A SaaS-based framework to support the management and deploy of web applications for exchanging information and sharing knowledge

Yliana RIVERO, Mónica SAMPIERI, Marco FERRUZCA[a], Joaquin FERNÁNDEZ, Josep Mª MONGUET, José Luis MUÑOZ, Hernando VILLALOBOS, Berenice BLANCO

Multimedia Applications Laboratory (LAM),
Polytechnic University of Cataluña (UPC), Barcelona 08028, Spain

[a] Metropolitan Autonomus University-Azcapotzalco (UAM-A), Mexico City 02200, México

Abstract

The Multimedia Applications Laboratory research group at the Polytechnic University of Cataluña has been working for the past four years in the development of a system denominated as COLS to support the management and deploy of web applications for exchanging information and sharing knowledge in the context of research organizations. The evolution of the web applications developed by this group and the requirements for its reuse, configuration, multi-user efficiency, scalability and fast delivery time, have made us consider the need to convert COLS into a SaaS-based framework, which provides an excellent opportunity for research, since in the domains of knowledge management and e-learning in research organizations there are few SaaS experiences. In this paper we present a first approach of this SaaS-based framework.

Keywords: SaaS, software as a service, knowledge management, e-learning, software framework

1 Introduction

For the past years, the techniques, processes and methods of software development have been dominated by supply-side issues, giving rise to a software industry oriented towards developers rather than to the users. To achieve the levels of functionality, flexibility and delivery time required by users, the software development demands a radical change, with a more demand-centric view, in which the software is delivered as a service in the frame of an open marketplace [1]. Currently, there are evidence of this approach is being adopted by industry: The Application Service Provider (ASP) model became popular during the dotcom bubble, with the emergence of the first wave of Internet enabled applications. It is about companies that install and support licensed software of third parties and provide remote access to their customers. However, the ASP services were never successful because in most of the cases the software was not sufficiently flexible for its daily use. The solutions were turning out to be cumbersome of using, since they had not been designed for its current use on web.

More recently, the term Software as a Service (SaaS) can be traced back to a white paper called "Software as a Service: Strategic Backgrounder" by the Software & Information Industry Association's e-Business Division published in Feb. 2001 [2]. SaaS was born as an evolution of ASP and it can be understood as a software delivery paradigm where software is hosted off-premise and delivered via web and the mode of payment follows a subscription model.

A wide range of online applications, including email, human resources, business analytics, customer relationship and enterprise planning, are available [3]. This market will be growing in the next years, according to Gartner, trends in SaaS business are [4]:

- By 2009, 100% of Tier 1 consulting firms will have a SaaS practice.
- By 2010, 20% of companies deploying ecommerce will use a SaaS delivery model.
- By 2010, 15% of large companies will have started projects to replace their Enterprise Resource Planning (ERP) backbone with SaaS-based solutions.
- By 2010, 85% of SaaS vendors will offer uptime service levels of 99.5% or beyond in standard contracts, as well as performance SLAs.
- By 2011, 25% of new business software will be delivered as SaaS.
- By 2012, Business Process Management Suites (BPMSs) will be embedded in at least 40% of all new SaaS offerings.
- By 2012, more than 66% of Independent Software Vendors (ISVs) will offer some of their services as SaaS.

However, a review of scientific literature about SaaS reveals that there are few experiences of its application in the domains of knowledge management* and e-learning practices in research

* In this study the knowledge management term refers to the method that simplifies the process of sharing,

organizations. Based on this situation, this paper presents the first approach of a SaaS-based framework†, denominated as COLS‡ in the rest of the paper, to support the management and deploy of web applications for exchanging information and sharing knowledge. The proposed framework includes the definition of a technological reference architecture that seeks to establish the foundations for future web developments. It also includes the establishment of set of unified specifications which facilitate the design and management of production processes, from the developer standpoint. The application of the SaaS's principles in the definition of the framework, will lead to advantages of configuration, multi-user efficiency and scalability; foreseeing the needs of reuse, maintenance, development and efficient distribution of future developments. Finally, the framework is presented with an example of its application.

2 COLS Architecture

COLS can be defined as a set of processes and technologies used to solve a problem. It is the framework on which several objects are integrated to give a solution. COLS takes into account not only the technology, but people and processes, and how they are organized into working groups and the methods used to carry out their activities. COLS integrates various web applications for multiple users. These applications allow: (a) manage knowledge and innovation, (b) improve the ways in which users create, share and reuse knowledge and useful content to their learning objectives, training and / or communication, (c) support the link between processes, people, tools, tasks, put them together in a virtual work environment, and (d) develop effective exchange practices among geographically dispersed work teams. The technological design of COLS has been developed following the Model-View Controller (MVC) design pattern [5]. The result is differentiate and separate data elements and specific functions of the platform from the presentation of data in the user interface and business logic. More specifically, MVC implementation is represented as: model (users, environments, artifacts, contents); view (php pages, and CSS layers implemented in appropriate formats to interact with users) and controller (communication between the model and view layers, control of events and access to contents). The conceptual design of the model layer is based on the idea that COLS can be understood as a cognitive system [6] composed of the four above-named entities involved: users, environments, artifacts and contents (also understood as products), as shown in figure 1.

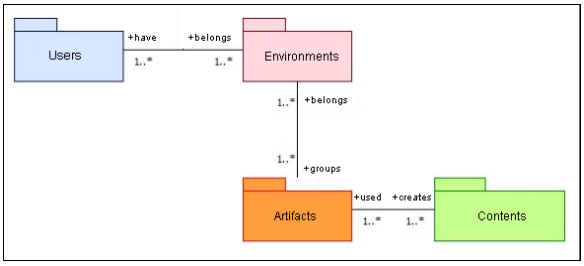

Fig. 1. Entities involved in COLS

Three modules are integrated from these four entities: (1) User Module: represents a single module that manages user data across a centralized platform. (2) Content Module: represents a single module that manages centrally the contents of the entire platform. The environments are interpreted as micro-platforms (or communities) that live within the macro platform. Artifacts are interpreted as applications (features or tools) that facilitate the users' tasks of a specific environment. Both, environments and artifacts are configured in the Control Module (3).

As noted above, the artifacts are defined as different applications (or features) that an environment may have. Thus, it is necessary to define configuration information describing each type of artifact. However, artifacts are not the final applications implemented in an environment. The environments use instances of artifacts. That is, the artifact describe the application type, how it works, who created it, who managed it, how it's installed, how to configure it, etc. The instance represents an implementation of the artifact in a specific environment with its own configuration. Figure 2 represents a systemic view of COLS.

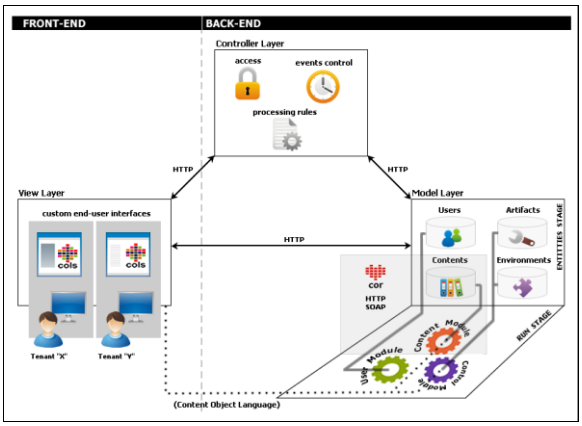

Fig. 2. Systemic view of COLS

distribute, create, capture and understand the knowledge of an organization.

† About the framework concept, the meaning given to it in this study relates to a setting that serves as a guide for the design, development, management and deploy of web applications.

‡ Should be noted that the acronyms have no meaning, is just a name.

For example: there is forum type, understood as an artifact, which has a manager for creating forums and registering their users. The artifact itself is instantiated each time it is added to a specific environment. All the contents of each instance are integrated in the content module and these contents are accessible for it. The generated content by an instance of the artifact carries an identifier that indicates which instance belongs to. From the identifier can be deduced in which environment has been created, by whom and under what profile.

Regarding the use of artifacts, it is very important to describe the profiles that can be used. The technical architecture on which COLS is based is flexible enough to accept all kind of artifacts, among which can be found simple tools with profiles of use of a single level, but also more sophisticated tools, such as managers, who employ two levels of profile.

Additionally, services architecture is used, represented by web services, in which components of the platform are prepared to facilitate the integration, communication and distribution of applications to multiple users in different environments. An important advantage of this approach is that it leads to a flexible and agile development process, able to meet rapidly changing needs of any organization that carries out tasks of knowledge management and innovation. "Software has become a critical element in all aspects of modern life, supporting wealth creation and being deployed in products and processes designed to improve the quality of life. The demands placed on the software engineering community, such as productivity, flexibility, robustness and quality, have increased at an exponential rate, leading to new development paradigms, formalisms and methods of working, the success of which have been truly remarkable" [1].

3 Conversion to SaaS

Software as a Service (SaaS) is a software delivery model, which provides customers access to business functionality remotely (usually over the internet) as a service. The customer does not specially purchase a software license. The cost of the infrastructure, the rights to use the software, and all hosting, maintenance and support services are all bundled into a single monthly or per-use charging. The literature [7, 8] considers four levels of SaaS Maturity models related to support of tenants face-to-face instances of software solution and database:

- Level 1 Maturity Model (Ad Hoc/Custom): In this model, the SaaS vendor creates a separate custom instance for each tenant. The vendor may choose to do this for a variety of reasons: 1. The system architecture of the application may not be amenable for customization and the vendor is required to create separate copies of the application for each vendor requiring customization. 2. The application may have not been designed for multiple tenants. 3. The application may not be scalable, which will require a separate instance for each tenant. Clearly Level 1 maturity model has many business issues as such as– high operational cost due to separate instances for each tenant, high support cost, high cost for development and configuration management of different versions of the application and high infrastructure cost.
- Level 2 Maturity Model (Configurable): In this model the SaaS vendor is also required to deploy separate instances for each tenant due to #2 and #3 stated in the Level 1 case. However, each instance may be configured to look different and show different characteristic. Unlike Level 1, single code base is used in Level 2. Though the instances are different for each tenant, there are no overheads of creating and managing different code bases. But, the infrastructure, operational and support costs are still high and comparable to the Level 1 model.
- Level 3 Maturity Model (Configurable and Multi-tenant efficient): Level 3 model allows the vendor to support multiple tenants on the same instance. The application, unlike in Level 1 and Level 2, is designed to support multiple tenants. The infrastructure, operational, support costs are significantly lower in the case of Level 3. However, development costs of a Level 3 SaaS application would be higher than Level 1 or Level 2. However, in Level 3, scalability may be an issue and support for additional tenants is usually done through separate instances or vertical scaling of upgrading the hardware on which the SaaS applications are deployed.
- Level 4 Maturity Model (Scalable, Configurable and Multitenant efficient): Level 4 is the most mature and advanced SaaS deployment model. Level 4 deployment model provides a high degree of support for scalability. The load balancer provides a single virtual instance to all the tenants. The load balancer will redirect each tenant to a particular instance depending on the load balancing policies. The assignment of a tenant to an instance in Level 4 is dynamic and determined at runtime by the load balancer.

There are many reasons why we decided to develop and distribute the COLS applications as SaaS applications: first, the challenge we have is to find a

way to develop web based applications that can evolve with changing business needs [9]. Also we have to develop these applications rapidly as well as in a cost effective manner. The development approach should also reduce the gap between what the users actually want and what is being implemented in terms of functionality [10].

In this regard, COLS applications are in a transition phase to a SaaS model, which will lead to advantages of configuration, multi-user efficiency and scalability; foreseeing the needs of reuse, maintenance, development and efficient distribution of future developments. Currently, in COLS each artifact instance may be configured to look different and show specific characteristics for each tenant. There are different aspects of a COLS environment which can be configured [11]:

- User Interface (UI): Configurability of user interface means the ability to change the look and feel of the UI available to the players. We can configure the user interface to change its look and feel or to reflect corporate branding. The UI features like icons, colors, fonts, titles, etc. can be changed. This not includes only adding/editing/deleting different controls on the pages but also add/delete/edit forms in the user interface. Every page structure is defined by templates with css styles and layers which facilitates setup and customization.
- Functionalities: Other than configuring the UI to give unique look and feel to COLS environment, we are able to configure the behavior of the application. This is important because it makes possible that the same kind of artifact may have different behaviors in different organizations.
- Data: Data is at the heart of any SaaS application. Data drives the SaaS applications. In case of a particular type of SaaS application, that is, in a specific domain like knowledge management, there will be similar type of data across different tenants. But there will be some unique features in the database of each tenant. COLS allows through its content module storing data which meets the most common requirements of the tenants with an option to add the tenant' specific data requirements like adding extra fields or adding specific constraints.
- Access control: Each tenant using a COLS environment will have multiple individuals using the system. We can create individual accounts for end users, and define which resources and functions each user will access. Since there will always be an organization's specific access control data, it is necessary allow create, edit or delete roles/users specific to the organizations. Users will be grouped into different roles according to the organizational structure and the access control privileges can be configured for each of these. The access control privileges specify what data and UI in a particular role can be access. There are some data which can only be viewed (only read permission) while there might be some other data which is editable (read and write permission) by users of a particular role.

4 COLS Framework

The framework that we are developing includes the definition of a technological reference architecture that seeks to establish the foundations for future web developments of our research group. Also, includes the establishment of a set of unified specifications which facilitate the design and management of production processes, from the developer standpoint. In order to explain the progress we have achieved in this direction, in this section we describe the definition of the components of COLS taking as a references one of the projects we have developed: E-FREN. It should be noted that we have already implemented other COLS environments, primarily on topics related to e-learning, including: *http://www.hoyunpocomas.net* (innovation & research community), *http://www.opera-elearning.com* (opera e-learning system), *http://www.disfagia.hoyunpocomejor.net* (treatment of dysphagia).

4.1 Environments

In COLS, an environment is defined as all those virtual spaces designed to facilitate the communication, learning and interaction among its users through the provision of different functionalities. The environment is the context in which the subjects create, share and reuse knowledge and useful content for its objectives. The environment defines which people are involved in the virtual work space (users), what software objects are used (artifacts) y what contents are available.

E-FREN (figure 3) is a COLS environment, which has been developed for the management of knowledge in nephrology. Specific goals of E-FREN system include: (a) Visualize and transfer the expert's knowledge to nephrologists residents, (b) Broaden the learning experience to the specialists in a useful format by sharing the learning process between several hospitals, (c) Complement the tutor task and increase the tutorial efficiency by doing more with less effort, and (d) Focus in the scope of

experience, transmitting more and better knowledge in nephrology and its skills (dialysis and transplant). The system is addressed to medical specialists (to update or contrast knowledge), and internal medical residents (doing nephrology specialization in hospitals) from Spain and Iberoamerica. The system E-FREN has the certification of the Spanish Society of Nephrology (SEN from its Spanish name), has credits from the Committee on Continuing Education of the National Health System and the guarantee of the National Commission of Nephrology.

Fig. 3. E-FREN web site index page

4.2 Artifacts

In COLS, an artifact refers to all tools or functionalities that facilitates the users work in a virtual space (environment), contributing to the achievement of the objectives of the context to which they are associated. The design and implementation of new artifacts is always done thinking in its possible reusability in other environments. When this reusability is needed the cost of adapting the artifact to another environment is very low. Most times this adaptation only involves the configuration of the new instance and the view update to apply it the design of the new environment. One of the ultimate aims of COLS development team is to have a complete repository of artifacts that can be easily reused by any new environment that is required to implement. E-FREN has the following artifacts:

- Menu: Contains all the functionalities to display the menu in the private and the public part of the environment and the options of content management. The options are enabled depending on the user's profile.
- Personal data: Defines the services of the user's personal profile, lists of persons per profile, etc.
- Forums: Contains the functionalities of query-execution of the discussion forums, and its management. It also contains the functions of all related contents (additional documents, papers, reviews, etc.).
- Activities: It is the central artifact of E-FREN, it defines the structure of courses (modules, themes, cases for resolution, study cases, bibliography, etc.). It contains the features of query-execution and management.
- Progress: Gathers information about events generated in the environment, e.g. access to contents, user's participation, evaluation results, etc.
- Publications: Contains the functionalities to post and view news and calendar of activities.

5 Future Directions

Currently we are working to extend COLS in different directions:

- The development of semantic tools and methodologies to build a domain module related to the content. This module will be the basis that provides semantic services about the contents in a new generation of artifacts.
- The use of workflows to implement processes integrated in COLS.
- The development of monitoring tools that could trace the use of the system done by every user. Beyond the contents that create the user, this monitoring information, like consulted contents or time spent, could be later analyzed and typified to let the system know better the users.
- The extension and exploitation of the user module to allow, jointly with the control and domain modules, the development of adaptive artifacts.
- Moreover, as we need new features in the environments, create new artifacts that can be instantiated, reused in other environments.
- The development of an incremental design methodology for improving the design process of web applications.
- Making innovation happens with the development of tools that allow users to participate in the design process of web applications.

References

1. Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L., Munro, M. (2000). *Service-based software: the future for flexible software*. In: Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific. pp. 214-221.

2. SIIA (2001-02). *Software as a Service: Strategic Backgrounder*. Retrieved from http://www.siia.net/estore/ssb-01.pdf
3. Jacobs, Jean (2005). *Enterprise software as s*ervice. July 2005. Queue, Volume 3 Issue 6.
4. *Predicts 2007: Software as a Service Provides a Viable Delivery Model* (2006). Gartner, Inc.
5. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. (1996). *Pattern Oriented Software Architecture: A System of Patterns*. John Wiley & Sons.
6. Ferruzca, M., Fabregas, J. & Monguet, J. (2007). *MAIA: a methodology for applying Distributed Cognition to the management of learning system*s. C. Montgomerie & J. Seale (Eds.), In: Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007 (pp. 1413-1422). Chesapeake, VA: AACE.
7. Chong, F and Carraro, G. (2006). *Architecture Strategies for Catching the Long Tail*. Microsoft Corporation, April 2006, http://msdn.microsoft.com/en-us/library/aa479069.aspx
8. Kwok, T., Nguyen, T., and Lam, L. (2008). *A Software as a Service With Multi-tenancy Support for Electronic Contract Management Application*. In: Proceedings IEEE Conference on Services Computing, July 2008.
9. Ginige, A. (2001). *New Paradigm for Developing Software for E-Business*. Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC 2001). IEEE Computer Society, Stresa.
10. Fischer, G. e. (2004). *Meta Design: A Manifesto for End -User Development*. Communications of the ACM 47(9), (pp. 33-37).
11. Nitu (2009). *Configurability in SaaS (software as a service) applications*. In: Proceeding of the 2nd Annual Conference on India Software Engineering Conference (Pune, India, February 23-26, 2009). ISEC '09. ACM, New York, NY, 19-26. http://doi.acm.org/10.1145/1506216.1506221