Abstract. SaaS (Software as a Service) has grown rapidly. Accompanied by the Internet’s evolution and increased resource availability, many believe that the traditional software delivery model may be at risk of being eclipsed by web-based software, remotely hosted that end all installation, maintenance and upgrade costs that weight in every IT organization’s budget. Several companies are providing a new type of online service, that provides clients with the tools to build their own services, totally configurable and with no programming needs. This dissertation’s course project goals to study and compare several of these SaaS initiatives. In addition, this project proposes a draft of a new set of features to be implemented on the WebComfort platform.

Keywords: Software as a Service (SaaS); Service Level Agreements (SLA); Internet; Business; Variability, WebComfort; WebC-SaaS.
1 Introduction

Since its early days, Information Systems greatly changed the way enterprises do business. In general, Information Systems are designed to be deployed and to run on isolated and independent machines. The appearance of computing networks gave Information Systems new ways to evolve, that led to other communication oriented software architectures such as client-server and peer-to-peer. As such, many organizations adopted these new technologies, connecting several workstations, sharing data and software through the network. However, the cost of developing, deploying and maintaining their own personal Information Systems was, and still is today, an enormous company budget effort. Thus, only large enterprises could afford such systems, leaving a sea of small and medium enterprises out of the equation.

To deliver software to a broader market share, a new business model emerged. Application Service Provider (ASP) provides Application hosting, maintenance and upgrade, that can be used by its clients over a network. This new business model was especially suitable for smaller enterprises who lacked the high-cost infra-structure necessary to run such systems in addition to the specialized personnel to maintain and perform upgrades.

Following the decline of the ASP model, the SaaS (Software As a Service) business model appeared as ASP's successor, to emend its disadvantages and covet its outstanding opportunities. Like the ASP model, SaaS delivers outsourced services to its customer via a network, but in this case the network is always the Internet; the software applications being run are intimately known by the provider; and every application is delivered to many customers. As a result, these providers can offer customers their own solutions with value-added features. While this would be expensive in the ASP model – due to the deficient knowledge over all the applications hosted for each client –, in the SaaS model this is possible because every service is well-known to the provider. Furthermore, this allows the SaaS model to achieve economies of scale. Contrary to ASP, SaaS clients don’t have to buy the software solution and then pay the provider to host it, but rather just pay for the usage of the service.

Initially, companies such Salesforce.com\(^1\) with their well-known Customer Relationship Management (CRM) focused on bringing online versions of popular enterprise software products. But while developing such applications, companies had overcome many obstacles to achieve high performance in a multi-tenant and single-instance environment, creating a set of software features to support the actual service providing. It would be a matter of time for Salesforce.com and other companies to realize the true potential of this supporting software.

In order to create such services, companies like Salesforce.com have to provide proprietary programming tools and documentation to their platform customers. This forces enterprises to require programming knowledge and an IT department dedicated into understanding and developing the service on top of Salesforce.com platform, thus creating an obstacle to many possible customers that don't meet such requirements. A new breed of SaaS providers have found the solution for this problem.

In an industry where companies whose revenues are greatly diminished due to the illicit use of their proprietary software, greatly because almost impossible to track down the usage of traditional packaged software, the winds will always favor Software as a Service.

2 Problems

In the early 2000's many prophesized the rising of Software as a Service as the most lucrative and used software delivery model [12]. In today's extremely variable market conditions, many organizations would prefer the less risky model that SaaS provides.

\(^1\) http://www.salesforce.com
Benefits typically attributed to SaaS include easier deployment, maintenance, and upgrades; flexible licensing; and better application life-cycle management. These qualities alone should be enough to eclipse the traditional software delivery model but, almost ten years after its birth, that didn't happen. SaaS' model faces a new series of issues that traditional software delivery didn't have to worry.

In general, some of the problems these organizations generally encounter when creating and managing their own SaaS are related to conservative companies, or those with truly critical data wanted to have the data on site, or wanted to self-support their own architecture, or wanted to modify the software at their choosing.

SaaS aims to satisfy all types of organizations, but their benefits attract specially the small and medium enterprises. Unlike large enterprises, SME's internal management may change very quickly, along with their unique business processes and internal company structure. Developing software that tries to attend all those individual different needs may be extremely difficult to perform, and many will likely fail.

It's not just the conservative companies that worry about securing their data. In fact, this is one of the main reasons why companies reject the SaaS model. Having financial and client data hosted off-premises include many risks of data leak or loss that stand tall between SaaS and Customer acceptance.

Putting customer's acceptance apart, developing SaaS also affects organizations internal structure. Selling software licenses and selling SaaS services are totally different business models, both need suitable organization structures to work.

Furthermore, market and credit laws deeply affect SaaS delivery model. Data protection services and payment transaction system credibility need a social credit system to support them. There is a considerable number of developed countries where conditions are met so that companies can opt for such services, but in less developed countries, without proper legislation and protection, customers are vulnerable to frauds and scams.

3 Objectives

This project aims to analyze and discuss several SaaS initiatives in the market, as well as apply the resulting study developing WebC-SaaS platform. Through this work and State of the Art analysis, a discussion is made around some of the most important initiatives of this type of service in today's market, based in a set of parameters that distinguishes them apart and also influence their potential. The practical component of this project, consists in developing, over WebC-SaaS platform, a set of functionalities that result from the State of Art discussion, considered essential to achieve success in the market. Finally, as a result of this work, an instance of WebC-SaaS containing all the resulting improvements will be produced, installed and evaluated.

4 State of the Art

Ever since the beginning of Software as a Service, organizations faced many obstacles directly related to SaaS model and have adapted according to their business goals. Some, like in the ASP model, narrowed their target market to subsist.

In the world of Small and Medium Enterprises, customers demand new functionalities regularly, pleasing every single client implies software customization and further maintenance of all client upgrades and versions. The exponential growth of development and maintenance costs is foreseeable, and is also one of the reasons why the ASP model collapsed in the 1990s[1].

This leads to the introduction of customization versus configuration. Both can support service adaptation effort to a certain limit. This limit that distinguishes them is related with complexity. Configuration doesn't involve code changing. Normally it supports variations through pre-defined parameters and settings, or providing tools to change application functionalities within a pre-determined
scope, for example: add new data fields, change field names, modify buttons and lists, change business rules, etc. In other words, configuration supports requirement adaptation within pre-defined limits. [7] Customization usually involves changing application's source code to create functionalities that go beyond the limits of configuration. When compared to configuration, customization is an expensive approach for SaaS vendors as well for customers. Changing code brings several issues with high costs associated: extra-qualified personnel with higher salaries to work in customization; resource and infrastructures allocation to manage multiple code versions; significant length increase of development/debug/testing/installation cycles; losing clients that can't afford customization costs. [7] SaaS vendors should avoid customization at all costs, using configuration to satisfy adaptation requirements and should maximize their configuration limits to fit as many client's requirements as possible. [3]

This project's work reflects on the state of the art in the Configurable Software as a Service business. To better understand this concept, this project will focus on five of the most relevant initiatives of this SaaS approach: Google Sites, Webnode.com, Moogo.com, Site2You.com and WebC-SaaS. Following is summarized a study of these services based on a set of parameters that reflect the main features to overcome some of the core issues in SaaS.

4.1 Concepts

In this section, some of the most common concepts are explained in detail in order to understand how different the studied initiatives are.

4.1.1 Site

Although the final product seems relatively similar between SaaS vendors, the concept of Site that supports configuration may be fairly different. There are two main approaches regarding this:

**Multiple Instance** - If each content page corresponds to a different instance, it's a case of a multiple instance site.

**Single Instance** - In some cases, customers only have access to a single page in which modules and content vary by updating certain parts of the same page, simulating several pages.

4.1.2 Page Templates

In this type of service, the page structural design is defined by templates that vendors provide. These given templates differ in the way that service pages are divided and where customers can add new modules. Some vendors also provide the definition of new templates.

4.1.3 Modules

Diving content by modules is a well-known and accepted architectural decision in this line of business. A module represents the base unit of all service functionalities. Each service may include several modules developed by the vendor that hold one or more functionalities.

In terms of presentation, this module-breakdown structure makes possible to change the position of both content and functionalities inside the page without effort. In terms of platform expansion, it allows both
the SaaS vendor and third-party organizations to rapidly include new functionalities totally independent from the rest of the application.

4.1.4 Themes

As part of the service presentation configuration, many vendors provide a set of pre-defined visual themes, with custom fonts and color schemes to apply over service's content. Some vendors, besides providing themes, offer the chance to create and define custom visual themes.

4.1.5 Usability

As in every application that requires user interaction, its interface has a fundamental role in customer's satisfaction. In today's reality, due to Web2.0 technology, it is possible to simulate many functionalities that once were reserved to traditional desktop applications. In the early stages of graphical user interfaces, designers had to balance performance, esthetics and usability to create a functional and attractive interface. In today's browser interfaced applications, these problems still persist and with increased response times that Internet adds to the equation.

4.1.6 Roles Access Management

In this business line, customers that subscribe this type of SaaS service, have in mind the possibility to profit with it. To do so, SaaS vendors need to provide such mechanisms that allow customers to manage and control their own clients and access to multiple service sections. This sub-division allows these SaaS customers to define their own service levels for their clients, or to provide exclusive features for registered users. Through the SaaS vendor perspective, Roles Access Management creates paths for their customers to obtain revenues, which often translates into more income for vendors.

4.1.7 Service Levels

One of the most variable aspects is how companies decide to explore business and acquire revenues. As such, studying these strategies becomes relevant as companies encourage customers to subscribe higher service levels. Designing service levels is, on its own, an extremely vast area filled with small variants that define a fine line between success and failure. Each company deals with an unique set of functionalities that, according to its business strategies, distributes through service levels.
4.2 Features

In this section, the most relevant features are described in detail as well as their importance in this business model.

4.2.1 Domain Name

Domain Name is the service entry point for clients. There are several different solutions for a domain name. Some providers offer a sub-directory inside the company’s site, others may offer a totally independent domain name solution.

4.2.2 Multi Language

A multi-language platform, with every functionality, tool-tip and documentation translated, speeds up the service configuration and increases customer’s satisfaction. Besides platform language, there is multi-language content definition. Few SaaS vendors provide such feature, without having to duplicate and maintain a copy of all of the service separately, forcing organizations that subscribed services to choose only a sub-set of their potential clients or, doubling their effort by maintaining a new copy of their service in another language, which often implies subscribing another instance of the service.

4.2.3 Statistical Analysis

Since SaaS vendors host customer's data and services, they can infer directly over them, gathering information about visitors or resource usage, offering extra information that go beyond services like Google Analytics. From a business perspective, this feature can help costumers to understand the market reaction to their service and to make adjustments in order to improve sales or number of visitors.
4.2.4 Support, Backup and Recovery

Having data hosted off-premises is one of the issues that scares many possible SaaS customers. Furthermore, not every SaaS vendor takes full responsibility in case of data loss or corruption. As such, many customers believe to be protected against such risks but, in reality, they're not. As a fully mature backup and recovery system is expensive to develop or maintain (especially if not carefully planned during application design), not every SaaS platform provides such features.

4.2.5 Configuration and Content Management

A service may include unlimited features but if they are difficult to configure, it is likely that customers will never use them. As such, it is crucial to present all available options to the user. Some developers approach this through a site oriented way, in which they gather all features that can be configured, order them by groups and then split them into menu sections. Another common approach is module oriented configuration where modules have their own configuration panels, and show only the existing options for the selected module.

4.2.6 Configuration Interface

Creating a totally new interface design is an approach that almost none SaaS vendor ponders. A new design created from scratch needs to be studied, tested, tuned and in the end, no one will guarantee customer's satisfaction. As such, most of vendors prefer to use a design that is largely used and recognized. In a standard enterprise environment, it is fairly common the use of office tools such as Microsoft Word, searching the web and the use of file systems to store or read files.

5 Solution

To understand the design concepts that support this project, next are described the fundamental classes and relations of the Domain Model. This section is centered around these concepts, exposing in detail what roles they have in the system. Firstly, the main actors are identified, along with the most relevant use cases. Then, it is presented the list of requirements that result of the State of the Art analysis and finally, the Domain Model is shown along with its main concepts.

5.1 Domain Model

During this project's development, some original features offered by the WebComfort platform had to be extended. Thus, WebC-SaaS includes a rich Domain Model that support all extensions and new concepts that, altogether, implement the SaaS model. To better understand the Domain Model behind WebC-SaaS, this section is centered around these key concepts.
As it can be seen in the picture above, the main concepts captured in this package are the Service, Subscription, Sandbox a Customer classes. These concepts assemble all the information relative to a service subscription, all its functionality and configuration, the validity of the subscription, and who can access what and where in the platform.

This view reveals the relations between the main concepts of the Domain Model. As it can be seen, every concept in this view relates to Service and Customer objects. What’s more, there are generally two types of associations between Customers and the other concepts. This is due to the fact that Customers can behave as Administrator and User abstractions.

The Subscription, Sandbox, and Customer concepts are defined as abstract classes. This means that these concepts are only generic types and are mainly used to define an interface that must be implemented by other classes.

### 5.2 WebC-SaaS Extensions

In order to meet the required levels of platform control, it was necessary to design a tracking system to capture users activity, leading to the development of these features.

#### 5.2.1 Page Crawling

The system is able to track which pages have users visited. In addition, this set of features include counting total page hits, recent visitors of a page. Not only the system can retrieve information about others activity, but also debrief the user of its own past activities, enabling an history log of its activity over time.

#### 5.2.2 Service Configuration
Since WebC-SaaS is built over a CMS, it benefits from many dynamic content tools. Offering such raw and powerful features to users could have drastic consequences. Thus, it was necessary to filter these tools, and provide them in a safe manner. To this result, the Statistics Package control is responsible to control resources and content production by the users, in order to maintain the platform coherent and stable.

5.2.3 User Interaction

In order to meet some requirements, users had to be capable of exploring services the platform provides. Thus, a system composed of a simple and an advanced search emerged as a connector between services. Not only users can search for services, but also users or user already deployed instances of their services. Then, according to privacy settings regarding visibility, users may or may not see, interact or find these services in the first place.

5.2.4 The Bookmark

As WebC-SaaS aims to be integrated in many different contexts, it also occurred the need to save some of these services as favorites. Since browsing repeatedly for the same services may be exhausting to users, the Bookmark feature enables this functionality. Either by doing a simple, advanced or just pure service browsing, it is possible to mark services as favorites.

5.2.5 The Follower

At this point, WebC-SaaS enabled users to search and mark services as favorites. However, users couldn't see the reflection of these actions. In other contexts, as Social Networks have learned, user participation is greatly motivated by showing the impact of their activity. Thus, searching, bookmarking and exploring is all measured and shown to users. In addition, it is possible to see who is bookmarking any given user, leading to the concept of Follower. Some modules emerged to profit from this features like Activity and Global Statistics and Customer Followers. Also, through the seller's perspective, these features play an important role in service receptivity and subscription extending.

5.2.6 The Profile

One of the most important concepts of WebC-SaaS is the Profile. In contrast with all other Domain Model concepts, the Profile captures all the user-interface interaction. Many of the WebC-SaaS features can be seen from different levels of depth: User; Service; Tab. With these levels of abstraction, users can benefit from many shared functionalities in these three different scopes.

6 Implementation

Following the traditional Webcomfort architecture, the WebC-SaaS infrastructure can be separated into Modules, Pages, Controls and Extenders that together represent the new functionalities and concepts. Next, in a summary, is a representation of the interaction between these components and followed by a deeper analysis of them.
During the development of this project, it was necessary to develop various WebComfort Modules to integrate the WebC-SaaS functionalities to the WebComfort platform. In the image below these modules are enumerated and organized in packages.

![Resource Relationship in the WebC-SaaS](image)

**Figure 3 - Resource Relationship in the WebC-SaaS**

### 6.1 Modules

To create and manage many of the concepts of this project, WebComfort Modules were not adequate, so WebComfort Pages were also employed to define workflows.

![List of WebC-SaaS developed modules](image)

**Figure 4 - List of WebC-SaaS developed modules.**

### 6.2 Pages

Without a specific context, WebC-SaaS is very abstract and unappealing to users, which made the testing experience harder. Nonetheless, since many WebComfort Applications are in development, it was possible to merge WebC-SaaS development with a real application context, overcoming the abstraction.
imposed by WebC-SaaS. Thus, two validation scenarios emerged: Integrating with WebComfort Pages and WebTrails.

### 7.1 WebComfort Pages

WebComfort Pages is a simple context that profits from WebComfort dynamic content creation. It allows users to create sites without any programming skills. In order to support this feature, the integration with WebC-SaaS should allow users to subscribe and manage this Site creation service.

This scenario is considerably relevant due to the fact that WebComfort Pages can benefit from every feature that WebC-SaaS provides. In fact, WebC-SaaS is the only framework installed over this WebComfort application. In WebComfort Pages, the WebC-SaaS is responsible for all the subscription process, that includes listing available services and previews and the whole subscription process.

### 7.2 WebTrails

WebTrails initiative offers solutions that, profiting from information technologies, promote turistic and areas of interest.

It offers visitors the appealing experience of the sightseeing and trail interpretation in natural parks and historic places. For organizations, that promote such turistic places, it provides tools that allow management and the promotion of contents. Such contents then can be used by Visitors on the same platform.

### 8 Conclusions

Developing a software product or web-based application is relatively easy. Making it a viable, profitable, and sustainable business is more challenging. The SaaS model presents itself as an ASP successor emending its inefficiencies and exploiting the outstanding opportunities of the Internet as a platform, with the goal of delivering services massively to innumerable users in any part of the world. But every software business model has its flaws, and SaaS is no exception. Considered an outsourcing solution and using the Internet as its only channel, SaaS model faces many challenges that traditional delivery models do not.

SaaS evolved and branched, many organizations approach SaaS from different ways: delivering service platforms; or the traditional single service providing such as CRM or E-commerce services. Predictions have foreseen a more explosive adoption and SaaS growing, but in reality, there are still only a few cases of enormous revenue making success, and now some skeptics began to announce the end of SaaS model. Many critics blame the intrinsic model flaws [9], others believe that approaches to explore this model's potential so far were less than effective [10].

During this work, it’s been proved that WebComfort is a robust and very extensible framework. Due to this, implementing the proposed WebC-SaaS infrastructure was made possible on this platform. Furthermore, it was possible to apply this framework onto real-context Applications such as WebComfort Pages and WebTrails.

As this work’s conclusion, it is probable that given the current technological advances and financial crisis, traditional software vendors will have to adapt to the new technological and business reality. The SaaS and ASP family of models propose alternative approaches to solve many traditional issues. For that reason, these models are feasible choices for both software vendors and customers to overcome their problems. On an optimistic note, this work’s author suggests the SaaS model has an especially favorable outlook for the near future.
References


3. Wei Sun, Xin Zhang, Chang Jie Guo, Pei Sun, Hui Su.: Software as a Service: Configuration and Customization Perspectives, in IEEE Congress on Services Part II (2008)


