

SaaS INTEGRATION





TABLE OF CONTENTS

> 1.0 Introduction	1
> 2.0 Why Integrate?	2
> 3.0 Integration Examples	2
3.1 Personnel	2
3.2 Customer Relationship Management	3
3.3 Financial	3
> 4.0 Standards	3
4.1 Web Services	3
4.2 Integration Servers	4
> 5.0 Integration Issues	4
5.1 Security	4
5.2 Technical Resources	5
5.3 Infrastructure	5
5.4 Impedance Mismatch	5
5.5 Process Integration and Workflows	6
> 6.0 Implications for SaaS Architecture	6
> 7.0 Summary	7
> 8.0 References	7
> 9.0 Glossary of Terms	8

1.0 INTRODUCTION

This paper is one of a series of papers that explore and discuss the technical architectural components of the Software-as-a-Service (SaaS) model.

SaaS shares the distinction of being both a business model and an application delivery model. SaaS enables customers to utilize an application on a pay-as-you-go basis and eliminates the need to install and run the application on the customer's own hardware. Customers generally access the application via a Web browser or thin client over the Internet. SaaS is most often subscription based and all ongoing support, maintenance, and upgrades are provided by the software vendor as part of the service. Application customization capabilities, if available at all, are generally provided to all customers in a consistent manner. From the perspective of the software vendor, the SaaS model provides stronger protection of its intellectual property, operational control of the environment running the software, and generally a repeatable revenue stream from the service subscription fees. Software vendors have varying capabilities and applications come in varying flavors, but SaaS applications most typically support many unique customers using a single instance of that application — also known as multi-tenancy.

As pointed out in the Saugatuck Technology Four Waves Evolution model¹, the second wave that we find ourselves in today sees the emergence of integration as a critical concept. SaaS is maturing and gaining more widespread acceptance. As a result, SaaS solutions are naturally becoming key components of the overall corporate software stack and these components often need to be aware of, or integrate with, each other.

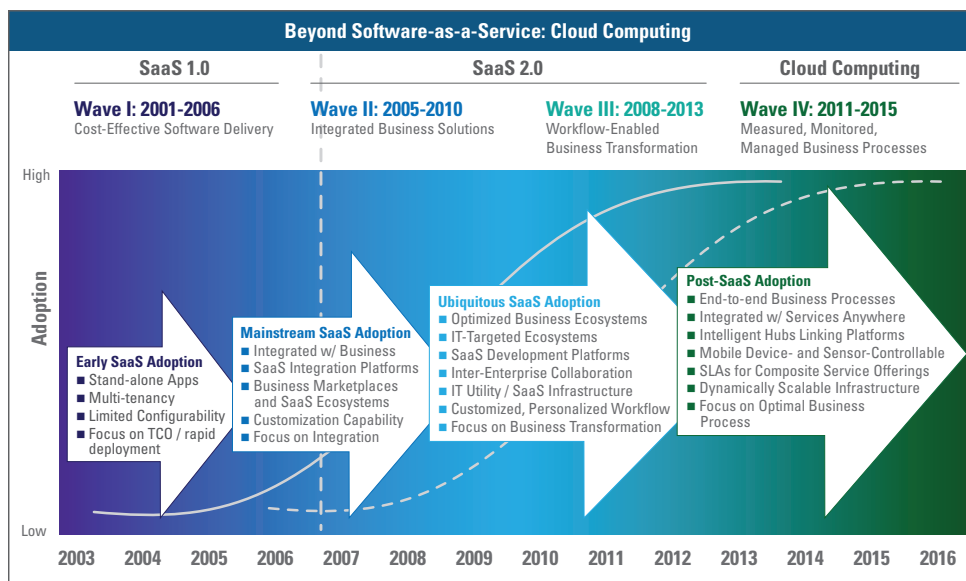



Figure 1 — Saugatuck SaaS Wave Evolution Model

Source: Saugatuck Technology Inc.



This paper will cover the topic of integration as it applies to SaaS architecture. We will look at why there is a need to integrate in the first place and then explore some examples of the types of systems that are top integration candidates. We will then discuss some emerging integration standards and some of the issues that will need to be addressed in implementing these solutions. Finally, we will focus on what implications integration will have for SaaS architecture.

2.0 WHY INTEGRATE?

The answer to this question is relatively simple. Integration saves resources (time, money, labor, etc.) and helps ensure data integrity and accuracy. Data in one corporate system is often needed in other systems and rather than manually re-entering or periodically dumping data from one system into another, it is simply more efficient to implement an integration layer that will keep data synchronized automatically at all times.

Another way to think about integration is to think of one application as a component in a larger flow of work. As a component, this system will take inputs from one or more other systems, do its own processing and then pass the results on to yet another system. The advantage of integration is reaped when as much of this interoperability is automated as possible.

3.0 INTEGRATION EXAMPLES

There are many examples of systems or data flows that would be excellent candidates for integration. We will now touch on a few that are present in practically every business and therefore will provide a basis for thinking about integration for many readers.

3.1 Personnel

People are essential to virtually all businesses and personnel are generally tracked in some sort of human resources system. There are even companies that rely on their network directory services (i.e., Microsoft Active Directory) or payroll service (e.g., ADP) as the source system that keeps track of personnel. In some instances, multiple systems from the above choices are needed to get a complete, consolidated view.

Any SaaS solution that requires knowledge of employees is a candidate for an integration layer to synchronize this data from the source systems. This data might include name, contact information, department, location, user ID, password and authorization information. Entering such data and keeping it up-to-date manually can be a monumental task, especially if the company has a large number of employees, so investments of time to integrate will generate returns in time and cost savings very quickly.



3.2 Customer Relationship Management

Customer Relationship management (CRM) systems are a class of software that helps automate the sales process, from marketing to prospects all the way through closing deals and offering additional services to existing customers. A great example for SaaS solutions is Salesforce.com.

Regardless of which CRM solution is used by a customer, there is a need to potentially integrate data from the CRM application to another application, either on-premises or SaaS. This may include financial sales data that integrates with a financial accounting package or order information that flows through to an enterprise resource planning (ERP) system for fulfillment, but there are many other examples where data may need to flow to or flow from a CRM application.

3.3 Financial


As touched upon in the CRM example, there are many situations where the need to integrate a SaaS solution with a financial accounting application may arise. The general ledger usually contains information regarding the corporate departmental structure as departments have unique account numbers to facilitate the tracking of revenues and expenses, and it may be the definitive source system for this type of data. There is a need to track all financial transactions in the accounting system, so any SaaS application that involves these types of transactions is a potential candidate for this type of integration.

4.0 STANDARDS

There is an entire class of technology involving integration that is lumped under the term EAI (enterprise application integration). For as long as applications have been able to communicate across a network this technology has continued to evolve and mature. Standards for this type of communication have evolved also. We will now explore some current trends.

4.1 Web Services

The emergence of XML (extensible markup language) as a means for encapsulating data so it can be processed in an automated fashion has led to nearly ubiquitous adoption of Web services as a standard mechanism for applications to interact across the Internet. By utilizing a protocol called SOAP³, applications using Web services can exchange XML data via HTTP.



Just as XML provides for an XSD (XML Schema Definition) file to define the data model, Web services provides for a WSDL (Web Services Description Language) file to expose the service interfaces. Developers can simply import the WSDL into their Progress OpenEdge® development environment and samples of stubs are created to make all the calls necessary to use the Web service.

4.2 Integration Servers

While Web services have become relatively common, they still require developers to write programs to utilize the services. In an effort to further simplify this process, a class of applications is emerging which we will refer to as integration servers. Leveraging paradigms from the old EAI space, these applications expose interfaces for both source and target systems, data is visually mapped from one to the other, and any necessary data transformations are defined. The servers can then run these data flows either by a schedule or via a triggering call from another system. While these types of applications do help simplify integration, they still require a technical resource to properly configure and maintain them.


5.0 INTEGRATION ISSUES

Achieving integration is a worthy goal but there are numerous issues involved with actual implementation. It is important to consider these issues if a SaaS vendor is going to effectively provide powerful and flexible integration solutions to its customers.

5.1 Security

The reason security is an issue is that integration often involves access to data within the customer's environment and getting that data from their environment to the SaaS environment. One aspect of this has to do with whether the data is pulled or pushed. If the SaaS application pulls data from the customer, it is making an unsolicited call into the customer's environment and this is often a problem for the corporate security officer. If the data is pushed from the corporate environment to the SaaS vendor, then some type of application needs to be running in their environment and sometimes the corporate security office has a problem with this. Not only that, but there needs to be a server to run this application on and therefore there is the expense for hardware as well as managing, monitoring and maintaining that hardware.

Another aspect of security that can impede SaaS integration efforts involves getting access to the internal corporate data. The source systems that hold this data have people that are responsible for those systems and the data. Just getting permission to get at the data can often become an exercise in navigating corporate bureaucracy. Once permission is granted, it can also be a challenge to access the data at an ideal point in the process.



For example, if you need to synchronize data from a corporate HR system it is ideal to inject a call to the SaaS platform right at the point the data is added, deleted or modified in the source system. More often than not this is unfeasible because either the source system does not provide this flexibility or the keepers of that system just do not have the expertise to program the proper API calls.

5.2 Technical Resources

Another issue that impedes integration with a SaaS application is that technical resources are needed to implement. Traditional corporate IT departments focus on developing their own solutions or maintaining on-premises software that has been purchased. With SaaS the solution is often brought into the corporation by the business users that require the functionality, many times out of frustration in not getting what they need from their own IT group. The IT department ends up being brought into the conversation at a late stage when it becomes difficult to allocate technical resources.


Because of this all-too-common dynamic, it is useful for the SaaS provider to have their own staff available to assist with technical implementations as consulting services. When these services are staffed, priced and sold in a methodical manner it can make the integration process smoother for both the vendor and the customer. The point here is that SaaS integration requires some technical work to implement and the people needed to do this work need to either come from the customer, the SaaS vendor or a third-party integrator. The SaaS vendor can facilitate the process by at least having the last two options readily available.

5.3 Infrastructure

A third issue that can make integration difficult is the need for infrastructure components to make it work. Whether servers are needed to run software or firewalls need to be configured to allow appropriate access to data, there is some impact on the customer's corporate infrastructure. If the SaaS vendor clearly understands this impact and offers flexible options for implementation, it will help to make integration easier to implement for both the customer and the vendor.

5.4 Impedance Mismatch

The concept of impedance mismatch⁴ may need to be addressed when dealing with integration between systems. Impedance mismatch refers to the inherent difference between object-oriented software technology and relational database technology. Both of these technologies are in use in mainstream application development and there is often a challenge involved with mapping data from one to the other. In broad terms, there tend to be two types of mismatches that are encountered: different data types and different data storage and manipulation approaches.



SaaS vendors should consider these possibilities for impedance mismatch when designing the various approaches they will provide customers to use when integrating other systems with the SaaS application. By making sure that each approach clearly and unambiguously allows customers to map their existing data to the data stored in the SaaS application and vice versa, the SaaS vendor will help to avoid problems with impedance mismatches.


5.5 Process Integration and Workflows

As SaaS applications mature through the SaaS Four Waves Evolution Model and SaaS solutions become critical components of overall corporate processes, it becomes more important for the SaaS solution to provide integration options that are easy to use and implement. Application vendors can easily fall into the trap of thinking about their solutions only in terms of the functionality they provide. It is useful for the vendor to also think about their solution as a simple “black box” that is only part of a larger overall process. This will allow for careful consideration of ways to allow customers to easily get data into the application that is produced by prior steps in the process (i.e., other source applications) and then pass that data on to other systems that might handle later steps in the process.

The concept of workflows can be applicable here. Workflows identify who participates in the steps of a business process, what needs to be done in each step and when this needs to be done. There are a number of emerging technical standards around workflows, including Wf-XML (Wf-XML and Workflow Reference Model from the Workflow Management Coalition) and BPEL4WS (Business Process Execution Language for Web Services)⁵, and the SaaS vendor will need to determine for itself which standards approaches will best serve its customers.

6.0 IMPLICATIONS FOR SaaS ARCHITECTURE

Now that we have looked at some of the issues involved with successfully implementing integration, it may be helpful to consider the implications these may have for the vendor's SaaS architecture. It will be ideal to offer multiple solutions in order to provide flexibility. A good place to start is in implementing a Web services API that exposes the application's data to automated interaction. To be complete this might include the ability to add, delete, modify and even select data for outside reporting or to pass on to another system as part of a larger process.



Adding an integration server option then provides another choice for the customer. This may need to be offered in two flavors due to some of the security issues covered. A hosted option would enable the customer to avoid procuring the resources needed to host it internally. On the other hand, this option may raise the security issue of pulling data out from the corporate environment. An option that can be installed on-premises with professional services to assist if needed would round out the offering in a way that would support the customer as much as possible.

7.0 SUMMARY

Integrating a SaaS application with other applications is becoming a critical topic as the SaaS model matures and gains more widespread acceptance. This integration can involve source systems running within a corporate environment or other SaaS solutions. There are many different types of systems that can be integrated and we covered some of the standard ones along with some of the emerging integration standards. Implementing an integration solution can be impeded by various issues and some of these were presented for consideration. Finally, we discussed the implications for a SaaS architecture so the vendor can think about how to best serve their unique customer base.

8.0 REFERENCES

¹M. West, B. Guptill, *Saugatuck Technologies Strategic Perspectives STR-458*, "Saugatuck SaaS Research: Waves and Platforms in the Cloud," April 29, 2008.

²Andreeson, Marc, "The three kinds of platforms you meet on the Internet," September 16, 2007 — <http://blog.pmarca.com/2007/09/the-three-kinds.html>. In this excellent post Marc outlines three levels of platform, the third of which involves applications that run inside the platform. This is now referred to as PaaS, or Platform as a Service.

³SOAP used to be an acronym for Simple Object Access Protocol, but that seems to have been dropped with the release of Version 1.2 and it is now just called SOAP. The W3C maintains the SOAP standards — see <http://www.w3.org/TR/soap/> for details.

⁴Ambler, Scott, "The Object-Relational Impedance Mismatch," February 15, 2006 — <http://www.agiledata.org/essays/impedanceMismatch.html>.

⁵Virdell, Margie, "Business processes and workflow in the Web services world," January 1, 2003 — <http://www.ibm.com/developerworks/webservices/library/ws-work.html>.

9.0 GLOSSARY OF TERMS

Term	Description
API	API stands for Application Programming Interface. This generally refers to a series of method or procedure calls along with a specific communications protocol to programmatically interact with an application.
Application Framework	An Application Framework is a software framework designed to aid in the development of dynamic Web applications.
Application Server	An Application Server is the software engine that delivers dynamic application functionality to a client computer or device generally via the Internet and generally using the HTTP protocol.
SaaS Offering	A SaaS Offering in this document is defined as a turnkey service offering, which includes the application license, maintenance, application support, subscription pricing, and hosting and associated delivery infrastructure.
Customer/Tenant	Customer/Tenant is a term in this document that refers to the company or business that subscribes to the SaaS offering.
HTTP	Hypertext Transfer Protocol is a communications protocol used to transfer data on the Web.
PaaS	PaaS stands for Platform as a Service and encompasses a SaaS development and hosting environment in which the applications can not only be written, but also run and provided to customers as services as well.
SaaS Offering	A SaaS Offering in this document is defined as a turnkey service offering, which includes the application license, maintenance, application support, subscription pricing, and hosting and associated delivery infrastructure.
SaaS-Enabled Application	A SaaS-Enabled Application is defined as an application that has been designed and built to be deployed and consumed in a service model, and incorporates many of the attributes previously defined above.
SOAP	The SOAP protocol provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment.

Term	Description
Users	User in this document is defined as an employee of the customer, as is the ultimate user of the service.
Web Server	A Web Server generally delivers static content via HTTP.
Web Service	A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.
XML	XML stands for eXtensible Markup Language. It is a subset of Standard Generalized Markup Language (SGML) of which HTML is also a subset. The power of XML is that it is self referencing with the inclusion of an XML schema in the form of an XSD file (XML Schema Definition). So, with the XML file containing marked up data and the XSD file defining this markup and the associated data types, a program has the information necessary to process the data.



Worldwide Headquarters

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA
Tel: +1 781 280-4000 Fax: +1 781 280-4095
www.progress.com

For regional international office locations and contact information, please refer to www.progress.com/worldwide

© Copyright 2008 Progress Software Corporation. All rights reserved. Progress is a registered trademark of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. or other countries. Any other trademarks contained herein are the property of their respective owners.