INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

SaaS (Software as a Service) - Infrastructures and Applications in Real Scenarios

Pedro Guedes de Miranda

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente: Prof. José Manuel Nunes Salvador Tribolet, Departamento de Engenharia Informática (DEI)

Orientador: Prof. Alberto Manuel Rodrigues da Silva, Departamento de Engenharia Informática (DEI)

Vogal: Prof. André Vasconcelos, Departamento de Engenharia Informática (DEI)

Outubro de 2010

Acknowledgements

Many thanks to my professors at Universidade Tecnológica de Lisboa – Instituto Superior Técnico: Professor Alberto Silva, my thesis supervisor for offering valuable advice; to João Saraiva and David Ferreira for introducing me to the WebComfort platform and giving guidance on how to approach technical issues. Also, I'd like to thank SIQuant for the environment and conditions that supported me throughout this project.

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 reduce all installation, maintenance and upgrade costs that enlarge 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 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 over the WebComfort platform.

Keywords

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

Resumo

Os SaaS (Software as a Service têm crescido rapidamente. Acompanhados pela evolução da internet e pelo aumento dos recursos disponiveis, muitos acreditam que o modelo de distribuição tradicional de software pode estar em risco de se tornar obsoleto por software baseado na web, remotamente alojado que termina com todos os custos de instalação, manutenção e actualização que pesam em todos os orçamentos das organização da área. Várias empresas estão a fornecer novos serviços online, que oferecem aos seus clientes ferramentas para criarem os seus próprios serviços, totalmente configuraveis e sem necessitar de conhecimentos de programação. Este projecto pretende estudar e compara várias dessas iniciativas, e no final, propor um modelo que extenda as funcionalidades da plataforma WebComfort.

Palavras-Chave

Software as a Service (SaaS); Service Level Agreements (SLA); Internet; Negócio; Variabilidade; WebComfort; WebC-SaaS;

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 SaaS

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 provided 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 Information Systems was, and still is today, an enormous company budget challenge. [10] 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. During the 1990s, due to being cheaper and providing more flexibility to its customers, many organizations adopted this model, reducing business risks and saving time and money to focus on their main competencies. However, providing support for several different applications being run by each ASP datacenters is not an easy task. Each application requires qualified personnel to be maintained and upgraded, whose knowledge is not re-usable for most of the other applications. With a market as heterogeneous as it is, the support for these applications became more and more difficult to deliver, operating expenses grew and several ASP providers didn't managed to survive. Although, some struggled by narrowing down their focus on a particular market segment. [2]

1.1.1 Oportunities

Following the decline of the ASP model, the SaaS (Software As a Service) business model appeared as ASP's successor, to fix its disadvantages and covet its 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. [17] 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. Moreover, since the payment model is based on a monthly/annually fee like ASP, it allows for the clients to better assess the costs. On the other hand, from a software vendor's standpoint, SaaS has the attraction of providing stronger protection of intellectual property, resulting on an ongoing revenue stream to the proprietors of the software solutions. What's more, the evolution of web technologies and the conception of new web standards have recently led to the development of new Internet-delivered web-application concepts.

Software as a Service is still in ascension. SaaS is the new re-incarnation of service and application hosting that introduces a new flavor on how to deliver software and services to enterprises, and some companies are already well established providing such services. 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.

Then, it is not surprising to see that some SaaS vendors, besides providing services, adapted their platforms to service other companies, in order for them to provide their own services to their own customers. The revenue-making potential of these SaaS platforms is clear, since many of these SaaS platform providers receive royalty fees for each customer that their hosted third-party enterprise has. [7]

But 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.

Following the boom and settling of Web2.0, new doors opened for developers to expand the SaaS business model into the next level: In addition to deliver software as a service, some SaaS vendors also provide service creation and configuration without any tool or programming skill, everything through the web. There are already several initiatives of this service model, one of them is WebC-SaaS by SIQuant, in which this project will focus the most.

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]

[1] http://www.salesforce.com

1.1.2 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.

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 that wanted to have the data on site, or wanted to self-support their own architecture, or wanted to modify the software at their choosing.

Some brands felt that being tied to SaaS meant they were at the mercy of the vendor when it came to the software architecture. While this is certainly the case not just for SaaS but all closed-source software, when a vendor would upgrade a version, some brands were forced to accept the changes. In some cases, some brands were not aware/prepared of some upcoming upgrades to the SaaS software and were blindsided to the changes.

Furthermore, these brands felt that the larger customers who wanted specific feature upgrades were able to influence the vendor through direct pressure or even fueling the work through feature requests. Since the software is delivered over the web, there was little recourse for the smaller brand to deny the changes.

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. [10] Having financial and client data hosted off-premises include the following risks of data leak or loss that stand tall between SaaS and Customer acceptance:

- SaaS operator's failure;
- ISP's failure;
- Hackers;
- Virus and Trojan horses;
- Natural Disasters;
- Sabotage;

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.

1.2 Context

This thesis subject was proposed by Professor Alberto Manuel Rodrigues da Silva, which aims to study the current state of SaaS applications in today's market, and to develop a framework of comparison that reveals qualities and flaws of these SaaS examples. As a result, this proposal comprehended the possibility of improving the WebComfort platform by implementing the SaaS model over that framework. Following this proposal, an internship in the company SIQuant was suggested, which allowed to better focus in the study and implementation of this work.

1.2.1 WebComfort

The WebComfort platform is a Web Portal and Content Management System (CMS) Framework promoted by SIQuant. It delivers contents to its users through any device with a common browser (e.g. IE or Firefox) over the Internet. The platform is developed using Microsoft's ASP.NET 2.0 technology (C#). WebComfort claims the separation between the content and its presentation. Content is presented through information modules, whose presentation can be configured without changing the data model and the underlining contents. In particular, regarding content presentation, a layout management mechanism is provided, at portal, tab and module levels. Furthermore, WebComfort supports authorization and security policy management through a flexible role-based mechanism.

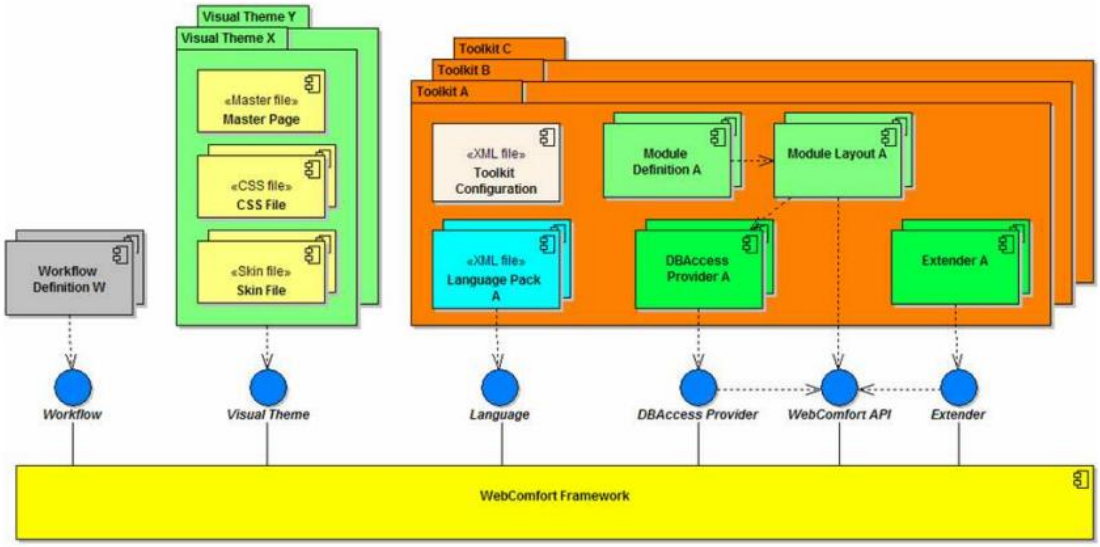


Figure 1 - WebComfort Extensibility

The “Framework” designation comes from WebComfort’s easiness of extension, by allowing adding new module types to manage and display existent information, or even new types of information, supporting the development of new module logic and design through a well-defined module API. Modules are one of the elementary and most important components of the platform. They are responsible for embedding content and conferring functionality in the WebComfort application. They can communicate and share functionality between each other and can be integrated in WebComfort Categories and Toolkits for easier deployment. WebComfort Categories are groups of module types defined at a portal level, which allow roles of users to add/edit/remove modules – and therefore content – to the portal. On the other hand, WebComfort Toolkits group module types at an applicacional level, in order to allow installation of different module types in WebComfort instances. Other WebComfort platform features include: visual themes support; multilanguage support; application extension support; workflow management support; data repository access; integration with Microsoft WebParts, among other things.

1.2.2 WebC-SaaS

Until now, the WebComfort platform only supported the hosting of WebComfort Applications, over instances that were manually deployed and maintained. Application Modules would have to be installed on the instance manually, in order to allow the usage of its functionalities. The WebC-SaaS project improves this subject, by implementing an automatic deployment mechanism capable of deploying and maintaining WebComfort Applications during a subscription period.

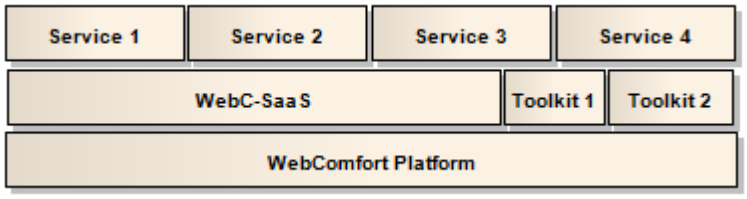


Figure 2 - WebComfort and WebC-SaaS Relationship

1.3 Objectives

This project aims to analyze and discuss several SaaS initiatives in the market, as well as apply the resulting study on further development and extention of WebC-SaaS platform. To better understand these goals, both theoretical and practical objectives are presented in further detail.

1.3.1 Theoretical Goals

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. From this study results a framework that compares the most relevant services studied, that focus its attention on the following parameters:

- Design Concepts - Introduction of concepts behind the product, defines the ideas of service, page or site for that company;
- Configuration and Content Management - How the configuration is made, How the content is managed;
- Usability - Measures the balance between elegance and clarity with which the interaction with the service is designed.
- Features - The package of functionalities that services provide, such as custom domain name, multi-language support or roles access management;
- Service Levels and Subscription models - Studies the way companies incentive customers to subscribe their products;
- Support, Backup and Recovery - Evaluates companies' mechanisms that ensure customer's data;

1.3.2 Practical Goals

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. In the late stage of this project, it is also expected to integrate WebC-SaaS with some WebComfort Applications.

1.4 Document Structure

This document is divided into six chapters.

Chapter 1 (Introduction) - This chapter introduces the key concepts, the context, the problems and main goals of this project's study.

Chapter 2 (State of the Art) - This chapter brings an analysis over some existing SaaS models compared by a framework that captures qualities and flaws of these models.

Chapter 3 (Conception) - The Conception of this project is described in this chapter. It captures and explains all the design decisions that are presented in the Domain Model.

Chapter 4 (Implementation) - In this section, it is explained how the concepts were implemented, and how requirements were achieved.

Chapter 5 (Validation) - The validation chapter shows how this project interacts with real contexts.

Chapter 6 (Conclusion) - In this chapter a conclusion is made, along with some future work suggestions.

2 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 infra-structures 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.

2.1 Concepts

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

2.1.1 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.

Through SaaS vendor's perspective, it is favorable to opt a single instance solution, which is easier to control, with less resource consumption and cheaper to develop. However, through customer's perspective, the potential to configure and adapt the service is somewhat limited. Many of these single instance services force the same structure and layout to all pages. Thus, clients can only explore content modules to achieve their requirements which narrows down the service to a simple web-site provider.

In a multiple instance site, customers have the responsibility to maintain each page structure and organize content separately. Since each page is a different instance, module position and presentation may vary from page to page, which further extends the time required to develop the service.

2.1.2 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.

2.1.3 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.

From a business perspective, modules allow the SaaS vendor to divide functionalities over a set of different service levels. Modules can hold endless functionalities that include: HTML content; a calendar; an e-mail interface; user management; inventory management.

2.1.4 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.

2.1.5 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.

Fortunately, resources, processing power and bandwidths increase exponentially, and so does interface complexity and functionalities. However, even with such resources available, SaaS vendors need to balance usability and interface complexity as the number of functionalities grows.

2.1.6 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.

2.1.7 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. There are several factors that influence service level design, the most common include:

Resource Usage - Service Levels are designed and separated based on resource usage, such as bandwidth, storage space used, offering more resources to customers who subscribed more advanced service levels.

Functionalities - Functionalities offered by the SaaS vendor are ordered and categorized according to its value and business potential. According to that order, vendors decide which features fit best in their service level hierarchy.

Payment - Some vendors choose to provide all resources and features in every service level, but creating different payment options for each service levels. Some levels may have a start-up cost and a monthly fee, when other levels may be free of start-up investment but the monthly fee is substantially bigger.

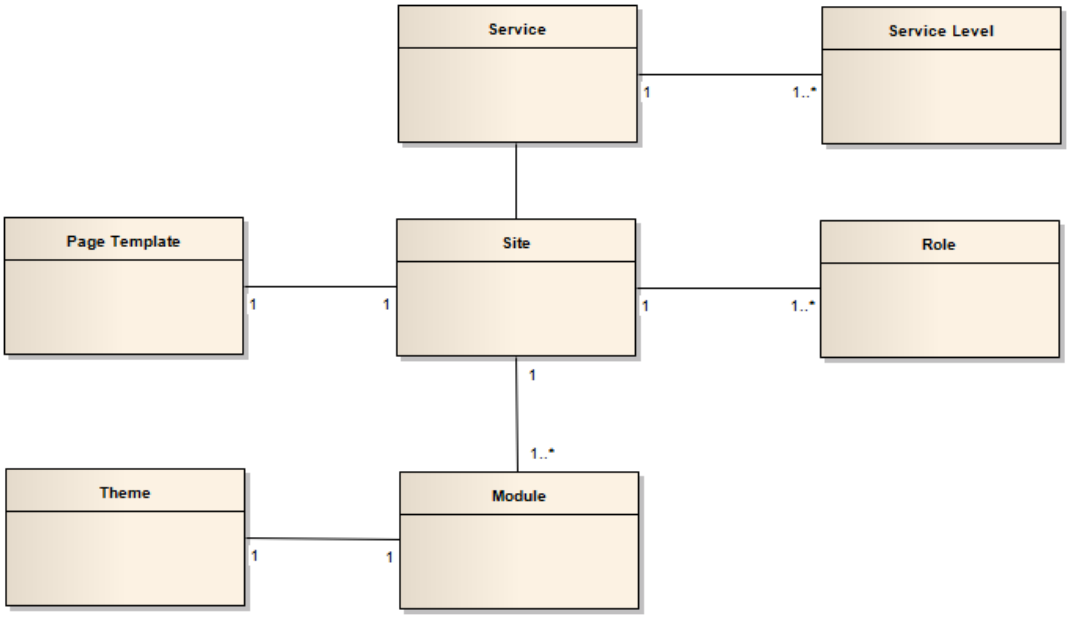


Figure 3 - Relationship of Concepts.

2.2 Features

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

2.2.1 Domain Name

Domain Name is the service entry point for clients. Generally, there are three different approaches to achieve this:

Non-Mapped Domain Name - In this situation, customer's services can be accessed by the same indexes that the SaaS vendor uses to identify them (example: www.saasvendor.com/sites?id=12345&user=12345). Usually, these vendors provide centralized authentication on their portal page that redirects the customer to its service.

Mapped Domain Name - SaaS vendors let customers to choose an unique name that will be associated with their services (example: www.saasvendor.com/customerservice). This helps customers to advertise their service and skip visiting the vendor page every single time they want to use their service.

Own Domain Name - From the business point of view, having their own domain name is essential for brand recognition (example: www.customerservice.com). Some companies wouldn't feel comfortable paying for a service that constantly advertises the SaaS vendor by forcing the service to be associated to the vendor domain.

2.2.2 Multi Language

As in every business, an enterprise must benefit the most from the advantages that its business model offers, and should also compensate for its flaws. One of SaaS advantages is its ease of raising new clients. Through the internet, almost any person in the world is eligible to use the services that a SaaS vendor provides, but that person can only be considered a potential client if the service is available in a language that he recognizes. Furthermore, 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.

2.2.3 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.

2.2.4 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.

2.2.5 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.

2.2.6 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.

To benefit from previous user experiences, designers create interfaces using metaphors based on such applications. The most common interface metaphors to configure a page are:

- Office Application;
- Internet Portal;
- Folder Tabs;

2.3 SaaS Platforms Survey

Although several more initiatives were subject of analysis, this project will focus on the following five services: Google Sites; Moogo.com; Site2You; WebC-SaaS; and Webnode.com. In this section, these services will be described in detail due to their unique characteristics and differences.

2.3.1 Google Sites

Google also has its own version of collaborative web sites[2].

Design Concepts. Google Site's modules inherits from other Google products, such as Google Calendar or Maps, with the exception of the content module, which holds customer's HTML code. Each Google Sites page is a separate instance, which stimulates the creation of a site hierarchies where every site is different from the other. However, each site structure obeys to a template that is common to every page.

Configuration and Content Management. Site configuration is supported by a web toolbar. Similar to a common text editor application, this toolbar provides features such as font size and style changing, text alignment and table management. This simple toolbar is a site oriented solution for HTML content in the page, but other module configurations are tuned individually in each module's settings panel.

Some content management approaches try to generalize module parameters, but in Google Sites, in order to capture all the features of other Google products, each module contains an extensive group of settings that cannot be shared between them due to their unique functionalities.

Usability. Elaborated modules requires a vast support documentation in order to develop a single page. Each feature has its own knobs, forcing the user to explore each module individually.

Features. Google Sites platform and tools are presented according to user's main language, even the extensive documentation is all translated. Despite this major effort, Google Sites doesn't support Multi-Language content. In order to present information in several languages, users need to create copies of their own site, changing every content manually to the desired language, with all the extra maintenance effort involved.

Customer's pages can be easily accessed, each page has an unique name that is used to map its address under the sites.google.com domain (egg: 'sites.google.com/site/pagetestingsq/'). Although Google Sites has a simple access control system that allows users to define permissions, it cannot be considered a mature Roles Access Management system. Users may only set their pages as public, private or shared with Google Mail friends, but they can't create their own user groups, or specify different page parts for different users.

Another inheritance from Google's catalog is Google Analytics. It offers full statistic reports about the site that goes far beyond the simple resource management information that the most companies offer. It offers useful information about their sites and what should be improved in order to improve customer's business.

Service Levels and Subscription Models. This service is available in only two service levels. Google is proud to provide a free base service that includes all existing

[2] http://sites.google.com

features without any adds or fees. For customers with bigger resource needs, Google offers 10Gb of storage for Google Apps subscribers.

In a short summary, Google Site's competitive advantages are deeply related to all other Google products and their integration with Sites. Some believe that one of the most relevant factors in SaaS adoption is brand's name[6], and since Google is a world-wide well-known company with millions of users, Google Sites has a huge advantage in this market segment.

Support, Backup and Recovery. As Google Sites provides an extensive integration with other Google products, content management is not an easy task. Fortunately, Google Sites presents an easy-to-use support documentation, that demonstrates all modules configuration with tutorials, videos and frequently asked questions.

In case of data loss or human error, if a page is deleted or for any reason content is lost, Google gives no guarantee of retrieving the lost data.

2.3.2 Moogo.com

Moogo.com provides web 2.0 site creation without any design or programming experience. It also provides several tools to retrieve extra statistical information about sites[3].

Design Concepts. Moogo.com service is based on the single instance page architecture, which structure is ruled by a template selected upon subscription. Modules are chosen along the site creation path, and they are visible in every page. It is possible to move modules within the page. Similar to other services alike, the addition of HTML content is restricted to content modules designed for this purpose.

Configuration and Content Management. Moogle.com presents a toolbar metaphor over the site, in order to edit all site's content. It Subdivides configuration into four folder tabs. In the Content Edition tab, through a drag-and-drop process, customers can arrange modules positions. In the Site Management tab, all content can be edited, whether presentation or module related. Preferences tab allow customers to monitor resource usage, protect the page with password or change personal access data. Confined into Additional Services, Moogo.com grouped all the extra features from the upper service levels.

The customer may edit each selected module but may not remove them. In summary, Moogle.com dynamic creation proposal is based on the edition of module parameters and adding new content pages. It may be too restrictive not being able to remove modules, as well as to add new instances of them into the page.

Usability. Moogle.com offers a simple option package that narrows configuration limits but makes it extremely easy to use.

---

[3] http://www.moogo.com

Features. Customer's page is mapped as a sub domain of Moogo.com, and it also allows customers to have their own custom domain name (egg: page.moogo.com). Language support is only available in English and neither tools or platform can be viewed in other languages. In addition, Moogo.com doesn't support multi-language content definition.

Although Moogo.com doesn't support Roles Access Management, it gives great emphasis to statistical data. Visitor statistics allow you to follow the number of visitor to customer's site from a number of perspectives. The daily and monthly number of visitors and page uploads are carefully presented in charts. You can also see which pages people have come to the site from, or which search words were used to find the page with a search engine. A simple version of the Statistics is included in the start-up features.

Service Levels and Subscription models. Both feature and resource distribution approaches were selected by Moogle.com to define their service levels. The basic level includes 200Mb of storage and most of the low business power features, mostly presentation oriented or personal site creation, like photo album or a blog. The second level, amplifies storage space, provides a custom domain name and withdraws publicity out of the site. The business level, offers a search engine, news module and a mailing list.

In summary, Moogle.com divides their functionalities through their service levels and tries to obtain revenues from more interesting modules in the paid subscription levels.

Support, Backup and Recovery. Support documentation is organized into a list of instructions that describe all existing features. Each module has an entry page with several images and tutorials that explain the most frequent tasks.

Moogo.com doesn't support backups nor previous version retrieving.

2.3.3 Site2you.com

Site2You.com offers services to create and maintain web sites with minimum developing knowledge. This company strategizes on high quality pre-defined templates for several types of business, so that with only a few clicks, the site may be up and running[4].

Design Concepts. With such powerful template designs, it is no surprise to see the single instance site approach. Site2You.com has struggled to generalize most of the modules, making their configuration much easier as they share almost every characteristics. The only re-usable unit of the service is the content module, any other module can only be instanced once or toggled off.

[4] http://www.site2you.com

Configuration and Content Management. Page configuration is organized in three different folder tabs, that hold site oriented edition settings. In Page presentation, a drag-and-drop system helps positioning page modules from page section to other page sections. Content tab allows editing at module content level. After selecting the desired module, available option for that module are presented. Finally, Advanced Editing controls which modules are active.

Usability. Generic modules, with very simple settings. Suitable for inexperienced users.

Features . Like most of these services on the market, Site2You.com links customer's page to an unique name for easy access, mapped into the provider's domain. A custom domain name is also available for an additional fee.

Site2You.com is presented in English, without any other language option. Usually services that didn't consider platform multi-language, also don't offer multiple language content definition or translated documentation. Site2You.com is no exception.

In terms of Roles Access Management, Site2you doesn't offer any control mechanism of user differentiation or protecting content. Offering unlimited storage space and no bandwidth limits, Site2you.com isn't forced to support statistical data.

Service Levels and Subscription models. All features are available in both service levels, which were designed based on payment options. Premium Service comes with all features but with no startup fee. The Anti-Crisis service offers a reduced monthly fee but requires a startup cost at subscription time.

Since Site2you.com doesn't offer a basic service, a seven-day trial is offered. This company tries to incentivize customers based on different payment models and their unlimited resource usage.

Support, Backup and Recovery. This service offers a Help Center with extensive details, including live chat support with administrators and many support topics also have demonstration videos. Each client has three dedicated hours of administration work. In case of any doubt on site customization, an administrator may execute all the modifications the customer requires.

2.3.4 WebC-SaaS

WebC-SaaS offers services based on web 2.0 configurable pages. [11]

Design Concepts. In the WebC-SaaS approach, each page is an independent instance. Page layout is pre-defined in two containers where modules can be inserted. It offers an extensive module list where customers can select which ones are shown in each page, without number or position restrictions. It is possible to apply different visual themes, but it isn't possible to change page structures.

Configuration and Content Management. When a client subscribes a service, it has access to his personal area where pages can be created. Page is sub-divided in static

containers, where modules are presented. Each client may add any number or type of modules they wish in the Tab Edition section.

WebC-SaaS configuration approach is module oriented. Every module is encapsulated in a configuration panel that shows every option available for that module type. These options include content edition, which allows for multi language content, add new entries, edit entries or delete existing ones.

Features. WebC-SaaS provides a centralized authentication system through its portal, which restricts access to registered users in order to access their services. It doesn't allow to create custom domain names or mapped ones. All customers must pass through the portal.

Configuration and platform are available in multiple languages, easily expansible to new languages if required. WebC-SaaS allows the creation of pages with content translated in several languages. Thus, each client can maintain two versions of each page with little effort.

One of its most precious features is Roles Access Management. It allows customers to define different roles for different users, and then apply those definitions to different parts of the service, controlling the access to these.

As for statistical data, WebC-SaaS lacks the structure and mechanisms to control resource or bandwidth usage.

Service Levels and Subscription models. WebC-SaaS is a platform, which allows creating service levels, associated to module toolkits and is ready to make restrictions on these modules based on settings, but these features are not yet implemented. It also allows to create free services, with variable subscription periods and prices.

Support, Backup and Recovery. WebC-SaaS works as plug-in on its mother platform, WebComfort. Neither WebC-SaaS or its platform provide mechanisms to manage and create automatic backups, version retrieval or customer's services. Furthermore, it lacks a support section that explains all functionalities.

2.3.5 Webnode.com

Webnode.com offers a configurable, web 2.0 pages service[5].

Design Concepts. According to Webnode.com approach, the chosen modules are static, common to every page and may only exist a single instance of them per page. However, customers can add limitless content pages to their pages with their custom HTML content.

In the end, Webnode.com works like a portal, in which the page concept is simulated by the substitution of the front section of the site for another content module, everything

[5] http://www.webnode.com

else is static. Thus, it fits in the single instance site model. Page structure is static and template don't change page section where modules can be held, it only changes the visual theme applied to the entire site.

Configuration and Content Management. Page configuration is made through a toolbar on top of the page, which shows all available edition options applied to the site. It also features a drag-and-drop system, constantly available during configuration, that allows changing page layout.

Usability. Module generalization provides option's centralization into a toolbar, that makes easier to use and to understand.

Features. Webnode.com links each page to an unique name under the company's domain name (egg: pagetesting.webnode.com), but it also offers the option to use customer's own domain.

All contents are adapted to a vast set of languages, automatically detecting and showing contents in the user main language. Besides a multi language platform and tools, it allows users to create multi-language content versions of their sites, but only in a premium service level.

The platform provides information about resource and bandwidth usage. In addition, their offer a small Access Management feature: the possibility to protect services with password.

There are companies with well defined networks and strict policies about internet exposure. At start, this market sector wouldn't be appropriate to benefit from SaaS, but with the possibility to download a copy of the service, it may be possible to use a service on-premises, without worrying about risks that Internet involves. Webnode.com offers this feature, while in a business perspective this may be interesting, this feature steps away from the SaaS model.

Service Levels and Subscription models. All features are scattered through four different service levels, that are different in number of functionalities and resource limits.

Webnode.com considered fundamental their features of offline version, multi-language content and premium support. Services were designed based on the market availability of these features, since only a few service providers of this kind offer an offline version or multi language content definition. As a result, these features are placed in the higher service levels, and it is no surprise to see that offline version is the only feature that the most expensive service level adds, because customer's fear of storing their data off-premises is well known.

Support, Backup and Recovery. Webnode.com maintains a free support page, which has a frequently asked question page and a discussion forum. Even if page configuration is somewhat intuitive, an additional structured documentation would help on the page creation process. As an incentive to subscribe higher service levels, the premium support is only available in paid subscription levels.

2.4 Discussion

Every service provider is unique, with its own business model, goals and with its own internal company structure. These factors influence every product that company develops, leading to an unique service as well. However, all these service providers use the same distribution model , meaning that not every design or business decision may result in a quality in the final product. Moreover, it may even be considered a flaw in this distribution model. In this section, a discussion is made based on these differences and their business impact.

	Site	Themes	Page Templates
Google Sites	Multiple	✓	✓
Moogo	Single	×	✓
Site2You	Single	×	✓
WebC-SaaS	Multiple	✓	×
Webnode	Single	×	✓

Table 1 - Concepts Comparison.

Google Sites

Pros

Powerful brand name. When enterprises ponder over the several available services in the market, Google is always in advantage. Through the years, Google has managed to associate quality and performance to its image, such image that is recognized all over the world. Thus, every Google product is considered reliable and refined.

Integration with all other Google products. Google Sites greatly benefits from all other Google's projects. Many companies have to design modules from scratch to provide in their services, spending more time and resources developing these rather than developing the platform itself. Google Sites worked with these products interfaces and adapted them to be used in client's pages. As an example, a user can have a Google Docs document and use his Google Site page to publish it.

Independent, specific modules. Each Google Sites module has a vast set of specific configurations, exponentially increasing configuration limits.

Effective Service Level distribution. Google Sites has simple and effective service level design. The only difference between both levels is storage space, and taken into careful consideration, is may be a successful strategy. Any personal-use site, rarely exceeds 100Mb of data, while if an enterprise whishes to host their own revenue

making machine, it will soon require more space to store their data, which leads them to Google Apps subscription level.

Cons

Slow development cycle. With less common settings between features, with modules so distinct as Maps, Docs or Calendar, users face the heavy task to explore support documentation in order to control every module. Usually, great configuration brings means great user effort.

	Approach	Interface
Google Sites	Module	Office toolbar
Moogo	Site	Tabs
Site2You	Site	Tabs
WebC-SaaS	Module	Tabs
Webnode	Site	Office toolbar

Table 2 - Configuration

Moogo

Pros

Free Statistical Data. Moogo.com offers, in all service levels, a free and extensive statistical data support with similar features of Google Analytics service. Such feature gives information back to the users about what type of client is visiting the site, what pages have been visited and which sections of the site they have visited. This information is very useful for customers, so they can refine their business and configure the service in order to build a better a product for their clients.

Cons

Risky Service Level design. Defining Service levels based purely on functionalities may be a risky approach. Since many vendors offer the same features for free, and define business strategies based on different parameters such as custom support, resources or versatile payment models, Moogo.com may lose position towards other competitors.

Site2You

Pros

Unlimited resources. Site2You offers unlimited storage space and bandwidth usage. It is fairly common to see service levels design based on resources, and it isn't rare to

find such vendors that even in their ultimate premium service levels, still have resource limitations. When a customer is choosing an SaaS service to store their data and if there is no service level that satisfies customer's resource requirements, that service is immediately ruled out as an option. This leaves Site2You as the only available option for some potential clients.

Cons

Short Configuration. Excessive module generalization restricts configuration. It may be easier and simple to configure a site which offers less options, but is clear that the service may never be suitable for some customers, as their requirements don't fit in Site2You's almost-static structure. Clients are forced to experiment every business template in order to find the suitable modules, and it is well likely that the 7-day free trial expires before finding the right template.

	Domain Name	Multi-Language Platform	Multi-Language Content	RAM	Statistical Analysis
Google Sites	Mapped	✓	×	×	✓
Moogo	Mapped + Own	×	×	×	✓
Site2You	Mapped + Own	×	×	×	✓
WebC-SaaS	Non-Mapped	✓	✓	✓	×
Webnode	Mapped + Own	✓	✓	✓	✓

Table 3 - Features

WebC-SaaS

Pros

Powerful Roles Access Management. WebC-SaaS features a powerful Roles Access Management system. It goes beyond the simple password-protection service or changing contents from public to private. WebC-SaaS' RAM allows roles definition, association between user groups and roles and link these roles to specific contents of the service. This combination allows customers to create their own client's hierarchy, pages hierarchy and a dynamic content control based on user type.

Multi-Language Platform and Content. Another great model benefit inherited from SaaS is the Internet as the main channel. In other words, the Internet expands business range overseas with no additional costs. But in order to be an advantage, SaaS vendors should adapt platforms to support multi-language content. WebC-SaaS supports full platform language support, with multi-language content definition.

Cons

Lack of Domain names. Due to usability, publicity and business reasons, there must be an easy way to rapidly access the service. Not every customer wants his service password-protected for personal use. As such, forcing a user to authenticate in order to access the service will deprive customers from publishing publicly their contents. This platform doesn't support the use of custom sub-domain names to identify services.

No resource monitor system. This platform is unable to monitor and manage resources used by its customers. Due to performance and security issues, it is essential that al stored data can be controlled and managed. Furthermore, this platform should also have mechanisms to retrieve statistical information about hosted sites to help customers improving their services.

Long Registration Process. As WebC-SaaS works as a plug-in on its mother platform WebComfort, and since its platform also requires user authentication, WebC-SaaS forces customers to manually register and login in the WebComfort system in order to subscribe a product, extending the registration process more than the regular customer can withstand.

	Design Approaches
Google Sites	Resource
Moogo	Resource + Functionalities
Site2You	Payment
WebC-SaaS	Functionalities
Webnode	Resource + Functionalities

Table 4 - Service Levels

Webnode.com

Pros

Offline Version. Almost every organization can profit by exposing their network and their workers to the internet. Although, some conservative companies prefer to stay isolated for several reasons. At first glance, these companies may look to be out of the SaaS market range, but some SaaS vendors have managed to reach them by creating

an Offline Version of their service, which can be downloaded and used in a private network. However, this approach is a sneaky way to diverge from the SaaS model's difficulties. It delivers a copy through the traditional model, instead of providing safety solutions to convince clients to use the SaaS model.

Premium Support. Back, Data retrieval and Support increases confidence in the SaaS vendor, giving the idea that customers data won't be lost, and in case of emergency, there are mechanisms to prevent catastrophes.

Cons

Static layout. All pages have the same layout. It may be easier to navigate but not every customer prefers the web portal style for his site, neither this view is suitable for all business models.

3 Conception Issues

To understand the design concepts that support this project, next are described the fundamental classes and relations of this project. 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 and the Extensions are shown along with its main concepts.

3.1 WebC-SaaS Actors and Use Cases

To better understand the conception of this project, it is important to know the entities that interact with the WebC-SaaS platform. Thus, next are presented the identified actors and main use cases of the WebC-SaaS platform:

3.1.1 Actors

Behind the SaaS model lies an implicit hierarchy of key actors that support it. As such, identifying these actors was the first step developing this project. Following is a summary of the identified actors and their hierarchy:

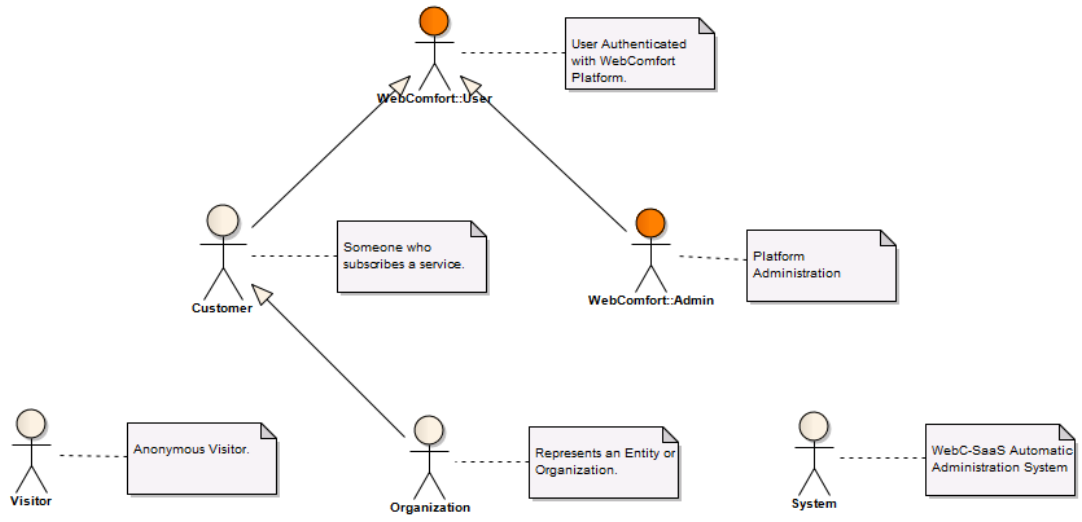


Figure 4 - Actors and Relationships.

System – This entity represents the system administration and is responsible for maintaining the application environment coherent. It is responsible among other things

to detect users that exceed the service level resources or which subscription has expired and requires termination.

Visitor – The Visitor actor represents a person which might be a WebComfort user or a SaaS user, but is not yet authenticated in the system. This user is only allowed to register/authenticate and to perform other basic interactions with the system.

Customer – This actor represents the service subscriber. It is the service administrator for a particular application instance. It can access and configure the system's services, as well as control possible subordinate users and their privileges.

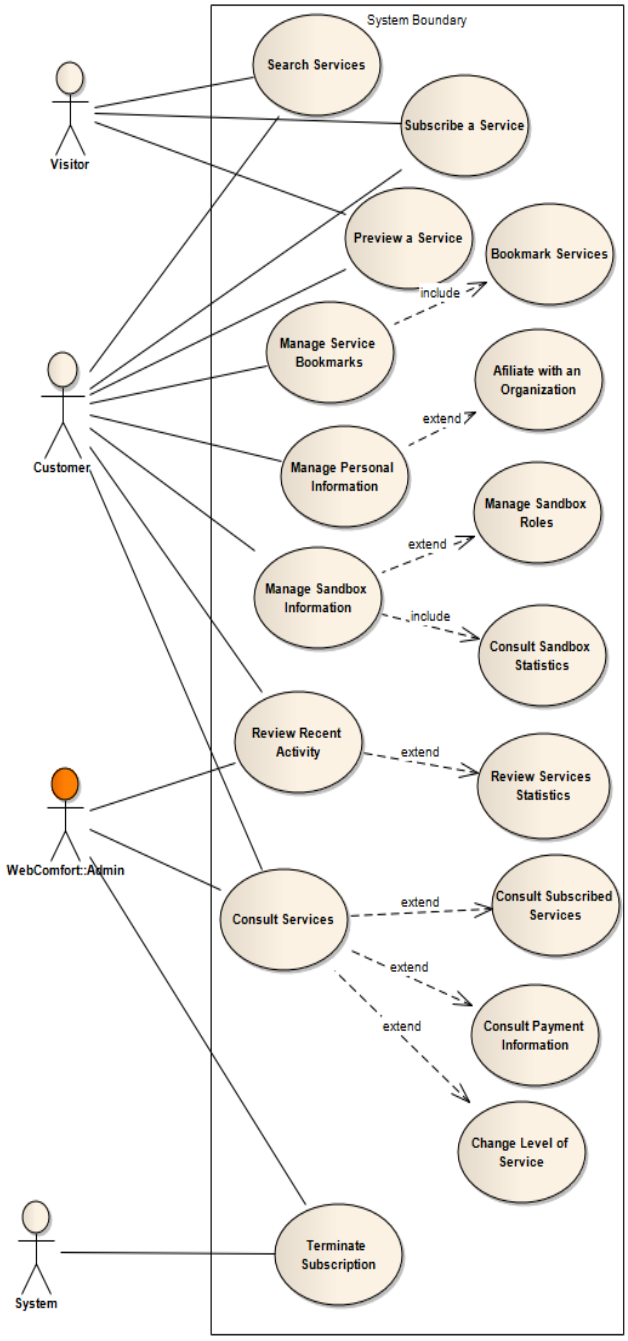
Organization User – The Organization User actor represents the subordinate users of an organization. These users can only access a subgroup of the services the Organization has subscribed. Note that Organization Admins also exist to administer Organization service subscriptions, but in this context are considered to function as a Customer actor.

WebComfort User – This actor corresponds to a generic WebComfort user. It cannot access SaaS services until it registers as a Customer. After registering, an authenticated WebComfort User is considered as a Customer.

Admin – This actor represents a WebComfort administrator. Besides being able to administer the WebComfort platform and its intricacies, the admin is responsible for creating Template Services and defining its details and Levels of Service (e.g. SLAs, pricing, description, etc.).

3.1.2 Use Cases

Various Use Cases were identified in this work. In order to better comprehend the use cases, these were grouped by similar functionality. Below, are summarized some of the most relevant use cases:



Manage Personal Information - Customers can consult, manage, store and edit personal information. They can also set privacy settings or affiliate with an organization.

Manage Sandbox Information - Customers can personalize their service host space. They can add a logo and a description, set roles or privacy settings. In addition, customers can consult resources, users that have recently access their service or added this service to their bookmarks.

Consult Services - Customers and Admins can review subscription information about services, information about payment and which services have been subscribed.

Review Recent Activity - Customers and Admins can consult statistical information about the platform. They can view, among other things, the most active users or services on the platform.

Search Services- Either Visitors, Customers or Administrator may search for services. They can do a quick search or a more refined one, finding services or users on the platform depending on privacy settings.

Manage Bookmarks - Customers can mark other user's services as favorites through a bookmark system. They can add or remove services from bookmarks and check who has bookmarks on any service they have subscribed (concept of Follower).

Review Service Statistics - Customers can consult information about Service's performance. Users can see in a user-friendly way the visitors and activity of their services.

Subscribe, Terminate or Change Level of Service - At any given time, a User must be able to subscribe to one or more services at a time. With multiple or single subscriptions active, Customers can terminate the subscription or change the level of Service. For all these Purposes, customers are presented with the Service Level Agreement and must fully comply before subscribing.

3.2 Goals and Objectives

Before uncovering any conception details, next are described the goals that have driven this project's design. Goals are presented as a collection of objectives with the respective requisites of the WebC-SaaS framework:

Goal 1 - Provide WebComfort Applications as Services.

Objective 1 - Allow full compatibility with existing WebComfort Applications.

1.1 - Integrate WebC-SaaS with the WebComfort platform, without interfeering with existing deployed applications;

1.2 - WebC-SaaS should operate independently of it's platform (WebComfort) or context;

1.3 - Provide mechanisms that allow Applications to create new functionalities while maintaining independency of WebC-SaaS;

Objective 2 - Deploy several services in a single instance.

2.1 - Allow various service types to use the same WebComfort Module Definitions;

2.2 - Differentiate Module functionality, depending on the level of service that is subscribed for a type of service;

2.3 - Provide mutliple subscriptions for any given service;

Goal 2 - Enhance the Web Application experience.

Objective 3 - Capture service consumers.

3.1 - Provide a mechanism for users to be able to preview services without having to subscribe to it;

3.2 - Create a mechanism for users to easily explore existing services;

3.3 - Allow users to easily refer to services;

Objective 4 - Provide subscription alternatives.

4.1 - Allow users to subscribe several services from different types.

4.2 - Allow users to subscribe services without interacting with the underlying platform WebComfort.

4.3 - Allow users to cancel a service subscription.

Goal 3 - Promote user contribution and interaction between services.

Objective 5 - Provide user control

5.1 - Provide user-confined spaces where services can be deployed and used;

5.2 - Allow for service configuration of all user's services, reusing WebComfort concepts like Roles, Tabs and Visual Themes.

5.3 - Allow users to configure personal information aswell as privacy settings;

Objective 6 - Promote service discovery

6.1 - Provide tools that allow users to find new services on the platform.

6.2 - Allow users to visit and interact with other user/organization services.

6.3 - Provide a bookmark system that easily allows users to access to their favorite services.

Objective 7 - Create service measurement metrics

7.1 - Allow users to gather information about all platform services, according to popularity, total visitors or recent activity.

7.2 - Users should be able to study the performance of their active services, using charts and other statistical information.

7.3 - Allow users to manage service resources and imported data.

3.3 Challenges

The WebComfort-SaaS model proposes dramatic changes over the normal procedures of application deployment over the WebComfort Platform. Typically, each WebComfort instance would have to be manually installed, by creating and managing each Application Module to meet functionality and layout requirements. WebC-SaaS provides automatic deployment of such services by the use of service templates. To acomplish this, several WebComfort limitations had to be overcome. Next are presented some of the challenges encountered during the development.

Multi-tenancy

One of the main pillars of the SaaS model is the multi-tenant theory, which provides applications to be accessed by different tenants on a single instance, at the same time. To accomplish this, SaaS applications implement multi-tenancy, which produce independent contexts of execution (virtual application instances) for each different client. Unfortunatly, many of the grounding rules of the WebC-SaaS environment are already defined by its platform (WebComfort), and as such, the multi-tenancy had to be emulated through a new concept: the Sandbox. And so, a sandbox would be the user-confied space where the service is deployed.

The Sandbox concept has a central role in the WebC-SaaS and is tied with security features provided by WebComfort, to ensure that customers have their services protected agaisnt illicit exploitation of private resources or data vandalism. This means that each customer has the privilege to administrate its own sandbox, but only that sandbox. However, according to privacy settings, the customer may allow users to view their contents, promoting interaction.

Resource Management

One of the key aspects to ensure platform stability is the resource management. Users can submit and upload an endless amount of data that can compromise the system's performance, so it was crucial to develop a resource system that could monitorize and limit resources. Since WebComfort-based Applications don't follow any pattern or rule accessing the disk or database, resources can be scattered among folders and database tables. This makes it hard to associate resources with applications or services. In order to overcome these problems, an API would have to be fully integrated with the WebComfort platform and force Applications to use this new system. With many Applications already developed over WebComfort, this would mean that all of them would have to be re-designed to meet these new requirements. Even with these problems, the WebC-SaaS platform still manages to associate resources to the user that submited them, providing tools to manage these files at a user level.

SaaS and Social Networks

Both the SaaS model and Social Networks feature the most recent tools of the Web 2.0. On one hand, the SaaS model uses this feature to dynamically create and maintain a multi-tenant, multi-version, multi-service application through the browser. On the other hand, Social Networks have been using the same technology to make users interact and contribute. This project introduces some of these interaction features into the SaaS model with the intent of creating a more appealing and interactive experience than the traditional SaaS approach of an isolated user experience. Struggling to have a steady performance, only some of these features have been integrated with the WebC-SaaS platform.

Integration with WebComfort

Because this project was to be developed as a WebComfort Toolkit, it was necessary to maintain WebComfort Applications' independence from it. For this reason, various approaches were necessary to be made, in order to maintain WebComfort independence from the WebC-SaaS Toolkit.To separate the two namespaces, a WebComfort Extender was created to alter and complement the WebComfort platform functionality. This concept allowed, among other things, to handle platform and WebComfort events, useful in monitoring and altering the behavior of various objects, while maintaining independence between projects. Despite the efforts to minimize instalation time, WebC-SaaS still requires that its modules have to be installed one by one into WebComfort platform.

Context Abstraction

While some service platforms on the market have an extensive list of features and functionalities, the main goal of WebC-SaaS lies with its integration with other applications. Thus, implementation guideance shouldn't be directed towards any specific context. This goal implies that all features provided by this framework should be applied to the majority of applications. Many features were re-designed to acomplish this, but unfortunatly some didn't fit the required abstraction, thus being removed.

In addition, further along into the framework testing and integration phase, it has been observed that WebC-SaaS and some of the existing WebComfort Applications had overlapped features. As WebComfort is an abstract CMS, many developers had to repeatedly implement the same features for each Application. Even if this makes integration harder for existing Applications, it shows that WebC-SaaS meets it's level of abstraction and may be used as a starting ground for future contexts.

Moving to the Cloud

As Several SaaS providers have successfully explored, using the Cloud to store Services brings great advantage either to Customers and to the Providers. By migrating resources and processing, a Single-Machine, Multi-Tenant Application

performance is no longer the main concern. However, since WebC-SaaS is built over the WebComfort Platform, it wasn't possible to migrate to the existing cloud platforms that support the same technologies used to develop this project. Furthermore, the WebComfort is a mature CMS that is used for several applications, and its adaptation for the cloud would have to re-structure many of the systems that current Applications depend on.Thus, migrating the WebC-SaaS project to the Cloud was postponed.

3.4 Concepts and 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.

3.4.1 The Customer

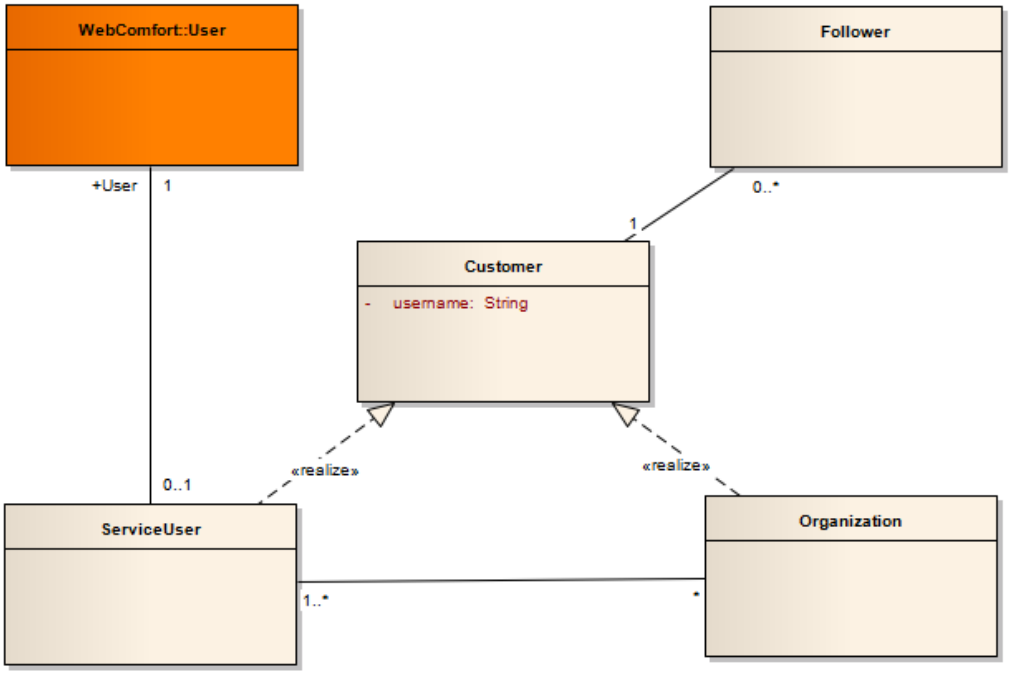


Figure 5 - The Customers View.

As shown in the image above, the ServiceUser and Organization realize the Customer class. This is due to the fact that WebC-SaaS has to support either individual as organizational clients.

For this reason, a special Customer type should exist to model this concept. This way, two distinct realizations of the Customer class were implemented. The Customer class defines a generic interface to interact with the WebC-SaaS platform and all its concepts. It can also define the necessary information to carry out payments.

The ServiceUser class implements a standard customer, with an associated user account in the WebComfort platform. This customer definition was created to complement the WebComfort User concept while serving as an interface to interact with all WebC-SaaS concepts. This type of object is automatically and invisibly created for an authenticated WebComfort User.

The Organization class is another implementation of the Customer concept, and is capable to assemble all the information of an Organization. This concept is not directly associated to a WebComfort User, but relates to multiple ServiceUser customers that are allowed to access and/or administrate the Organizations' subscriptions as well as configure other properties. A ServiceUser customer is allowed to be part of various Organizations. The Organization concept, nonetheless, must be created manually by a customer.

3.4.2 The Sandbox

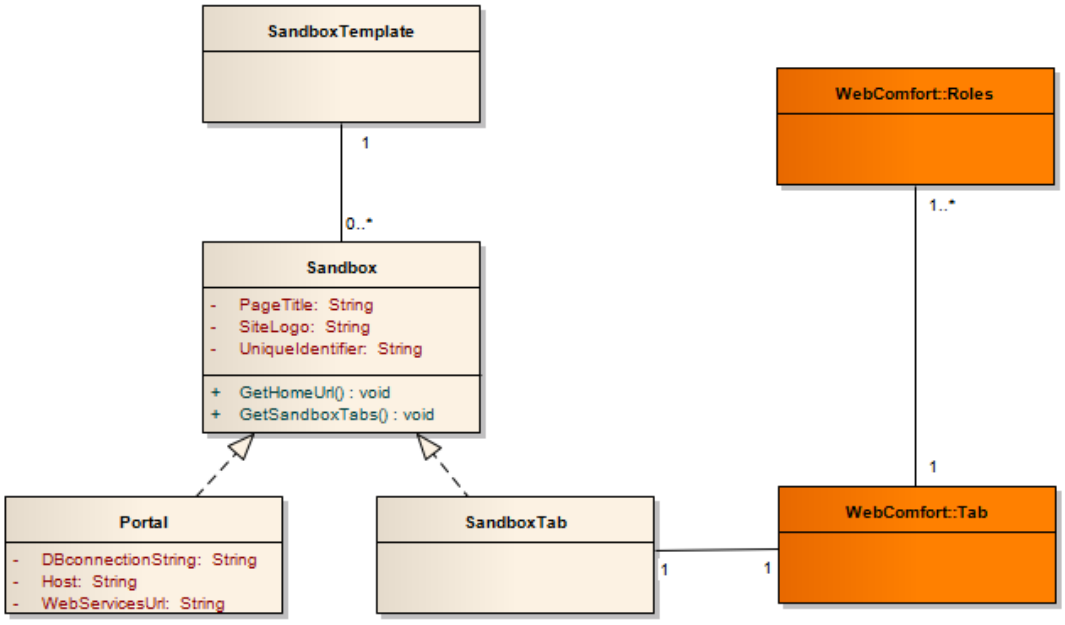


Figure 6 - The Sandbox View.

The Sandbox concept is an abstract class that represents the area in which services can be executed. There are two possible implementations for this class, which are the SandboxPortal and the SandboxTab. All of these classes are generated from a particular SandboxTemplate definition. For this reason, a reference to the originating SandboxTemplate is saved on the Sandbox class.

The SandboxPortal class represents an implementation of the Sandbox abstract class, and defines a Sandbox that is capable of managing services on a single-tenant WebComfort instance. For this reason, the SandboxPortal must define the necessary references to identify and communicate with the originating WebComfort instance.

Objects of this type are created and complemented from SandboxPortalTemplate definitions.

The SandboxTab class represents an implementation of the Sandbox abstract class, and defines a Sandbox that is capable of managing services on a shared WebComfort instance. These Sandbox types are created within a shared WebComfort instance, using a Tab hierarchy to define the sandbox context. Objects of this type are created and complemented from SandboxTabTemplate definitions.

3.4.3 The Service

Lastly, the service concept is explained as well as its relation with the others.

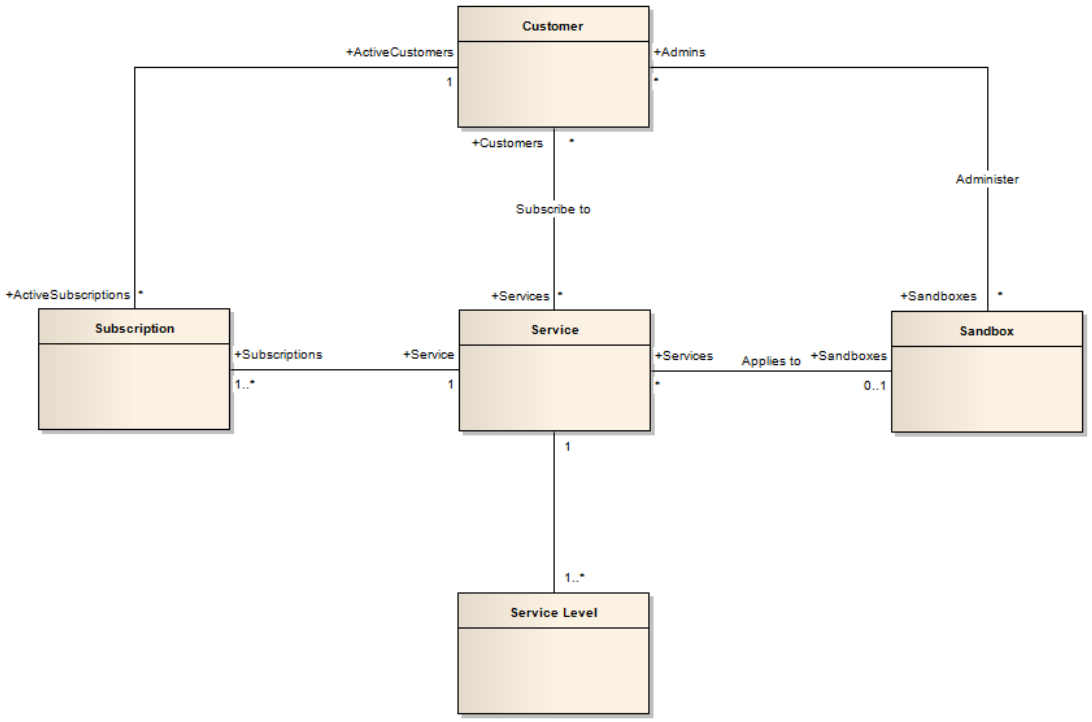


Figure 7 - The Service View.

The Service class represents a WebC-SaaS service. This concept is used throughout this platform to capture the module definitions, settings and roles a service defines. In order to be able to track this relation in the future, a reference to this object is maintained in the Service class.

3.4.4 Service Levels

Each Service may have several Service Levels. Tipically, in the SaaS model, Service providers create different products in order to achieve Customers acceptance. With more Service Levels, they can reach customers that only need a certain ammount of features, and paying accordingly. Thus, the Service Level class achieves this by creating independent packages of Modules and Definitions that can be tuned by the provider, creating many variations of the same service, for many different types of customers.

3.4.5 The Subscription

Another common approach in the SaaS model is the broad list of payment methods that providers offer to customers. Some examples are a kick-start fee and a monthly subscription, free plans for small and standard versions or any other pay-as-you-go subscription plan. Whenever a user wishes to use a Service, that bond is created by a Subscription. As long as the Subscription is active, Customers can access their services. Again, Service Providers may configure several Subscription Plans to match each Service Level.

3.5 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. Initially, this set of features were only designed to achieve service control and configuration, giving feedback to users about their service status and performance over time. However, once the system was deployed, more uses have emerged. The resulting features are followed explained in detail.

3.5.1 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. The main concepts involved in this system are described in the following image:

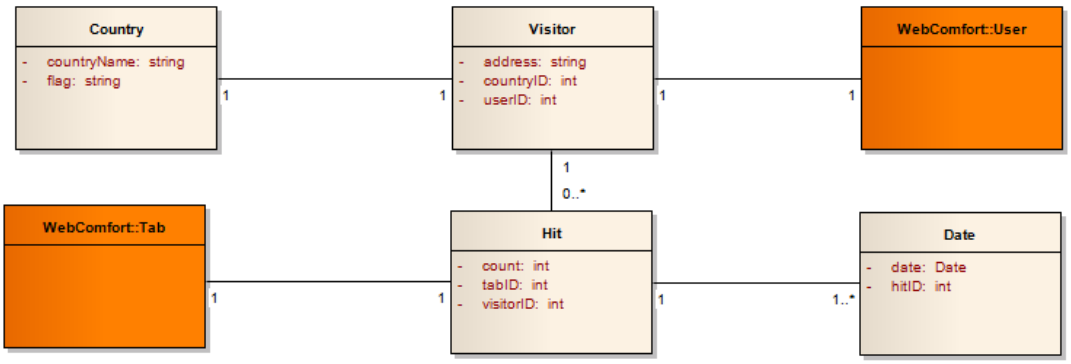


Figure 8 - The Visitor View.

As it can be seen, when a given user visits a Tab, that user becomes a Visitor. In addition, the Visitor concept extends user with other information such as Origin and address. The Hit concept is also born of this interaction. While the user becomes a Visitor, the relation between a Tab and a Visitor becomes a Hit. Also, when a Hit happens, the momment of that hit is recorded through the Date concept. These two concepts are separated in order to prevent an exponential growth of the database: The Hit will record how many times that user visited a given tab in the same instance saving space; Dates can be deleted when they are no longer needed or obsolete.

3.5.2 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 Service Layout is responsible to control resources and content production by the users, in order to maintain the platform coherent and stable.

3.5.3 User Interaction

The conservative SaaS model is directly related to the single-user Desktop experience. In other words, many SaaS platforms on the market, including all studied in this project, only provide tools to create and maintain the subscribed service. In contrast to this, this project offers, through the platform, features that allow users to review, interact and explore other services on the platform.

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 that have visited their services. Then, according to privacy settings regarding visibility, users may or may not see, interact or find these services in the first place.

3.5.4 The Bookmark

As WebC-SaaS aims to be integrated in many different contexts, it also ocurred the need to save some of these services as favorites. Since browsing repeatedly for the same services may be exausting 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.

3.5.5 The Follower

At this point, WebC-SaaS enables 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, Global Statistics and Customer Followers. Also, through the seller's perspective, these features play an important role in service receptivity and the extending of subscriptions.

Although the added value of combining Social Networks and the SaaS model is debatable, this project brings some new ideas to the discussion. Next, is presented the relation between the key concepts of the Domain Model responsible for the Interaction process:

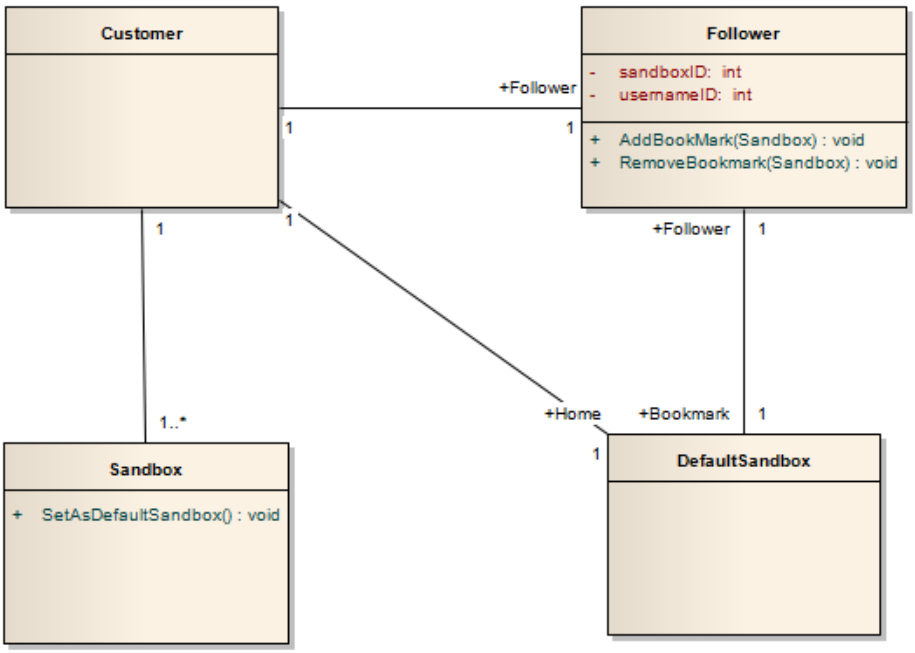


Figure 9 - The Follower View.

3.5.6 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 deph: User; Service; Tab. With these levels of abstraction, users can benefit from many shared functionalities in these three different scopes, as shown below:

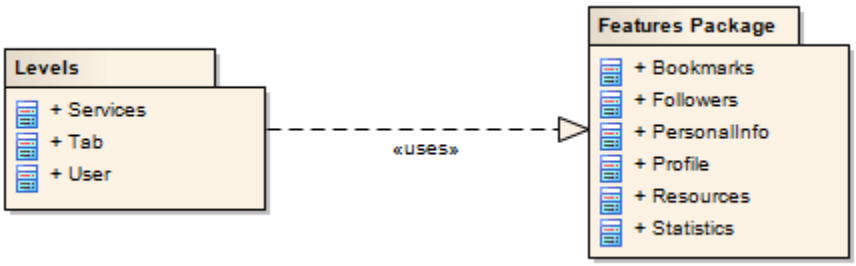


Figure 10 - User scopes and Features.

The User Level - At this level, users have an overview of all services that have been subscribed. Some of them may be active while others expired, but all of them may be reviewed. As an example, users can see the total services visited or count how many visitors they had in total. In addition, this level is also related to user's personal information, showing modules that manage these features.

The Service Level - When more refined information is required, users can brief one of their services. The Service Layout Page provides all functionalities at the Service level. They can configure the subsequent service parts (known as Pages or Tabs), configure privacy settings, review service visitors and resources as well as the logo and description.

The Tab Level - A service may consist in a set of different Tabs (or Pages), each one with different features. In fact, the same service may be visited by different users, but not all of them visited the same tabs. At this level, users can dig deeper into their service, reviewing the activity of each individual page.

4 WebC-SaaS – Design and Implementation Issues

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.

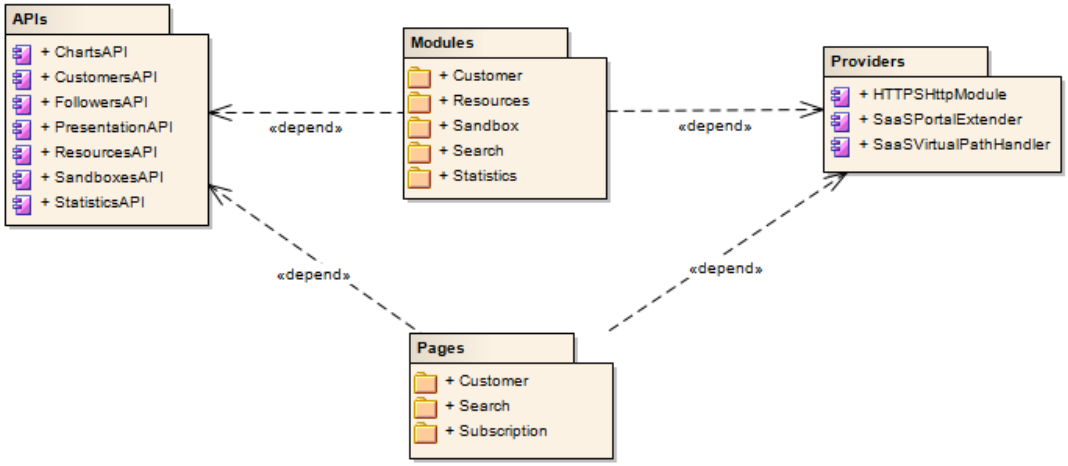


Figure 11 - Resource Relationship in the WebC-SaaS

Note that, the image above shows the dependencies between the WebC-SaaS components that were fully implemented thoughout this project's development.

4.1 Integration

The WebC-SaaS project was conceived to be organized as a standard WebComfort Toolkit, composed of a set of WebComfort Modules and Pages, as well as a set of function components including Providers and APIs. Because of this, the WebC-SaaS toolkit can easily be integrated in the WebComfort architecture.

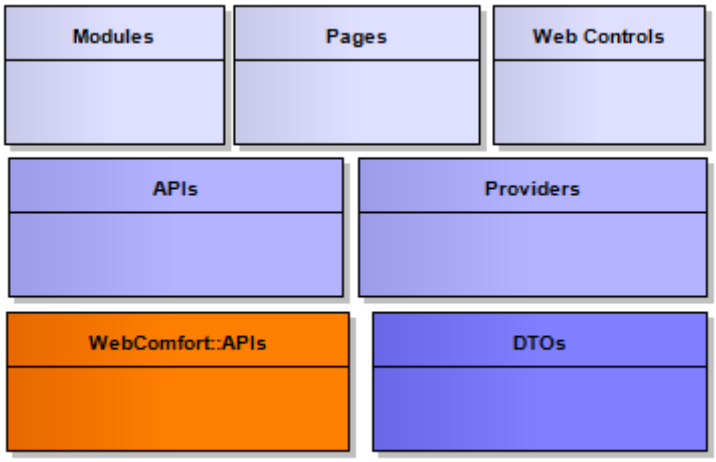


Figure 12 - Application Layers.

As mentioned above in this chapters introduction, WebC-SaaS follows it's platform architecture leading to an easy integration with WebComfort. As seen in the picture above, WebC-SaaS components are layered providing a series of abstraction levels. The DTOs (Data Transfer Object) manages all direct database transactions. The API layer encapsulates all DTO interaction and manipulation, preventing direct access from the upper layers. This means that all modules, pages or web controls act in a controled environment created by the APIs, making it possible to extend WebC-SaaS itself with more content without compromising the existing structure.

4.2 APIs

The APIs Package is divided into a subset of components that are semantically related. Following, the APIs are enumerated and a brief description of functionalities they hold.

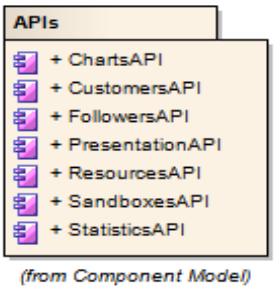


Figure 13 - The WebC-SaaS APIs

ChartsAPI - Organizing statistical data is critical when showing them to users. One of the possibilities of showing these data is through Timeline charts. To acomplish chart generation in WebC-SaaS, ChartsAPI was developed. There are many ways to implement such feature that would include a graphics library, drawing charts into files, and storing them in servers. But in a multi-tenant applications, this process would be

repeated several times for each user, decreasing performance and consuming resources.

Moreover, users can visit sites any instance, which means that charts generated by these methods would be outdated in a matter of minutes and would have to be redrawn. Thus, a better solution had to be found, using Google Charts. Instead of implementing and managing all files generated by a chart drawing library, WebC-SaaS delegates these features to Google Charts. What's more, the platform doesn't even have to download the generated image, it just refers to the Google Server where it was generated, where the user directly downloads the chart. The result is an almost-zero resource consuming feature to the platform that can be used several times by many users without affecting system's performance.

PresentationAPI - Although great part of the interface is defined by CSS classes, many concepts and interface parameters have to be reused along the platform. All interface design constants are defined and managed through this API. The result is a reutilization of concepts, keeping all modules and pages with the same look.

StatisticsAPI - Statistical data is generated everytime there is activity. Thus, the system has to store efficiently information in order to maintain reasonable levels of performance. On the other hand, when modules and features require these data, they may have to deal with tricky database accesses and combination of tables to meet their requirements. So, to maintain performance storing and retrieving this information, this API is attached to a series of database Views, designed for the most common and repeated tasks.

CustomersAPI - The Customer concept is an heavy extention of the WebComfort::User concept. To support all actions and features regarding customers, this API has emerged.

ResourcesAPI - This API was developed as an abstraction layer above the File System. WebComfort platform lacks orientations concerning file storage. Thus, each user and each module developer may store files as it suits them best. The result is an uncontrolable environment where the server host may be at risk of resource exaustion. To minimize these effects, ResourcesAPI creates simple rules for WebC-SaaS file storage, enabling user resource control.

FollowersAPI - To implement all functionalities related to User Interaction, this API was created. It enables the Bookmark and Follower concepts.

SandboxesAPI - Finally, this API manages all features related to Sandboxes.

4.3 Modules

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.

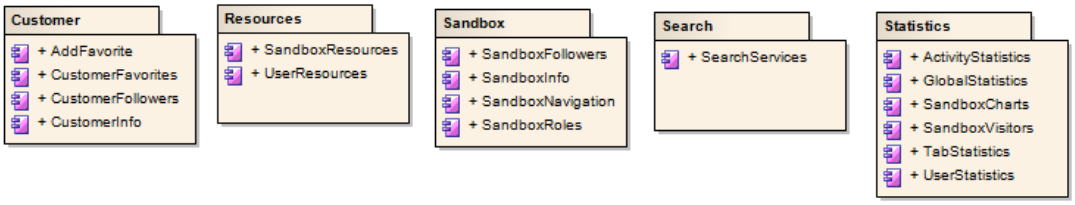


Figure 14 - List of WebC-SaaS developed modules.

The customers' package include all features related to the user. The CustomerInfo allows users to edit, delete and change personal information, aswell as changing privacy settings. CustomerFollowers , Favorites and AddFavorite are the key modules of the interaction concept, enabling bookmarks and showing followers of all customer's services.

Resources package holds the sandbox and user resource management modules. These two modules allow users to control resources at the service level(sandbox) or at an higher level(user).

Regarding Sandbox management, SandboxInfo is responsible for changing interface details such as logo or description. In addition, SandboxRoles completes Sandbox management with role definition and privacy settings. Simillar to UserFollowers, users can also see customers related to their services. Finally, Navigation module enables a tree navigation inside the sandbox.

The Search module may be simple but acts as a connector between services. It enables users to search for any given site, either by user or service name but only if those services respect the service owner privacy settings.

Finally, grouping all statistical information together, appears the Statistics Package. At a platform level, global and activity statistics retrieve information about all services and all users, ranking them according to visits or recent activity. Sandbox Charts and visitors are used for service reviewing, allowing users to see in a graphic manner the number of visitors and which visitors have recently viewed or used their services. Finally, User and tabs statistics group charts and visitors functionalities and show them in different abstraction levels: User; Service.

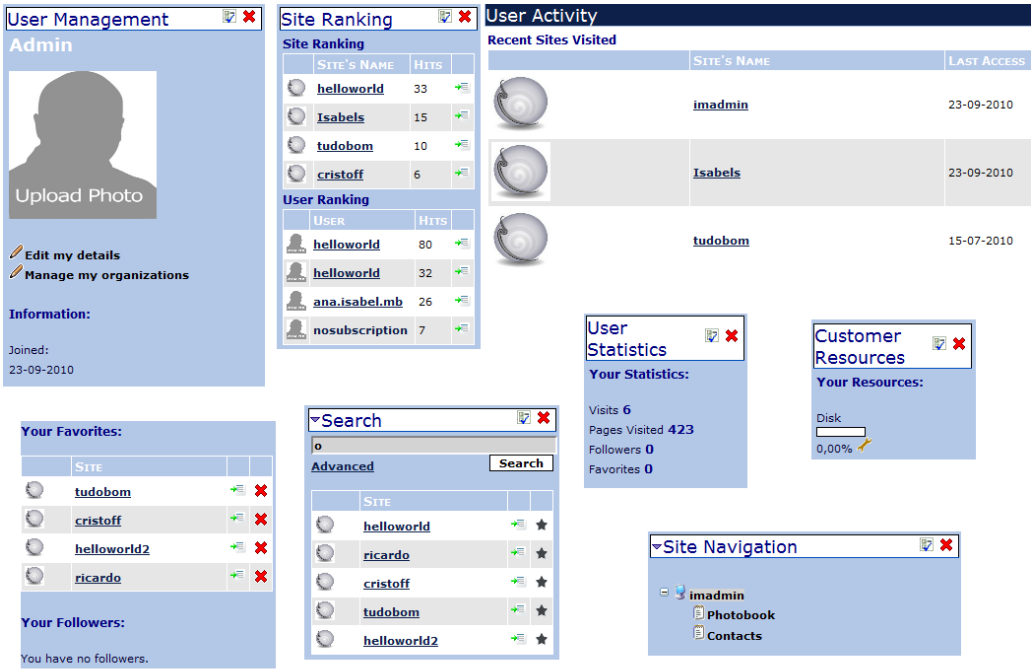


Figure 15 - Examples of WebC-SaaS modules.

4.4 Pages

Not all functionalities can be acomplished using the WebComfort Modules. To complement such features, it was necessary to develop auxiliary WebComfort Pages. Normally associated with editing and submiting data rather than presenting information, these pages complement some modules' functionalities. In the image below all of these pages are enumerated and organized in packages.

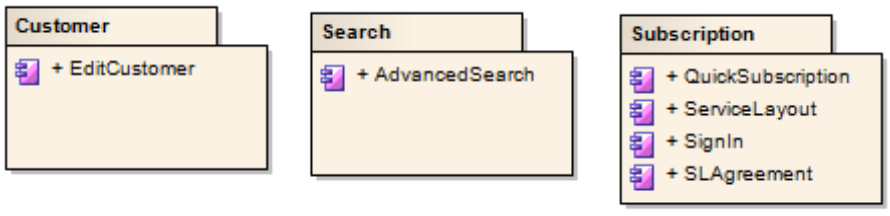


Figure 16 - WebC-SaaS main pages.

WebC-SaaS extends the User concept of WebComfort, enabling new features such as logos, nicknames and other information. That information is managed through this page. This page is divided into four categories: Personal Information; User Information; Organization; Roles.

The Search feature is composed of a brief and quick search, usually on the front page of a website, that is acomplished by its respective Module. Although, sometimes users require a more refined search, leaving the task to this specific page that enables searching by criteria and showing extended information about services.

Figure 17 - Customer Management Page.

The Subscription Process

Finally, the Subscription Package holds some major overhauls of the WebC-SaaS. The subscription process begins when an anonymous or registered user subscribes a service. For that purpose, the quicksubscription, SignIn and SLAgreement were created to simplify that process. Since WebC-SaaS extends WebComfort User concept, users had to register to the platform first before subscribing, leading to a long chain of pages between choosing a service and finally being able to use it. This workflow consisted of 5 to 7 different steps which included: preview service; choose service name; register to WebComfort; agree to SLA; confirm. The QuickSubscription simplifies all these steps into one page, summing all information and automaticly registering to the platform upon subscription, grealty decreasing the subscription process.

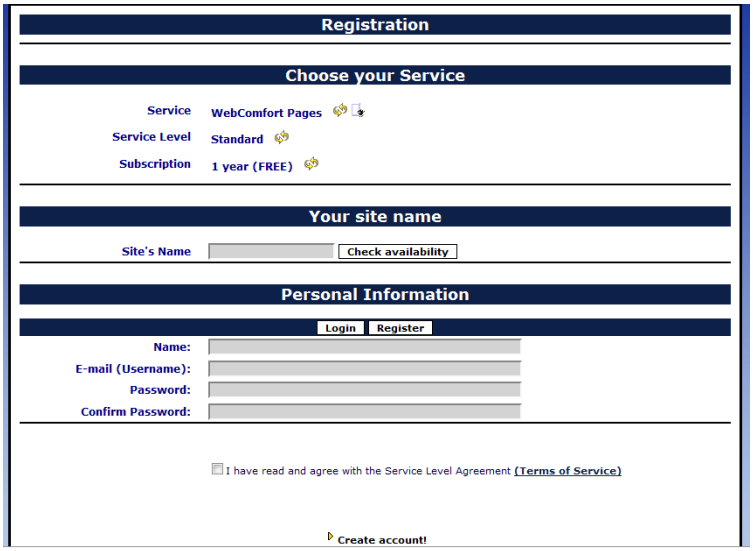


Figure 18 - The Simplified Subscription Process.

Service Control

Simillar to EditCustomer Page, ServiceLayout presents extensive information about the subscribed service. In the general tab, users can change the service logo and description as seen by other users. Following, the Statistics tab holds visitors, and graphics about service usage. Regarding the subscription, it also allows users to view expiration date or cancel it. Since a service may consist of several pages, the Pages tab allows users to manage their service at a more specific level. Finally, roles define privacy settings and Connections tab shows who is currently refering to that service with bookmarks.

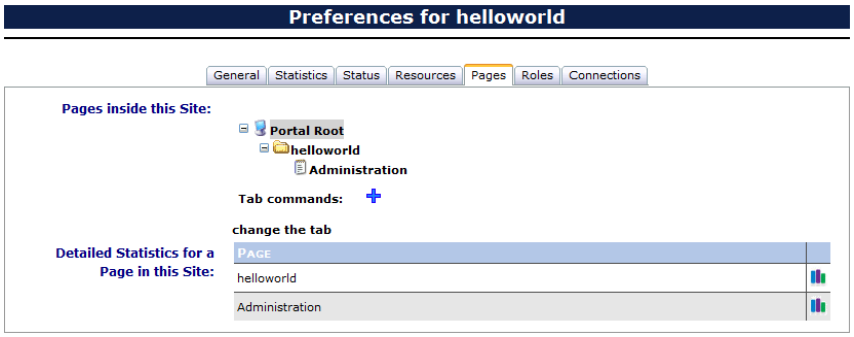


Figure 19 - Service Layout Page.

4.5 Providers

To support many of the functions of the WebC-SaaS platform, another structural component needed to be added to the project – WebComfort Providers. These components represent the boundary of a platform. As such, these concepts were used to manipulate the normal workflow of a WebComfort instance. In the image below are enumerated the implemented Providers components.

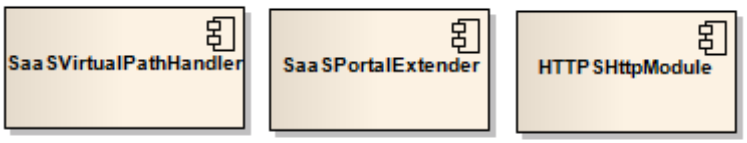


Figure 20 - The WebC-SaaS Providers

Of these components, the SaaSPortalExtender is the most relevant to this project. In this component is defined a class that extends the WebComfort Extender class. This makes the SaaSPortalExtender class a device for capturing WebComfort custom platform events, as well as the Begin and End events of a standard Web Application.

This device allowed implementing subscription validation and revoking; detection and treatment of WebComfort events like Module or User creations. Furthermore, this abstraction supported altering some additional aspects of the Portal, including: adding subscription links to the banner of the site; changing the behavior of the Home link button in order to redirect to the users' main sandbox Homepage; change the page's title and the banner Tabs, when in the context of a particular sandbox.

5 Validation

During this project's test phase, in order to validate its new features and performance, several Services were deployed both on a local machine and in a public server host. To this end, users have been invited to test the platform and provide feedback.

Unfortunatly, without a specific context, WebC-SaaS is very abstract and unappealing to users, which made the testing experience harder. Due to this, it wasn't possible to to evaluate this input by conducting a workshop or an inquiry with the users that tested the platform. Users would be asked to complete several tasks over the application, namely the subscription and access to a service, and in the end to fill up an inquiry to assess the level of functionality and satisfaction with the platform.

However, due to the fact that the WebC-SaaS project works in great part transparently to the user – as opposed to the actual applications that WebComfort support –, these inquiries would be hard to be unambiguous to the user. Users would likely tend to evaluate the actual WebComfort platform and applications instead of WebC-SaaS. Due to this, this method of evaluation was discarded.

Nonetheless, since many WebComfort Applications are in development, it was possible to merge WebC-SaaS development with real-application contexts, overcomming the abstraction imposed by WebC-SaaS. Thus, two validation scenarios emerged: Integrating with WebComfort Pages and WebTrails.

5.1 Case Study: WebComfort Pages

5.1.1 Overview

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 toolkit 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.

5.1.2 Services Definition

In order to meet the requirements of this Application, WebC-SaaS had to be configured accordingly. Note, that all configuration is done in back-office areas and with administration privileges. In every WebC-SaaS setup, there are several common steps that must be followed:

1. **Back-office Tab Configuration** - In order to manage users and services, there are specific WebC-SaaS modules for administrators that allow a general overview of all platform status. In addition, the Service creation modules are also added to this back-office area. This Tab is available for administrators and must be the first step during this toolkit setup process, since it will allow further operations to be executed.

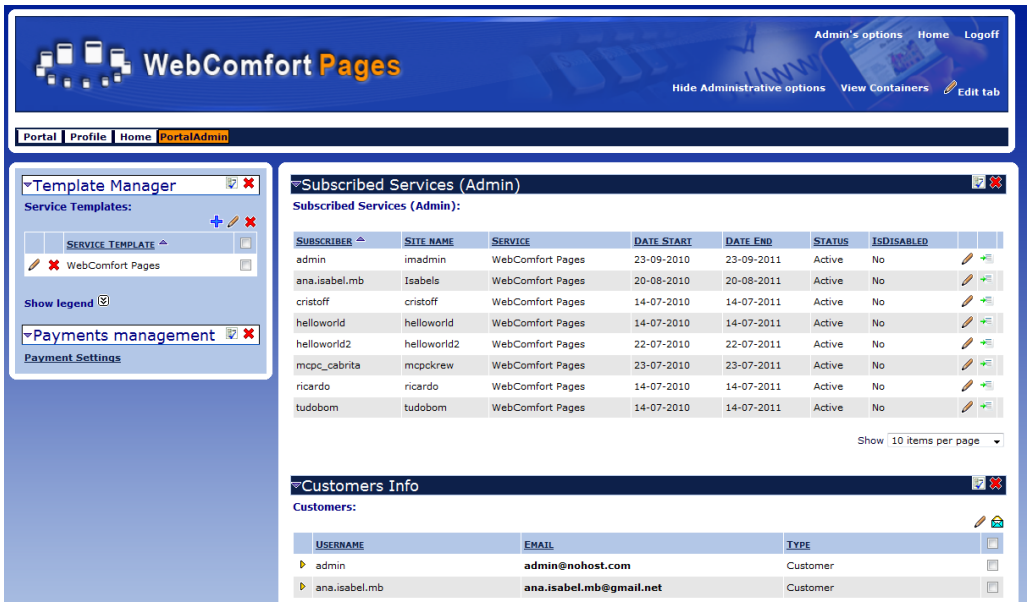


Figure 21 - WebComfort Pages Back-Office.

2. **Page Template Definition** - Before jumping into the Service definition, Administrators must design the page templates that services will use. This is done by creating a template tab page, with the desired modules and layout.

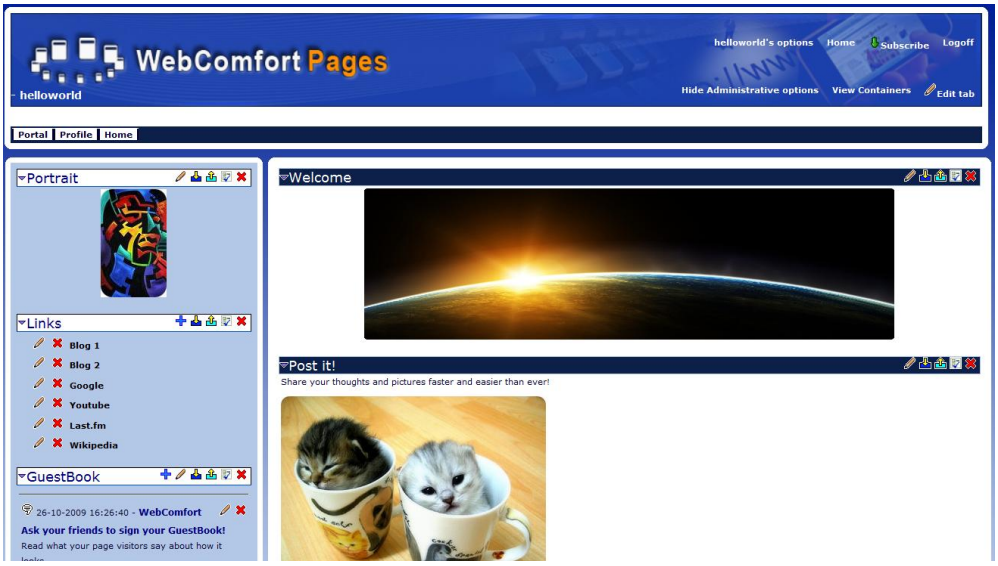


Figure 22 - WebComfort Pages Standard Service.

3. **Service and Service Levels definition** - Finally, everything is set for the Service creation. For this specific instance, only one service was created. This Service offers a configurable site for each user, with configurable domain name and up to 5 pages per site.

5.2 WebTrails

5.2.1 Overview

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

On one hand, it offers visitors the appealing experience of the sightseeing and trail interpretation in natural parks and historic places. It supports different media types such as maps, texts images, 3d models and audio. On the other hand it offers to Organizations, that promote such turistic places, tools that allow management and the promotion of contents. Such contents then can be used by Visitors on the same platform.

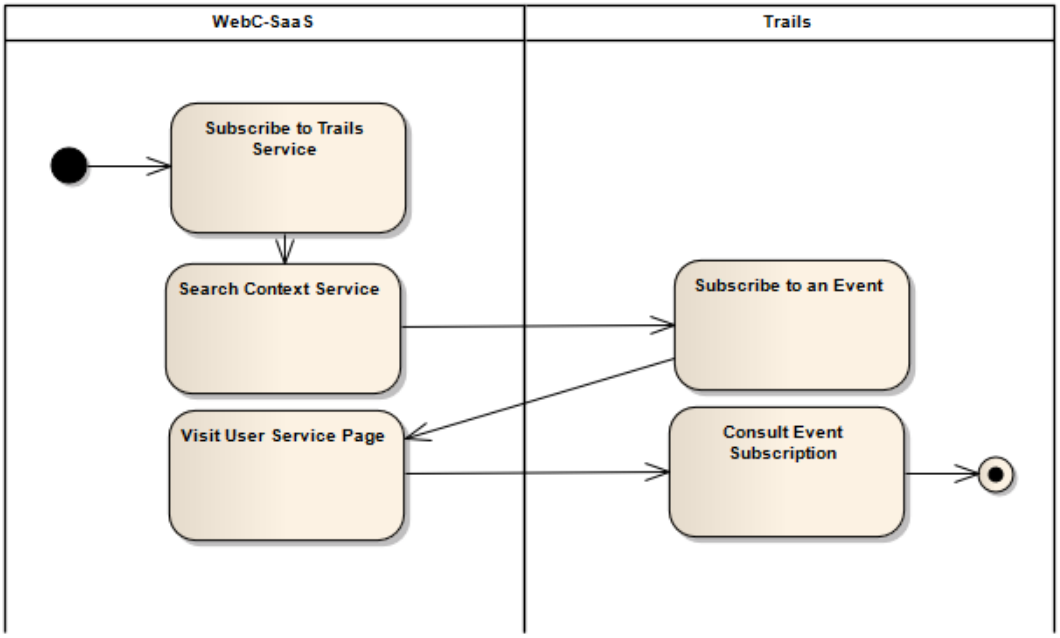


Figure 23 - Interaction between Trails and SaaS.

The image above describes an interaction between the two contexts inside the SaaS-Trails instance. This scenario starts when the user is willing to subscribe a service. In that momment, the WebC-SaaS is in charge of subscribing, showing previews, listing available services. After the subscription, another popular activity is to search for a service. However, when the user finds the desired service and whishes to access to it, the Trails system gets involved, when the event subscription occurs. Users that now have subscribed events, can see their personal page where this information can be managed, leading to another interaction with the SaaS platform. At the end, we see that this simple example involves multiple interactions between the two independent projects, both over the WebComfort Platform.

5.2.2 Services Definition

In order to satisfy these requirements, the SaaS framework needs to be configured accordingly. For this matter, two different services in this new instance had to be created:

Trails Partner

The Partner Service holds feature for Turistic organizations that wish to promote cultural places. This service offers modules that allow them to create and manage their turistic site and to create events over these sites.

Figure 24 - WebTrails Partner Service.

Trails Visitor

The visitor's service, is directed to users that wish to participate in such events created by the parters. It provides modules to manage subscribed events, as well as an activity log of past events.

Figure 25 - WebTrails Visitor Service.

6 Conclusion

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 innumerous 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].

All SaaS products have their own unique characteristics, small variants that deeply influence in SaaS' adoption. This project aims to understand which are the minimum requirements for companies to achieve success in this business, compensating for its model deficiencies and catapulting its advantages.

Software Industry evolves faster than ever, and it is possible that new delivery models emerge, possibly SaaS successors, that be far more robust and that emend SaaS' flaws for which no solutions came out yet. Until then, Software as a Service has the chance to grow and there is still much to explore.

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.

7 References

1 Dimitrios Georgakopoulos, Norbert Ritter, Boualem Benatallah, Christian Zirpins, George Feuerlicht, Marten Schoenherr, Hamid R. Motahari-Nezhad.: Service-oriented Computing. Springer (2006)

2 Hancheng LIAO, ChangQi TAO.: An Anatomy to SaaS Business Mode Based on Internet, in International Conference on Management of e-Commerce and e-Government (2008)

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)

4 Alexander Benlian, Thomas Hess, Peter Buxmann.: Drivers of SaaS-Adoption – An Empirical Study of Different Application Types, in Business & Information Systems Engineering (2009)

5 Farooqui N.K.: Software as a service: Analysis of 'Google Sites' as KM Tool for Academic Environment, in Communications of the IBIMA Volume 5 (2009)

6 Manish Godse, Shrikant Mulik.: An Approach for Selecting Software-as-a-Service (SaaS) Product, in IEEE International Conference on Cloud Computing (2009)

7 Nitu.: Configurability in SaaS (Software as a Service) Applications, in ISEC'09 (2009)

8 João de Sousa Saraiva, Alberto Rodrigues da Silva.: The WebComfort Framework: an Extensible Platform for the Development of Web Applications, in Euromicro Conference Software Engineering and Advanced Applications (2008)

9 Jon Brodkin.: 12 Issues You Need to Know about Software as a Service (2007) Retrieved 14/11 2009 from Network World: http://www.networkworld.com/news/2007/073107-software-as-a-service-12-things.html

10 Paul Korzeniowski.: SaaS Approach Not A Fir For Enterprises (2009) Retrieved 2/12 2009 from Network Computing: http://www.networkcomputing.com/servers-storage/saas-approach-not-for-a-fit-for-enterprise-search.php

11 Keith Bennett, Paul Layzell, David Budgen, Pearl Brereton, Linda Macaulay and Malcolm Munro.: Service-Based Software: The Future for Flexible Software. Bennet et al, Seventh Asia-Pacific Software Engineering Conference (2000)

12 Mark Turner, David Budgen, Pearl Brereton.: Turning Software into a Service, (2003)

13 Reginald Davis, F.Dennis Kennedy.: Working in the Cloud, ABA Journal (2009)

14 Hyun Jung La, Soo Dong Kim.: A Systematic Process for Developing High Quality SaaS Cloud Services, in CloudCom (2009)

15 Uwe M. Borghoff, Johann H. Schlichter.: Computer-Supported Cooperative Work: Introduction to Distributed Applications, Springer (2000)

16 Gavin Bell.: Building Social Web Applications: Establishing Community at the Heart of Your Site, O'Reilly Media (2009)

17 Gabriel Coimbra.: Software as a Service: As múltiplas dimensões do SaaS. Microsoft Solutions Day conference (2008)