each have been looked at to determine the worth of mashups in this industry. The five questions that needed to be answered to see if mashups were worthwhile in the supply chain industry were answered. These questions that were set out in the introduction were answered by reviewing if mashups would work in an enterprise setting, the advantages of using mashups, the costs of implementing mashups in the supply chain, the security implications of using mashups, and if enterprises would openly share information.

Before deciding if mashups had a place in the supply chain, it had to be determined if they even had a place in an enterprise setting. It was determined that presentation layer mashups offered some advantage and had little down side since they have so few security concerns. They offer little to no way to cut costs or optimize processes though; rather they offer a way to present data in easier to read formats. Situational mashups offer most of the advantage of mashups in the supply chain, but do have more security risks than presentation layer mashups, especially if

external data sources are used. Process mashups offer limited use in the supply chain because of the nature of mashups and because they offer the most security concerns of the different types of mashups. Mashups are not usually used to solve large process problems because of the standardization needed to do this. Despite this, there are still some processes that are important to the supply chain that mashups could help solve if enough parties were involved such as with the collaborative cargo management scenario. After reviewing the different types of mashups it was determined that there would be many uses for mashups in an enterprise setting and that they would work in an enterprise setting.

Next the advantages of mashups were reviewed and advantages specific to mashups in the supply chain were presented. Some of the general mashup benefits previously discussed were reuse of services, central repository to maintain and fix code, and use of existing systems such as SOA. The advantages to using mashups that are specific to the supply chain industry are allowing many parties to more easily share data and using real-time data to solve problems. Both of these advantages were discussed to be very beneficial and could play a large role in cutting costs in the supply chain. Mashups still appeared a very good candidate for the supply chain industry after reviewing the advantages they offer.

The costs were reviewed to decide if they would be too great for mashups to be worthwhile. Reviewing the costs brought up a number of questions that need to be answered by each enterprise separately before deciding on the best use of mashups. Because there is a wide range of situations where mashups can be used, the best way to set mashups up and what the associated costs will be should be decided for each

situation individually. The four costs that were looked at were the initial cost of infrastructure setup, level of training provided, maintenance of the service repository, and data source verification. As mentioned these costs would differ for each enterprise, but they were determined to be low enough that mashups could still add value overall.

The security of mashups in the supply chain was looked at next. Both scenarios laid out in this chapter were put under scrutiny and looked at for security vulnerabilities. There are several vulnerabilities that present themselves. Several of the techniques to deal with these vulnerabilities are ensuring that data is both authorized and authenticated if multiple parties are involved, properly protecting data sources with API keys, and using security access control models where useful. By using these techniques, the largest vulnerabilities should be accounted for and securely handled. After reviewing the security of using mashups in the supply chain, it was determined that they can be secure enough to still be successful.

The last question to answer was if enough enterprises would openly share information to make mashups worth pursuing. It was determined that the security should be good enough to help enterprises feel confident in making their data available to other enterprises. However there will always be some enterprises that decide it is not worth it to make their data available at all. It was demonstrated with the scenarios discussed in this chapter that there would be opportunities with mashups even if many enterprises decided against making data available for others to use.

After reviewing all the questions set out in the introduction of this paper, it was determined that mashups are worth pursuing in the supply chain industry. The benefits and costs should be reviewed before being used in any enterprise since they will still offer more benefit for some parts of the supply chain than others. Two scenarios were presented that offer considerable value to the supply chain industry. The first was using mashups to create an application for mobile devices to be used by truckers to help avoid traffic and weather. The second scenario was using mashups to help small and medium enterprises combine less than full containers to cut costs. It is difficult to define all the ways in which mashups will be used best because of the nature of mashups. Mashups can be used by any person at any level of an enterprise to solve a problem they find, so this means there is opportunity for even more uses of mashups in the supply chain industry.

References

[1] D. Braga, S. Ceri, D. Martinenghi, F. Daniel, “Mashing Up Search Services,” *IEEE Internet Computing*, vol. 12, no. 5, 2008. pp. 16-23. Print.

[2] Dreyfus, Daniel, David Fitzpatrick, Michael Onder, and Joanne Sedor. "The Electronic Freight Management Initiative White Paper." *Electronic Freight Management*. 01 04 2006. US Department of Transportation, Web. 13 Nov 2009. Print.

[3] N. Carrier, T. Deutsch, C. Gruber, M. Heid, L. Lucadamo, “The Business Case for Enterprise Mashups,” (August 2008). [Online]. Available: ftp://ftp.software.ibm.com/software/lotus/lotusweb/mashup/EPW14002-USEN-00.pdf (accessed June 21, 2009).

[4] Juttner, Uta. “Supply Chain Risk Management,” *The International Journal of Logistics Management*, vol. 16, no. 1, 2005. pp. 120-141. Print.

[5] European Conference of Ministers of Transport, *Container Transport Security Across Modes*. Paris, France: Organisation for Economic Co-operation and Development, 2005. Print.

[6] Warner, Janice. “Privacy Protection in Government Mashups,” *Information Polity*, vol 14, no. ½, 2009. pp 75-90. Print.

[7] Fichter, Darlene, and Jeff Wisniewski. "They Grow Up So Fast: Mashups in the Enterprise." *Online* 33, no. 3 (May 2009): pp. 54-57. *Academic Search Premier*, EBSCO*host* (accessed August 16, 2009).

[8] Hinchcliffe, Don. "Mashups: The Next Major New Software Development Model?", [Online]. Available: http://.blogs.zdnet.com/Hinchcliffe/?p=106

[9] Catone Josh. "Forrester: Enterprise Mashups to Hit $700 Million by 2013." *ReadWriteWeb* (2008) Web. 23 Oct 2009. <http://tinyurl.com/59a27f>.

[10] Stephen Farrell, "API Keys to the Kingdom," *IEEE Internet Computing*, vol. 13, no. 5, pp. 91-93, Sep./Oct. 2009, doi:10.1109/MIC.2009.100

[11] Obrenovic, Zeljko, and Dragan Gasevic. "Mashing Up Oil and Water: Combining Heterogenous Services for Diverse Users." *IEEE: Internet Computing*. 13.6 (2009): 56. Print.

[12] Kodali, Raghu. "What is Service-Oriented Architecture?" *What is Service-Oriented Architecture?*. 13 Jun 2005. Java World, Web. 7 Jan 2010. http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html.

[13] Stallings, William, and Lawrie Brown. *Computer Security: Principles and Practice*. 1st ed. Upper Saddle River, NJ: Prentice Hall, 2008. Print.

[14] *Internet Security Glossary: RFC 2828* [web page]. www.ietf.org, 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2828.txt

[15] http://www.whatisrss.com/

[16] http://www.xml.com/lpt/a/1619

[17] http://pipes.yahoo.com/pipes/

[18] http://techcrunch.com/2009/07/17/microsoft-popfly-gets-squashed/

[19] http://www.jackbe.com/

[20] http://code.google.com/gme/

[21] http://www.dapper.net/

[22] http://www.edissweb.com/

[23] Hammer-Lahev, E. "The OAuth 1.0 Protocol." *IETF.org*. Internet Engineering Task Force, 10 Feb 2010. Web. 20 April 2010.

[24] Blanchard, David. "Top Nine Supply Chain Challenges for 2009." *Industry Week* 01 Feb 2009: 42-44. Web. 02 May 2010.

[25] Hinchcliffe, Dion. "The top 10 challenges facing enterprise mashups." *Enterprise Web 2.0*. ZDNet, 16 Oct 2007. Web. 24 May 2010.

[26] Cooper, Lane F. "Strategic Value of Enterprise Mashups." *JackBe*. BizTechReports.com, 2010. Web. 25 May 2010.

[27] Bloomberg, Jason. "Enterprise Mashups That Both Leverage and Justify SOA." *JackBe*. Zapthink, 09 May 2008. Web. 28 Jun 2010.

[28] Crupi, John. "A Business Guide to Enterprise Mashups." *JackBe*. JackBe.com, Apr 2008. Web. 28 Jun 2010.

[29] Benson, Jim, and Dion Hinchcliffe. "EMML Changes Everything: Profitability, Predictability & Performance through Enterprise Mashups." *JackBe*. hinchcliffeandcompany.com, 11 Dec 2009. Web. 30 Jun 2010.

[30] Web. 9 Jul 2010. <http://www.freefoto.com>.

[31] Web. 12 Jul 2010. < http://google.com/maps>

[32] Web. 12 Jul 2010. http://mapquest.com

[33] Cinzia Cappiello, Florian Daniel, Maristella Matera, Cesare Pautasso,

"Information Quality in Mashups," IEEE Internet Computing, vol. 14, no. 4, pp. 14-22, July/August, 2010. 11 Aug 2010.