

# Deploying Software as a Service (SaaS)

Daniil Fishteyn, *Chief Technology Officer*

WebApps, Inc. a.k.a. SaaS.com

## Challenges of Implementing SaaS

### *Multi-tenant Deployment*

A multi-tenant platform is one that uses common resources and a single instance of both the object code of an application as well as the underlying database to support multiple customers simultaneously. Although the technology stack (servers, switches, bandwidth, data storage, etc.) in a multi-tenant deployment are shared and a single instance of code exists, customers' user experience should be similar to that of a user whose application was dedicated on an individual basis, much like that of an on-premise application. Multi-tenant deployment of an application optimizes the use of a limited set of computing resources across a large number of customers.

Multi-tenant deployment can be seen in current Web 2.0 deployments, in which applications aim to facilitate collaboration and sharing between users. This perceived second generation of web-based communities and hosted services, such as social networking sites, utilize centralized application access, and are therefore able to enhance applications based on real usage statistics, in many cases easier than the on-premise model. Gathering usage statistics and understanding user behavior is anonymized, so the identity of the data source is kept confidential in both consumer and enterprise-class applications. However, this delivery model poses a formidable challenge to delivering reliable and secure application performance, in an isolated yet logically independent, customer interaction.<sup>1</sup>

Since few standards have been established for multi-tenant application delivery or the operational governance to ensure isolation among customers, questions may surface regarding the suitability of the SaaS model for mission-critical applications, or those applications that collect, process, and store sensitive enterprise data. As a result, some customers still choose an isolated tenancy, or on-premise application, to ensure complete isolation. This is a common problem for most innovations coming to the mainstream, as they are guilty of immaturity until proven otherwise.<sup>2</sup>

Several solutions to the issues associated with multi-tenant deployment exist, namely having separate databases per customer, a shared database but separate schemas (set of database tables), or a shared database and shared schemas. A separate database for each customer is the most traditional approach. Although providing each customer their own database simplifies meeting customers' individual concerns for isolation, server (and other infrastructure) provisioning as well as installing multiple instances of a database can become time consuming and difficult to manage. This approach also tends to lead to higher costs, which makes it less favorable.<sup>3</sup> Like

---

<sup>1</sup> Gartner, Section 3, <http://mediaproducts.gartner.com/reprints/oracle/150447.html>

<sup>2</sup> Gartner, Section 3, <http://mediaproducts.gartner.com/reprints/oracle/150447.html>

<sup>3</sup> Multi-Tenant Data Architecture, [http://msdn2.microsoft.com/en-us/library/aa479086.aspx#mltntda\\_topic2](http://msdn2.microsoft.com/en-us/library/aa479086.aspx#mltntda_topic2)

the isolated or on-premise approach, the separate schema approach requires more resources to install, configure, and maintain, but differs in that it offers a moderate degree of logical data isolation and can support a larger number of customers per database server. However, the approach that tends to be most favored under SaaS deployment involves using the same database as well as the same schema to store multiple customers' data. When proper design and implementation is achieved, this approach results in the highest efficiencies and has the lowest server and related infrastructure costs because it allows companies to serve the largest number of customers per database server.<sup>4</sup>

### *Scalability*

An application's ability to service user requests in a graceful manner without giving way to lagging response times, while remaining easy to upgrade is many times referred to its scalability. Typically, scalability is the capability of an application to increase total throughput under an increased load when resources (typically more servers) are added.<sup>5</sup> For example, an application might be considered scalable if it could be moved from a limited set of servers to a larger configuration of more robust servers, while taking full advantage of the additional server resources and/or processing power in terms of performance (quicker response times) and handling a larger number of simultaneous customers. Typically, it is easier to have scalability upward rather than downward since developers must often use available resources efficiently when an application is initially coded. Therefore, scaling an application downward may mean trying to achieve the same results in a more constrained environment.<sup>6</sup>

Given SaaS applications are delivered via the Internet, the major challenges that apply to scalability are performance and load management. This aspect of deployment can be affected by many complex factors, including the design of an application's architecture, database schema, network connectivity, available bandwidth, back office services (such as mail, proxy, and security services), and other server resources. Depending on the size and complexity of an application, it may be able to handle anywhere from a handful to tens of thousands of simultaneous customers. Another contributing factor, the number of simultaneous connections made either by a customer through the Internet or through web services will have a direct impact on an application's performance. Therefore, performance objectives must include two scopes: the speed of a single customer's transaction and the amount of performance degradation related to the increasing number of simultaneous connections.

Load management refers to the method by which simultaneous customer requests are distributed and balanced among multiple servers. Effectively balancing loads across servers ensures that they do not become overloaded and eventually unavailable.<sup>7</sup> Load balancing allows an application to scale out across a group of servers, making it easy to add capacity by adding more replicated servers. It also provides redundancy, giving the site failover, or backup capabilities, so

---

<sup>4</sup> Multi-Tenant Data Architecture, [http://msdn2.microsoft.com/en-us/library/aa479086.aspx#mltntda\\_topic2](http://msdn2.microsoft.com/en-us/library/aa479086.aspx#mltntda_topic2)

<sup>5</sup> Wikipedia, <http://en.wikipedia.org/wiki/Scalability>

<sup>6</sup> Search Data Center, [http://searchdatacenter.techtarget.com/sDefinition/0,,sid80\\_gci212940,00.html](http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci212940,00.html)

<sup>7</sup> [http://www.adobe.com/livedocs/coldfusion/5.0/Advanced\\_ColdFusion\\_Administration/overview2.htm](http://www.adobe.com/livedocs/coldfusion/5.0/Advanced_ColdFusion_Administration/overview2.htm)

the application remains available to customers even if one or more servers (sometimes referred to as clustering as further described below) fail or need to be taken down for maintenance.<sup>8</sup>

A primary method to achieve scalability is horizontal scaling. This method involves using hardware (such as load balancers) to distribute service requests across multiple servers. Horizontal scaling occurs by using many machines all of which essentially function together as a one, which is also known as clustering. By dedicating several machines to a common task, application fault tolerance is increased. (Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components.)<sup>9</sup> Horizontal scaling provides a method of scalability that is not hindered by server limitations, such as overloading. The key to successful horizontal scaling is location transparency. Location transparency refers to making all business components or services seem as if they reside within the same server or programming space. If any of the application code depends on knowing what server is running the code, location transparency has not been achieved and horizontal scaling will be difficult. This situation is called location affinity. Location affinity requires code changes to horizontally scale an application from one server to many, which can be costly. Thus, strong consideration should be given to location transparency when designing an application.<sup>10</sup>

### *Reliability*

Informally, reliability is the level of accuracy in which an application provides its intended services, usually dictated by user documentation or application specifications. Reliability is about providing correct results and handling error detection and recovery in order to avoid failures. More formally, the mean time between failures (MTBF), that is, the average length of time the application runs until a failure occurs, defines reliability.

Reliable applications are increasingly critical to customers. Because failure of an application can result in lost data (as well as the possibility of lost business, data for analytics, etc.) and considerable recovery costs, companies are requesting 24 X 7 reliability for SaaS applications in terms of a service level agreement (SLA), commonly know as four 9s or 99.99% availability. An SLA is a formally negotiated agreement between two parties which records the common understanding about services, priorities, responsibilities, guarantees, etc. For example, an SLA may specify the levels of availability, serviceability, performance, operation, or other attributes of an application.<sup>11</sup> The Internet has made information and immediate, error-free access to this information, a norm to which customers have become accustomed, making reliability so much more important. Reliability of an application as a whole depends on the reliability of the individual components. Therefore, due to the fact that some components in a system may be related, a failure in one component can affect the reliability of others.<sup>12</sup>

---

<sup>8</sup> [http://msdn2.microsoft.com/en-us/library/aa292203\(VS.71\).aspx](http://msdn2.microsoft.com/en-us/library/aa292203(VS.71).aspx)

<sup>9</sup> Wikipedia, [http://en.wikipedia.org/wiki/Fault-tolerant\\_system](http://en.wikipedia.org/wiki/Fault-tolerant_system)

<sup>10</sup> [http://msdn2.microsoft.com/en-us/library/aa292203\(VS.71\).aspx](http://msdn2.microsoft.com/en-us/library/aa292203(VS.71).aspx)

<sup>11</sup> Wikipedia, [http://en.wikipedia.org/wiki/Service\\_Level\\_Agreement](http://en.wikipedia.org/wiki/Service_Level_Agreement)

<sup>12</sup> [http://msdn2.microsoft.com/en-us/library/aa292168\(VS.71\).aspx](http://msdn2.microsoft.com/en-us/library/aa292168(VS.71).aspx)

In order to ensure reliability, code review generally takes place before any major application testing (both user interface and background services) begins. Code review can improve the overall quality of an application by examining source code to find and fix mistakes overlooked in the initial development phases. Testing the application through regression, unit, functional, integration and acceptance testing are the best ways to determine its reliability. Regression testing looks for regression bugs, or errors, which prevent the application from behaving as intended. Regression bugs occur as an unintended consequence of program changes, usually to functions that existed prior to a software release or update. Unit testing is used to validate that individual units of source code, or the smallest testable part of an application, are working in the most efficient and error-free manner possible. Functional testing measures the quality of the business components of the system, or underlying code which is created to execute specific functions. The test determines if each business component responds correctly to all conditions that may be presented through inputting of data, workflow associated with data moving correctly from one business component to the next, and that business events are initiated in the order required to meet the business objectives of an application.<sup>13</sup> Integration testing is used to verify functional, performance, and reliability requirements placed on the application as a result of passing data from one system to another through some form of common communication protocol. Integration testing can expose problems with the interfaces among application components before errors occur in live application execution.<sup>14</sup> Acceptance testing is the last step in the process and allows customers to ensure that the system meets their business requirements – business logic, application-specific workflow, and screen behavior as anticipated.

It is important to have benchmarks in place to measure application reliability. These benchmarks can be measured through regression, unit, functionality, integration, and acceptance testing. Such benchmarks include identifying: faults through severity ratings, the number of continuous hours an application operates without failure, the mean time between failures (MTBF), and reasonable amount of time it takes for an application to recover smoothly.<sup>15</sup>

## *Usability*

The trend in application development is migrating towards a more dynamic user experience. A user interface is the means by which with an application, generally providing a means for input as well as output. Technology has significantly shifted from traditional, static applications that work independently to Web 2.0 applications, which require the capability to collaborate between multiple users and/or applications. Gartner reports that by 2012, consumer technologies will be fully integrated into all settings, including the office, home, remote office and recreational areas.<sup>16</sup> This consumerization has driven the development of the user interface to increase efficiencies.

---

<sup>13</sup> <http://www.devbistro.com/articles/Testing/Requirements-Based-Functional-Testing>

<sup>14</sup> [http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92\\_gci1243430,00.html](http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92_gci1243430,00.html)

<sup>15</sup> <http://blogs.ittoolbox.com/eai/implementation/archives/establishing-software-reliability-objectives-13464>

<sup>16</sup> <http://www.internetnews.com/dev-news/article.php/3636806>

Many SaaS providers are leveraging Asynchronous JavaScript and XML (Ajax), a current development technique, to improve overall user experience. Ajax is a method of building interactive applications for the Web that process user requests immediately. Combining several programming tools, Ajax allows content on Web pages to update immediately when a user performs an action, unlike other development approaches which users must wait for a whole new page to load. Screens refresh more quickly because only partial data is being sent back and forth, and only certain widgets are being refreshed. For example, a weather forecasting site could display local conditions on one side of the page without delay after a user types in a zip code.<sup>17</sup>

By incorporating Ajax, a user interface becomes more efficient, dynamic, and improves response time. In many implementations it allows users to have a better concept of the screen they are viewing and know where they are in a system because the background can remain visible. Ajax can also help reduce cost in software delivery as smaller packets of data are sent back and forth, ultimately using less bandwidth and processing resources.

The development of the user interface has also accommodated the mobile and flex workforce, whose third screen often takes over more tasks traditionally done on a users desktop. The third screen refers to a video screen, particularly the screen on a cell phone or personal handheld device that a person uses almost as often as their television or computer screens. When designing an application, developers need to consider who will be accessing the application and from what device. The third screen specifically relates to SaaS because, as a model that parallels to peoples' software usage being available anywhere at any time, users should have the ability to access from mobile devices.

### *Data Security*

Data security is the means of ensuring that data is guarded from corruption and that access to data is controlled. Thus, data security helps ensure the privacy and protection of sensitive information.<sup>18</sup>

The very nature of SaaS poses security challenges. In order to detect and prevent intrusion, strong encryption, authentication, and auditing must be a part of the application design to restrict access to private and confidential data. Liability for the cost and damages associated with any data breach typically rests with the independent software vendor (ISV) and / or managed services provider, individually or collectively referred to throughout this whitepaper as the SaaS provider.<sup>19</sup> Therefore, it is imperative for SaaS providers to make certain the SaaS applications they are providing their customers adhere to security policies outlined in any related SLA.

Encrypting sensitive data to ensure privacy in such a public space as the Internet should remain a top priority when designing a SaaS application. To encrypt data, information must be encoded in such a way that only the customer or computer with a key can decode it. Two different types of

---

<sup>17</sup> [http://searchwindevelopment.techtarget.com/sDefinition/0,,sid8\\_gci1107521,00.html](http://searchwindevelopment.techtarget.com/sDefinition/0,,sid8_gci1107521,00.html)

<sup>18</sup> Wikipedia, [http://en.wikipedia.org/wiki/Data\\_security](http://en.wikipedia.org/wiki/Data_security)

<sup>19</sup> [http://www.news.com/8301-10784\\_3-9726592-7.html?tag=more](http://www.news.com/8301-10784_3-9726592-7.html?tag=more)

encryption exist: symmetric-key and public-key. In symmetric-key encryption, each computer uses an undisclosed key or code to encrypt information before it is sent over the Internet to another computer. This requires the two computers that will be communicating to have the necessary key to decrypt the information passed to it. Public-key encryption uses a combination of a private-key and a public-key. The private-key is known only to the computer encrypting the information while the public-key is given to any computer that requires (and is granted) secure communication with it. A popular implementation of public-key encryption is the Secure Sockets Layer (SSL). SSL is an Internet security protocol used by Internet browsers and servers to transmit sensitive information. SSL has become part of an overall security protocol known as Transport Layer Security (TLS).<sup>20</sup> Two examples of TLS are Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) and File Transfer Protocol (FTP), both of which encrypt data before transferring it over a secure connection.

Authentication tools provide the ability to determine the identity of a customer attempting to access an application. A typical method of authentication is a username and password. Authentication is rarely used alone; in fact, it is the basis for authorization and privacy, which can be respectfully defined as the determination of whether access to an application will be granted to a particular customer and the act of keeping information from becoming known to unauthorized customers.<sup>21</sup> Authentication on the administrator and developer side is also imperative. When an administrator or developer needs access to object code they would typically validate their identity through token authentication. Tokens are a device characteristically small enough to be carried in a pocket or purse, often designed to attach to a keychain. The purpose of the token is to generate a unique code through an algorithm every “x” number of seconds. (RSA is the chosen security partner of more than 90 percent of Fortune 500 companies.<sup>22</sup>) The number being displayed at that moment on the secure token is used in conjunction with a personal identification number (pin) and password. Because the token changes every “x” number of seconds and must be used in combination with a pin and password, this approach makes data access hacker-resistant.<sup>23</sup> This approach can also protect intellectual property (IP) of a SaaS provider in that source, obfuscated, and object code is better protected from misuse.

Besides the transmission and retrieval of data, the storage of data is also a pressing security concern. Outsourcing data storage to a managed services provider (a company that provides delivery and management of network-based services, applications, and equipment to enterprises, residences, or other service providers)<sup>24</sup> is seen by many industry experts as the better of three primary alternatives to ensure data security. Managed services providers are able to focus on proper technical maintenance of servers and the remainder of the infrastructure on which a SaaS application resides, including redundant connectivity to the Internet. In addition, most of the leading managed services providers are required to meet auditing standards such as SysTrust

---

<sup>20</sup> <http://computer.howstuffworks.com/encryption.htm>

<sup>21</sup> <http://www.objs.com/survey/authent.htm>

<sup>22</sup> <http://www.rsa.com/node.aspx?id=1003>

<sup>23</sup> <http://www.rsa.com/node.aspx?id=1158>

<sup>24</sup> [http://searchitchannel.techtarget.com/sDefinition/0,,sid96\\_gci522393,00.html](http://searchitchannel.techtarget.com/sDefinition/0,,sid96_gci522393,00.html)

and Statement of Auditing Standards No. 70 (SAS70) (further described in the Auditing section below).

### *Auditing*

Audits consist of two types, namely security and information. The first type, a security audit, is when a third party validates a managed services provider's security profile. This is similar to an accounting firm reviewing a company's books. The second type, an information audit, refers to a subsystem that monitors actions to, from, and within an application. For example, an information audit may keep a record of all of the customers and associated users that log onto an application. Such a record is known as an audit trail.<sup>25</sup>

The need to track and log customer activity and transaction flows becomes all the more important, especially since the ratification of the Sarbanes-Oxley Act of 2002 (SOX). Virtually every transaction needs to be logged and/or have a sub-journal created so that after-the-fact analysis of customer activity can be accomplished.<sup>26</sup> Thorough and well-known audits such as SysTrust and SAS 70 – more particularly Type II, measure compliance to high standards of data security, and help identify whether security policies are operating efficiently over a period.

SAS 70 was developed and is carried out by The American Institute of Certified Public Accountants (AICPA), while SysTrust was jointly developed and carried out by (AICPA) in conjunction with the Canadian Institute of Chartered Accountants (CICA). SysTrust uses the framework of the Principles of Trust Services and their Criteria when evaluating companies such as managed services providers. In a SysTrust audit, the auditor evaluates whether a particular managed services provider is reliable when compared against the principles set for security, availability, processing integrity, and confidentiality of sensitive information which may be set forth in the form of an SLA between the managed services provider and its customers. Using these Principles and Criteria the following are determined:

- **Security:** whether the system is protected against both physical and logical unauthorized access.
- **Availability:** whether the system is available for operation and use as committed or agreed.
- **Processing Integrity:** whether processing is complete, accurate, timely, and authorized.
- **Confidentiality:** whether business information designated as confidential is protected as committed or agreed.

---

<sup>25</sup> <http://linux.about.com/cs/linux101/g/audit.htm>

<sup>26</sup> [http://www.aspnnews.com/strategies/article.php/11223\\_3686131\\_2](http://www.aspnnews.com/strategies/article.php/11223_3686131_2)

Tests are performed to determine whether a managed services provider's system controls were operating effectively during a specified time period. This time period is typically no shorter than six months and generally one year or greater. If during the audit period controls are operating effectively, an unqualified attestation report is issued. This report addresses whether a managed services provider has maintained effective controls over its system.<sup>27</sup>

On the other hand, SAS 70, is designed more to evaluate a managed services provider's internal controls. The auditor gives an unbiased opinion as to whether internal processes and procedures are suitably and properly designed, put into operation, and operating effectively.

It should be noted that there are two different types of SAS 70 reports, namely Type I and Type II. Type I assesses whether a managed services provider's internal controls are fairly and completely described, and whether they have been adequately designed to meet their objectives.<sup>28</sup> Contrarily, Type II reports are more rigorous and include the auditor's opinion on how effective the internal controls operated under a defined review period. The review period can be as little as six months, but is typically one year or longer. The substantial difference between Type I and Type II reports is that Type I only lists the controls, where Type II tests the efficacy of these controls to ensure the controls are functioning correctly.<sup>29</sup>

The major difference between the SysTrust and SAS 70 audits is the scope of the audit. While the SAS 70 audit provides a report on the effectiveness and adequacy of internal controls for a managed services provider, the SysTrust Audit provides a report covering an application's reliability.<sup>30</sup>

### *Ownership of Data*

While the SaaS model has a number of benefits that make it compelling, one of the more challenging hurdles facing SaaS providers has been data protection and ownership. The party responsible for safeguarding data may very well not be the same party that owns the data. In order to minimize risk and ensure that data is properly safeguarded, it is imperative that data is stored in a SysTrust and SAS-70 compliant environment, regardless of whether the data center is directly operated by the SaaS provider or through a managed services provider (see Data Security section above).

The need to restore data files may arise in as simple a situation as when a single file is inadvertently deleted by a user from primary disk storage or as catastrophic as an on-site disaster. Therefore, a restoration procedure and corresponding disaster recovery plan should be in place, which document specific measures to be taken in the event that data needs to be

---

<sup>27</sup> <http://infotech.aicpa.org/Resources/System+Security+and+Reliability/System+Reliability/Trust+Services/SysTrust/What+Are+Trust+Services+and+Why+Should+I+Get+Involved.htm>

<sup>28</sup> <http://www.technewsworld.com/story/saas/61448.html>

<sup>29</sup> <http://www.tech-faq.com/sas-70.shtml>

<sup>30</sup> <http://www.sas-70.us/sas-70-audit-vs-systrust-audit.html>

restored.<sup>31</sup> These set of procedures and/or plan should identify mission critical data and how often it should be backed up, describe where data is housed, backup procedures, what tests are run to ensure the probability of recovery, and what resources are responsible for data recovery. Additionally, a clear path should be detailed as to how easy a customer can retrieve data, both raw and processed, if a decision is made to leave a particular SaaS provider.

### *Integration*

A common definition of integration is the combination of parts so they may work together to form a whole. In technology integration, it many times refers to the act of bringing different applications together to run smoothly as one.<sup>32</sup>

Integrating SaaS applications with back-office systems is an important consideration in achieving the highest level of automation possible. Many applications integrate data or functionality from other applications through the use of mashups, whereby widgets (smaller subsets of data/functionality) are incorporated into a single viewable screen. The ability to successfully build such interfaces using a variety of protocols, formats, and other methods can be a critical factor when customers evaluate SaaS offerings.<sup>33</sup> Service Oriented Architecture (SOA) is the approach of choice when it comes to many integration strategies.

SOA is an architectural style for creating and using business processes, packaged as services, throughout their lifecycle. SOA also defines and provisions the IT infrastructure to allow different applications, both SaaS and on-premise, to exchange data and collectively participate in business processes.<sup>34</sup> These functions are loosely coupled with the operating systems and programming languages underlying the applications.<sup>35</sup> SOA separates functions into distinct units, or services, which can be distributed over a network and combined to create larger business processes and/or workflow.<sup>36</sup> These services communicate with each other by passing data from one application to another, or by coordinating an activity between two or more services, such as web services with varying degrees of encryption or security (further described under Data Security). SOA allows applications to pass data back and forth without in-depth knowledge of each other's IT systems behind a firewall.<sup>37</sup> From a business perspective SOA is beneficial because it allows for a more effective integration with business partners and supports customer service initiatives while protecting intellectual property.<sup>38</sup>

---

<sup>31</sup> <http://www.itpronews.com/itpronews-116-20080129HowtheRiseofSaaSRelatestoSOXSAS70YourLegalContracts.html>

<sup>32</sup> [http://searchcrm.techtarget.com/sDefinition/0,,sid11\\_gci212359,00.html](http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci212359,00.html)

<sup>33</sup> [http://www.aspnews.com/strategies/article.php/11223\\_3686131\\_2](http://www.aspnews.com/strategies/article.php/11223_3686131_2)

<sup>34</sup> [http://en.wikipedia.org/wiki/Service\\_Oriented\\_Architecture](http://en.wikipedia.org/wiki/Service_Oriented_Architecture)

<sup>35</sup> Newcomer, Eric; Lomow, Greg (2005). *Understanding SOA with Web Services*. Addison Wesley.

<sup>36</sup> Erl, Thomas (2005). *Service-oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River: Prentice Hall PTR.

<sup>37</sup> [http://www.webopedia.com/TERM/W/Web\\_services.html](http://www.webopedia.com/TERM/W/Web_services.html)

<sup>38</sup> <http://blogs.ittoolbox.com/eai/business/archives/soa-benefits-challenges-and-risk-mitigation-8075>

## *Application Updates*

Traditionally under the on-premise model the deployment of application updates and patches can be cumbersome and time consuming. Updates are meant to deliver enhancements to an application in order to improve functionality, user interface, or efficiency. Patches, on the other hand, are primarily released to fix problems or bugs with an application or its supporting data structure.<sup>39</sup>

With on-premise applications, updates tend to be larger and usually address major system extensions or improvements. Thus, updates to on-premise applications are scarce and usually occur between four to six months. Updates to SaaS applications, on the other hand, can happen as frequently as every four to six weeks. The reason updates are able to occur so often is because of the ability to roll out the application in a controlled environment. In other words, because a SaaS application is typically delivered through a multi-tenant platform (further explained in the Multi-Tenant Deployment section) only one instance of the application is being updated versus having to run updates on multiple instances of an application in the on-premise model.

More commonly, SaaS applications are developed using an agile development model. Agile development is a method of software collaboration, which allows for small development updates or iterations of an application to occur more frequently throughout the life cycle of an application. Agile development minimizes risk of creating new bugs because of the minimized scope of development. Development during one unit of time is referred to as an iteration, which commonly last from six to eight weeks. Each iteration is usually comprised of planning, requirements analysis, design, coding, testing, and the creation of user documentation. The goal of each iteration is to have an available application release without software bugs being introduced.<sup>40</sup>

With smaller updates happening more frequently there is less probability of unexpected maintenance outages. Updates are also relatively quick and typically scheduled on days and at times where there is the least amount of user activity (i.e. for business applications, late at night on a weekend), therefore, updates are less likely to disrupt access to an application.

## *Support*

Support is provided in various formats, either electronically or through human intervention to assist a customer in solving a problem or improved usage of an application. There are several ways a SaaS provider can assist customers. For example, support staff may offer to show a customer what steps to take via screen sharing over the Internet or perform a task remotely. Remote support is often used in conjunction with phone support.<sup>41</sup>

---

<sup>39</sup> [http://en.wikipedia.org/wiki/Patch\\_%28computing%29](http://en.wikipedia.org/wiki/Patch_%28computing%29)

<sup>40</sup> [http://en.wikipedia.org/wiki/Agile\\_software\\_development](http://en.wikipedia.org/wiki/Agile_software_development)

<sup>41</sup> <http://www.bitpipe.com/tlist/Remote-Support-Services.html>

According to Mural Ventures, top performing SaaS providers receive nearly 50 percent of their sales through customer referrals<sup>42</sup>, therefore, it is imperative for SaaS providers to offer customers the most responsive and effective support tools possible. Typically, support is broken down into two tiers, namely first and second tiers. First tier support refers to customers contacting a SaaS provider's support staff for assistance, primarily trained on application usage as defined within user documentation. First tier support is the initial point of contact and typically solves 60 to 80 percent of application-related issues. Often necessary to solve customer's more difficult issues, second tier support many times refers to a SaaS providers ability to make available more technical, and in many cases a software engineer, to help assess and resolve an issue. Second tier support staff is usually also responsible for training key members of first tier support staff, and help to provide further guidance on how to most efficiently troubleshoot an issue.

A major challenge in providing support is the ability to manage and prioritize telephone calls that come into a support center. Call volume is ever changing and can be difficult to predict. Some ways to reduce the number of calls and streamline the support process is to provide back-office tools such as an issue tracking or feedback tracking system, which allows for accurate and up to date reporting of issues (other back-office tools include project and implementation tracking, client billing, mass editing, and events management – further described in the second series of this whitepaper entitled "Back-Office Automation").

Feedback tracking systems are commonly used to create, update, and resolve reported customer issues through tickets. A ticket is a record contained within a feedback tracking system that includes information like submitter, application component, severity, along with ongoing progress being made to resolve an issue. Typically, each ticket has a unique reference number, which allows the support staff to quickly locate, add to, and communicate the status of an issue or request, both internally and externally. Urgency is also assigned to a ticket based on the overall importance of that issue. Other details found in tickets may include a time stamp of when the issue was experienced, date of submission, detailed descriptions of the issue, screen captures, attempted resolution, and other relevant information.<sup>43</sup>

Feedback tracking systems are beneficial because they are an effective accurate way for customers to report the issue that they are experiencing at the moment it is being experienced, which in turn allows for a higher percentage of relevant tickets to be submitted and greater accuracy of each ticket. As data is collected within a feedback tracking system, it is pooled to create an overall knowledge base of information for decision-making, including improvement of an application. A knowledge base can be very powerful in provider a great degree of customer support, in that it gives support staff the ability to pull information from previous tickets to solve current issues. Feedback tracking systems also reduces the number of phone calls that come into a support center, allowing support staff to better prioritize and attend to issues in a timely manner.

---

<sup>42</sup> <http://www.muralventures.com/increasing-saas-sales.htm>

<sup>43</sup> [http://en.wikipedia.org/wiki/Issue\\_tracking\\_system](http://en.wikipedia.org/wiki/Issue_tracking_system)