**Title**
The User Experience of Software-as-a-Service Applications



**Author**
Rhoads Lindholm, Katrina

**Publication Date**
2007-03-02

# The User Experience of Software-as-a-Service Applications

**Katrina Rhoads Lindholm**

2007 Information & Service Design Symposium
The School of Information, UC Berkeley

UCB iSchool Report 2007-005
February 2007

**Abstract**

Over the last several years we have seen a dramatic increase in the number of software applications offered over the internet. The ability to release user interface changes on a potentially daily basis has forced user experience professionals to rethink their traditional linear methodologies. With a new set of internet-based usability techniques as well as the remarkable ability to receive real-time, continuous feedback from end users, designers today have the potential to create the most usable and competitive software user interfaces to date.

## Introduction

The software-as-a-service (SaaS) delivery model has grown in popularity over the last decade as computers and the internet have gained acceptance in both the corporate world and in the home. In conjunction with this movement, there have been significant changes in how software is developed. Shorter development cycles and more frequent releases are becoming the norm for web-based software. New agile development processes which promote increased productivity through shorter cycles have also gained popularity in the software industry. Accordingly, user experience tasks such as interface design and usability testing have been forced to adapt. This paper will discuss the user experience challenges and solutions that have emerged in response to the advancement of SaaS.

## Growth of the SaaS Delivery Model

Progress in technology has enabled software providers to look beyond disks and CD's as a medium for delivery; today an increasing number of companies like Google, Yahoo! and Salesforce.com are offering their software in a web-based environment that can be accessed at any time and from any computer across the globe. A 2005 report found that of the North American professionals surveyed, 79% said that they were already using or were considering using SaaS applications.[1]

Advancements in computing and the internet are the primary reason that SaaS providers have been able to gain a foothold in the software market. Decreasing hardware costs, increasing computational power and improved network reliability have encouraged both businesses and consumers to invest in computers and internet services. Nearly 70% of people who connect to the internet at home in the U.S. do so with a high-speed connection.[2] Within the business setting, 89% of workers access the internet with a high-speed connection.[3] This widespread adoption of high-speed internet service has enabled SaaS applications to perform at speeds competitive with traditional, locally-installed applications. Furthermore, secure connections and encryption techniques have helped convince the public that the internet is a safe medium for exchanging sensitive data.

In addition to the technical factors involved in the growth of SaaS acceptance, there are several benefits over the traditional software delivery model that have attracted businesses and consumers. One such benefit is derived from the one-to-many model that is intrinsic to SaaS applications. This means that the SaaS provider maintains a single code base on its own set of servers from which every customer instance is derived. This is a powerful architectural design that makes it possible to roll out a release across all customers at the same time, without the need for customer-specific modifications.

Because SaaS software is not hand-tailored to the needs of the customer, providers often build a set of tools into the interface that allow users to customize the application themselves. For example, Salesforce.com delivers an identical product to each new customer. However, the customer can choose which

modules they would like to use, which tabs appear on the page and what fields appear in each form.

Another benefit of the SaaS delivery model is that the responsibility of maintenance is shifted almost entirely to the SaaS provider. Because the software is hosted by the provider rather than installed locally at the customer's site, all hardware and software maintenance must be performed by the provider. In a business setting, this means that a special IT team is not required in order to install updates, maintain hardware and back-up data. For consumers using SaaS software, it means that any information they enter into the software will persist even if their own personal computer fails.

Cost is another major factor. Many consumer SaaS applications are completely free of charge; the provider earns revenue from advertising. Business SaaS applications are typically tied to a subscription model which is attractive because the upfront costs are minimal and the contract can be terminated at any time.

There is also reason to believe that customer support is enhanced by the SaaS delivery model. This may be partially due to the fact that it is easier for providers to diagnose and solve problems when the server environment is in-house and predictable. In addition, the single code base of SaaS providers eliminates the need to support legacy versions, simplifying the work of customer support representatives. Another factor that may lead to improved customer support is the ability of SaaS providers to monitor the usage of their product in real-time. Analysis of these logs can help providers gain intimate knowledge of customer behavior and may lead to faster problem resolution in the short term and fewer bugs in the long term.

## Release Early, Release Often

For years, the development of software closely modeled the traditional design and manufacturing processes of physical products. Feature planning and design take place upfront, followed by a long development phase and concluded with QA testing, bug fixes and, ultimately, the release of a new version. This process might last anywhere from six months to several years. Microsoft, for example, recently announced the debut of its latest Windows version, Vista, which took over five years to develop.[4]

This lengthy waterfall-style development process is a natural fit for versioned software that is delivered via a physical medium such as a CD because 1) it is not economical for companies to repeatedly issue new disks, packaging and printed documentation and 2) it is not practical to ask customers to purchase and install a new version more than once a year. If software developed in this fashion could be guaranteed to be perfect, infrequent updates might be acceptable. But in reality, there are always bugs that QA overlooks, spelling mistakes that copy editors miss and, worse, the not uncommon situation in which the market changes before the software has even shipped.

Today, the internet as a delivery medium has helped revolutionize the software development process. Traditional desktop software providers like Microsoft are now utilizing the web to allow customers to install new versions

and to download the latest bug and security patches. Meanwhile, SaaS providers that rely entirely on the internet to deliver their products are able to push out changes both large and small to their software as often as they desire. For example, Flickr, an online photograph sharing website, is said to have released new code as frequently as once every half hour[5], while Netflix, an online movie rental company, maintains a consistent two-week update cycle.[6]

## From Version Numbers to Beta Labels

Frequent releases have also changed the way that companies communicate the state of the software to their customers. Version numbers are unable to keep up with changes that take place on a potentially daily basis. Instead, we are seeing more and more new SaaS applications labeled with the words "alpha" and "beta" which quickly provide users with a rough indication of the stage of development.

By labeling an application as "beta," some believe that the provider is simply making an excuse for any bugs, mistakes or incomplete features in the software. Others think that the beta label is an honest declaration to customers about the state of the software and a powerful way to gather valuable feedback and conduct live user testing. Either way, the beta phase has allowed SaaS companies to beat would-be rivals to the market and to innovate in a way that may have been considered too risky just a few years ago.

> "I think that people are maybe ashamed of their products and are worried about releasing something that's not perfect. It feels like it's almost an excuse. They're putting something out there and saying, 'Use this, but if it's not perfect, it's not our fault.'"[7]

In addition, some SaaS providers are including sections on their website that showcase new features and applications in their infancy. For example, the Google Labs page lists several new applications such as Google Ride Finder and Google Code Search that, according to Google, "aren't quite ready for prime time." YouTube also has a section called Test Tube where they feature "recipes and concoctions that aren't quite fully baked" for users to experiment with. Similar to the beta label, these experimental sites allow SaaS providers to gain valuable feedback from customers that they can then incorporate into subsequent releases.

## User Experience Best Practices for SaaS

The ability to release software updates on an as-needed basis can be a powerful advantage for SaaS providers when implemented properly. With that said, the company has a responsibility to the end users to take care in how user interface changes and new features are released.

Unlike traditional shrink-wrapped software, changes to SaaS applications are made live the very next time the user logs into the system. As a result, the customer has very little say in what changes will be released. They also lack the

option of uninstalling the new version and reinstalling the previous one as they did with traditional software. It is not surprising then that changes to a user interface can disrupt and even anger users who are accustomed to a certain page organization and set of tools for accomplishing their tasks.

In early September 2006, the social networking site Facebook released a news feed feature with the purpose of helping members keep track of changes that friends made to their pages. The feature received an enormous amount of criticism with approximately 700,000 members signing a petition for its immediate removal. [8] The Facebook team clearly did not anticipate such a reaction and may have avoided it by performing usability testing during the design and development phases. If standard usability tests failed to reveal the issue, there are several effective user experience techniques for SaaS applications that could have given Facebook more control over the situation.

### *Limited Releases*

One technique that has been used successfully by companies like MSN and Yahoo! is to offer a new feature or an updated look-and-feel to a small percentage of the user-base. MSN employed this technique when they were redesigning their home page a few years ago. Rather than forcing the change onto all of their users, they advertised a preview to a small percentage of their users. They used the feedback they received to gauge the users' reaction, make improvements and then eventually released the new home page to all of their users.

> "Don't people hate it if you change the pages every two weeks? Of course. If you have enough traffic, test the changes on a slice of users."[9]

In the case of Yahoo! Mail, existing users were shown a link that invited them to try out the new Yahoo! Mail Beta application. Yahoo! Mail Beta offered "Web 2.0" features such as drag-and-drop and automatic completion of email addresses and was a drastic change from their existing Mail application. Yahoo!'s invitation strategy helped them avoid angering existing users by forcing a drastic user interface change on them. Users who voluntarily opted to try the new application were given a very prominent link to switch back if they preferred the previous version. Like MSN, the real-time usage statistics and feedback were incorporated into subsequent updates, creating a more mature, stable application.

### *Supporting Older Features*

YouTube has dealt with rolling out major feature changes in a similar fashion. Hong Qu, a UI designer at YouTube, explained in a telephone interview that they mitigate the rejection of changes to popular features by first performing several iterations of traditional usability testing and then, following a public release, by allowing users to continue to access the previous version of the feature. Over time, even the reluctant users tend to adopt the new version of the

feature. By performing log analysis, YouTube can determine when the old version is no longer being used and will subsequently remove it from their site.

> "We are really careful – every time something new comes out, the users are hesitant to accept it. We are careful to preserve the impression that we are supporting the old feature."[10]

Like many other SaaS providers, YouTube also maintains a close watch on customer feedback in order to validate interface changes. Several times a week, Hong will check-in with the customer support department to find out what features customers are requesting and what they are having difficulty with.

### *Gradual Change*

Another strategy is to avoid major design overhauls altogether. Rather, design the end result and then slowly release elements of it overtime. This slow-release process ensures that a user's productivity is not impeded by having to learn a whole new interface. Ideally, the user will not even notice the changes.

There is an unconfirmed anecdote about EBay taking this approach after they attempted a drastic change from a yellow background to a white background and elicited severe criticism from their end users. Rather than give up on the idea altogether, they decided to slowly fade the color from yellow to white over an extended period of time. Because the change was so gradual, users did not even notice.[11]

Gradually releasing small changes to a user interface is also beneficial because it allows the user experience team to pinpoint the exact cause of positive or negative feedback. If several changes are made, it can be difficult to determine which one was the true cause. Furthermore, it is easier to remove a small change from an interface than a large change if it is found that it is not being used or is disliked by users.

## Different Breeds of SaaS Applications

Certainly, there are differences between SaaS providers that make some user experience techniques more practical and effective than others. Although the line is fuzzy, there appear to be two main camps of SaaS applications which are divided primarily by target audience and cost structure:

1. Business-to-Business SaaS: Applications that are sold to small-to-medium sized businesses and that charge a monthly subscription fee. Examples: Salesforce.com, WebEx, SugarCRM and 37 Signals.

2. Business-to-Consumer SaaS: Applications that are made available to individual consumers and that are either free to use or may charge a nominal fee. Examples: Google GMail, Google Spreadsheet, Yahoo! Mail, PayPal and Flickr.

It may be that the greatest difference lies in how critical the application is to the end user. As one may expect, end users of a business-to-business (B2B) SaaS application such as Salesforce.com are not using the application voluntarily. Rather, their employer has selected the software on their behalf. The software's availability, stability and usability are of utmost importance to the customer. This is particularly true in the realm of customer relationship management (CRM) software because the productivity of the employee using the software is directly related to the success of the company. As a result, changes to this type of SaaS software cannot be taken lightly.

> "...if you're running an online billing system, hosted CRM, online banking or hosted VOIP services - you can't afford to screw things up for your users. At the end of the day, consistent quality is the key to keeping customers happy - and generating revenue for you."[12]

Salesforce.com's longer release cycles may be indicative of this criticality factor. Although they recently converted to an agile development process in an effort to increase the number of yearly releases, Salesforce.com must still meticulously navigate new features and user interface changes in order to maintain a high level of customer satisfaction.

When Salesforce.com started thinking about updating the look and feel of their navigational tabs in 2005, they utilized their successforce.com community website to collect comments from current customers on three potential designs. When the change was later implemented in early 2006, they released the new look and feel for all new customers but allowed existing customers to revert back to the "classic" style if they so desired. Even a year later, Salesforce.com is supporting a small percentage of users who are using the old look and feel. This underscores the value of giving customers an option when significant interface changes are implemented.

For Salesforce.com, it is not just about what is perceived to be better for the customer. Each design decision they make has the potential to cause productivity loss and an increase in human error. Furthermore, for larger customers, changes to the interface can mean weeks lost in retraining employees. As a result, Salesforce.com tries to conduct as much usability testing as possible and leverages online and offline communities to receive feedback about features. They will also often make new features optional by providing on and off switches in the software's administration panel. At the end of the day, these safeguarding tactics must be balanced with the need to constantly evolve in order to stay competitive with other SaaS CRM providers.

> "When considering the value of a conceptual feature or usability change, ask 'What does the customer do with what she has today, will she still be able to do it tomorrow, and does she need what I plan to give her next?'"[13]

For SaaS applications that cater to consumers (B2C), the story can be quite different. Applications like Flickr, Facebook and Google Earth are predominantly used for social networking, community building or as a leisure activity. They are

all offered free of charge. Certainly, one would expect that these factors would substantially influence the expectations of the users.

Although it would be erroneous to say that users of free B2C SaaS applications are accepting of design flaws and reckless change, in some cases they may be more flexible than users of B2B software. Furthermore, there may be an inherit expectation that these types of applications should remain cutting-edge, thus demanding the constant addition of new features and an updated look and feel.

> "One cool thing about these constant updates, is that it involves the user in the development process, even just a little bit. It makes me feel like I can direct how the software goes, even if I'm not the one actively working on it. I think I only get this feeling because of the _immediate_ feedback. If I made a request and it gets implemented a few months later, I probably would have forgotten about it already."[14]

While there is no evidence that users of B2C SaaS applications are more open to interface changes, we know that providers like Netflix, Flickr, Facebook, MySpace and Google are constantly tweaking their interfaces in an effort to stay competitive in a market driven by rapid innovation.

## It Comes Down to the Users

As satisfying as it would be to draw a thick line in the sand between B2B and B2C SaaS applications, the reality is that design practices always need to come back to the needs of the users. B2B SaaS applications will often have a less technical, more conservative user base and B2C SaaS applications will sometimes offer an opportunity to push the limits of design. But there are exceptions to every rule.

Microsoft's Hotmail application is a good example. The designers of Hotmail, one of the first free email programs on the web, took one look at Google's GMail and decided it was time for a major upgrade. The team designed and developed a major overhaul to the interface including all of the cutting-edge "web 2.0" bells and whistles. Their first set of user testing delivered good results. However, when they started broadening their tests to a more representative set of users, they began getting some unexpected feedback. Users actually preferred the existing Hotmail interface.[15]

The rationale was two-fold. First, many of these users had held a Hotmail account for almost ten years and were very comfortable with the current functionality. Furthermore, Hotmail's design had not changed substantially over its lifespan so users were not accustomed to change. The second reason for the negative reaction was that the proposed design took significantly longer to load in the browser than the current design. With a large number of users relying on dial-up internet access, the exciting new "web 2.0" features ended up being less usable than the features they were replacing.

## User Experience and Agile

Another trend in software development that is gaining popularity among many SaaS software providers is the agile development process. Agile attempts to make development more productive and efficient by replacing a single long cycle with several shorter ones. Each of these "sprints" focuses on a prioritized set of user stories and is intended to deliver real, functional value. Typically lasting between a week and a month, a sprint will cycle through the phases of design, coding and testing.

Companies that have implemented agile have had to reevaluate the role of the user experience team within the development process. In a traditional user-centered design process, almost all of the user interface (UI) design work is done upfront. User needs serve as the foundation for creating a series of rough mockups which eventually evolve into high-fidelity interactive prototypes through several iterations of customer testing. When the team is satisfied with the design it is typically handed off to the development team for coding. Ideally, user experience members will have the opportunity to work closely with engineering to ensure that the code stays true to the design. A final round of usability testing will reveal any unexpected issues and minor changes can be made before releasing the feature to the public.

With agile, however, long periods of upfront planning are discouraged. In fact, some user experience professionals have criticized the agile methodology because it seems to leave UI design and usability testing entirely out of the process. However, companies that have adopted agile but want to continue supporting the user-centered design process have started to find ways to reconcile the two seemingly opposing methodologies.

Most user experience professionals agree that some design work still needs to be completed upfront along with the initial feature planning. This is because highly usable software is not derived just by how things look and work on a single page but how well the various pages interact with one another. For particularly large features or features spanning multiple sections of an application, taking a holistic perspective in the design process is a necessity. With a holistic design completed, the UI Designer can then segment the design to fit the needs of each sprint.

Alias, a graphics software company, has said to have effectively integrated user experience work into their agile development process. They complete their design work one iteration ahead and conduct usability testing on the features developed in the previous iteration.[16] In this arrangement, the team is designing and testing in a single sprint and never holding up the work of development or QA. Any recommendations for change that stem from usability testing can be prioritized and then incorporated into a subsequent sprint.

Salesforce.com, which transitioned from a standard waterfall approach to a Scrum agile development process in the fall of 2006, has also had to make adjustments in order to align the user-centered design process with their agile schedule. According to Jerry Sherman, Senior Director of Development Services at Salesforce.com, it is nearly impossible to fit the full user-centered design cycle into a 2 or 4-week sprint although the company considers this step to be critical

to getting the feature right. Their solution has been to have the user experience team start their work several cycles ahead of development.

> "We're focusing on having the designers work proactively with project managers several sprints ahead of the rest of the team. The goal is to have fully-tested prototypes ready to hand off to development before the release sprints start."[17]

Some claim that, when done properly, the agile process naturally enhances the usability of the software. For example, the short sprints may prevent designers from become overly attached to the feature and therefore they may be more open to changing or even throwing out their work if usability testing reveals flaws or if market demand has changed. In addition, frequent updates to the software give the user experience team an opportunity to see real-time usage statistics and determine if the change was accepted by the end users. In the case where a change or feature receives a negative reaction when made live, it can be removed or modified in the very next release.

> "In an agile world, teams are allowed to be wrong and quickly change their course."[18]

## The Future of SaaS

The benefits of software-as-a-service applications are myriad. More attractive pricing models, reduced dependency on IT and convenience of access are all reasons why companies and individuals will continue to embrace SaaS applications. But with those advantages aside, there are three fundamental reasons why the internet will become the dominant method for delivering software: the ability to watch, listen and react to users in near real-time.

It is difficult to identify another industry today that has such a direct connection to the needs and wants of their users. With SaaS software, the interface can be tweaked and moments later users are flooding the customer support department with feedback. Furthermore, server logs are able to provide insights into customer behavior. For example, it is possible to see where users are clicking and what they were doing when they decided to leave the site.

User experience teams working for SaaS companies truly have every resource at their disposal for creating innovative and highly-usable interfaces. However, the advantage of continuous evolution via ad-hoc code updates must be balanced with the users' needs for consistency, efficiency and transparency. SaaS providers that practice intelligent user-centered design and that can form a two-way line of communication with their users by leveraging user feedback and log data will be able to differentiate themselves from the competition.

## References

[1] Erin Traudt, Amy Konary, *2005 Software as a Service Taxonomy and Research Guide*, June 2005, <http://www.idc.com/getdoc.jsp?containerId=33453&pageType=PRINTFRIENDLY> (9 December 2006).

[2] Website Optimization, *April 2006 Bandwidth Report*, April 2006, <http://www.websiteoptimization.com/bw/0604/> (9 April 2006).

[3] Ibid.

[4] Peter Galli, *Pushing Forward - the next version of Windows*, 30 July 2001, eWeek.com, <http://www.eweek.com/article2/0,1759,113701,00.asp> (11 December 2006).

[5] Tom Coates, *Cal Henderson on "How We Built Flickr"...*, 20 June 2005, <http://www.plasticbag.org/archives/2005/06/cal_henderson_on_how_we_built_flickr/> (9 December 2006).

[6] Joshua Porter, *The Freedom of Fast Iterations: How Netflix Designs a Winning Web Site*, 14 November 2006, <https://www.uie.com/articles/fast_iterations/> (9 December 2006).

[7] Jason Fried, *Mo' Beta Testing Blues*, 29 May 2004, Wired.com, <http://www.wired.com/news/infostructure/0,1377,63631,00.html> (9 December 2006).

[8] Bruce Schneier, *Facebook and Data Control*, 21 September 2006, <http://www.schneier.com/blog/archives/2006/09/facebook_and_da.html> (9 December 2006).

[9] Walter Underwood, in response to *The Freedom of Fast Iterations: How Netflix Designs a Winning Web Site*, 15 November 2006, <https://www.uie.com/brainsparks/2006/11/14/uietips-article-the-freedom-of-fast-iterations-how-netflix-designs-a-winning-web-site/> (9 December 2006).

[10] Hong Qu, Telephone Interview, 8 February 2007.

[11] Jared M. Spool, *The Quiet Death of the Major Re-Launch*, 20 May 2003, <http://www.uie.com/articles/death_of_relaunch/> (9 December 2006).

[12] Matt Smith, in response to *Ultra-fast release cycles and the new plane*, 17 March 2006, <http://headrush.typepad.com/creating_passionate_users/2006/03/ultrafast_relea.html> (9 December 2006).

[13]Frank Fulton, *Release 101 for SaaS and Web 2.0*, 3 October 2006, <http://blog.klir.com/klir_technologies_weblog/2006/10/release_101_for.html> (9 December 2006).

[14] Skrud, in response to *Ultra-fast release cycles and the new plane*, 17 March 2006, <http://headrush.typepad.com/creating_passionate_users/2006/03/ultrafast_relea.html> (9 December 2006).

[15] Ina Fried, *Hotmail's New Address*, 26 April 2006, <http://news.com.com/Hotmails+new+address/2009-1038_3-6064507.html> (11 December 2006).

[16] Jared M. Spool, Joshua Porter, *Agile Development Processes: An Interview with Jeff Patton*, 12 September 2006, <https://www.uie.com/articles/patton_interview/> (9 December 2006).

[17] Jerry Sherman, Email Conversation, 12 February 2007.

[18] Jared M. Spool, Joshua Porter, *Agile Development Processes: An Interview with Jeff Patton*, 12 September 2006, <https://www.uie.com/articles/patton_interview/> (9 December 2006).