

# On Using Software as a Service to Deploy an Early Warning Service

Franclin S. Foping, Ioannis M. Dokas, John Feehan, Syed Imran

Cork Constraint Computation Centre

University College Cork

Ireland

E-mail: {f.foping, i.dokas, j.feehan, s.imran}@4c.ucc.ie

## Abstract

*This paper was motivated by the involvement of the authors in a case study aiming at deploying an early warning system (EWS) for water treatment plants in Ireland using the Software as a Service (SaaS) business model. The goal of this paper is twofold. At first, it attempts to clarify the relationship between the SaaS business model and other cloud computing technologies and to point out the advantages of SaaS over on-premises software. The second goal is to propose a novel early warning service architecture for critical infrastructures that will hinge on the SaaS business model in order to be deployed over the Web. The novelty of our project lies in the fact that it will propose a new approach based on the SaaS business model to provide an EWS for water treatment plant operations in Ireland that will be accessible over the Web.*

## 1. Introduction

The United Nations and other organizations warned of a condition of instability known as “water crisis” [12]. Sound indicators of the so-called crisis are, among others, the scarcity of usable water in a large proportion of world’s population and the increasing numbers of water contaminations and pollution cases in a global scale. Drinking contaminated water can bring about several adverse effects to the society and to human health including diseases such as cholera, typhoid fever, diarrhea, hepatitis and dysentery. According to the World Health Organization, about 1.8 million people pass away every year across the world from waterborne diseases mainly caused by polluted drinking water. Sadly enough, in 2004 about 90% of these people were children under five years old [29]. These deaths could be easily avoided by improving the way we manage sanitation and drinking water. Thus, one target of the Eight Millennium Development Goals set by the United Nations, is to significantly reduce, by 2015, the proportion of the population

without sustainable access to safe drinking water and basic sanitation in order to address the risk of waterborne disease outbreaks [25]. This proves the urgent need to provide appropriate solutions to counteract this ominous situation.

Assuring safety and security of water treatment and distribution systems is the primary concern of drinking water management. Systems that provide an early warning service (EWS) in the domain of water supply can be viewed as very useful tools in supporting proactive risk management approaches and strategies. Some types of sensor and computer systems can enhance the collection and interpretation of monitoring data related to critical water and operational parameters. Other types of IT systems can enhance feedback control mechanisms for maintaining good water quality and can support the effective and reliable dissemination of important information to stakeholders in a timely manner.

The ultimate goal of our research, which is at its early phases, is to design and develop a Web-based EWS which will be used by the personnel of institutions involved in the drinking water delivery governance model of Ireland, such as the Environmental Protection Agency (EPA), local authorities and the Health Service Executive (HSE). Its aim is to make visible, as early as possible, the disturbances of the performance of drinking water treatment plants to appropriate stakeholders. Based on the requirements, it was decided to deploy the EWS using the Software as a Service (SaaS) business model. In our case, the EPA will be the SaaS provider and each water treatment plant in Ireland can be considered as a tenant of the service. The goal of this paper is to propose a high level architecture of the EWS having discussed first the SaaS business model and its requirements.

The paper continues with a brief description of our use case in Section 2 as well as the relevant Web technologies in Section 3, followed by a brief overview of SaaS applications in Section 4. In Section 5, the link between SaaS and other related cloud computing technologies will be explored. In Section 6, a proposed architecture for our SaaS

based early warning system will be presented, followed by a presentation of research issues related to our project in Section 7. Previous works will be highlighted in Section 8 and the paper will be concluded in Section 9.

## 2. EWS Requirements and Work in Progress

### 2.1. Project description

A brief description of the requirements of the EWS is given as follows: Users, scattered all over Ireland, will be able to change the status of the components of the socio-technical system. In addition, real time sensor data of critical water quality parameters will be collected in order to assess the possibility or the likelihood of an occurrence of any operational problems and failures in a timely manner. Given some evidence on the status of a component (e.g. on or off, available or not) or about critical values of water quality parameters, the EWS will compute the likelihood of a drinking water contamination incident. If the likelihood is above a threshold then alarm messages are instantly sent to stakeholders in different components of the socio-technical system. The information embedded in the alarm message that is sent to stakeholders should hinge on their status. In addition, emergency response procedures must be sent to appropriate stakeholders such as plant managers and local authority personnel. The requirements of our EWS can be stated as follows:

### 2.2. Work in progress

Our research aims at devising a deployment model based on the SaaS business model for the EWS defined above. Water treatment plants in Ireland will be used as case studies. The Web-based application should be able to provide an interactive map of all Irish water treatment plants as well as offering to authorized users the opportunity to assess the likelihood of failures of water treatment plants. Our application should be able to handle simultaneously several users hence its multi-tenancy property. Currently, we are still at the design phase where the goal is to first identify important components of the EWS and secondly the technologies that will be used to permit reliable and effective interactions among them.

- To be accessible by all stakeholders who are scattered in Ireland;
- To provide important and relevant information to each stakeholder;
- To allow the profiling of each water utility and monitoring of critical operational parameters;

- To allow the modeling of knowledge about causes - failures - effects;
- To allow updating of knowledge related to causes - failures - effects;
- To allow estimation of the potential of failure of water utilities given some evidence and to be able to compute the potential of failures with uncertain data;
- To notify the appropriate stakeholders when the estimation of a potential failure is considered to be “high”;
- To allow as inputs real time sensor data and to allow updates of the status of the components of the water utilities by water utilities personnel;
- To provide alert messages to the appropriate stakeholders;
- To provide advice and emergency response procedures to the stakeholders.

These requirements therefore dictate the type of services our system must provide. In Figure 1, a UML Use Case diagram is shown to depict the functional requirements as desired features of the EWS. This diagram provides a model of the interactions between the EWS and the entities which will interact.

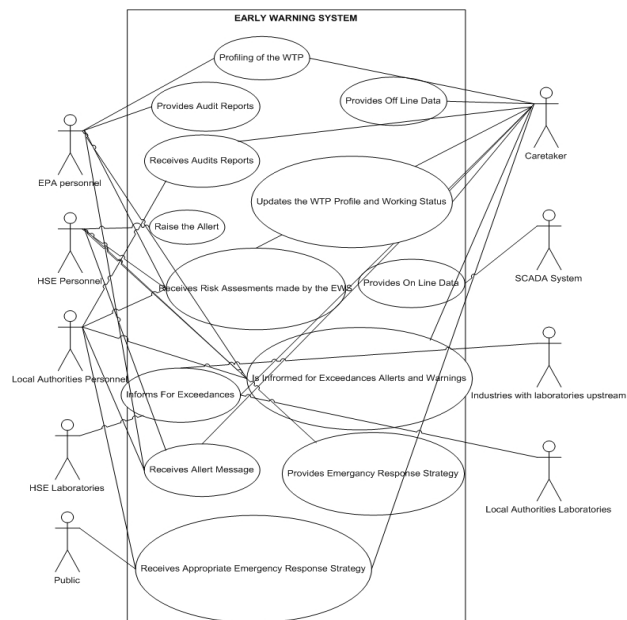


Figure 1. Functional requirements

### 3. Concepts

#### 3.1. Cloud Computing

Cloud computing was built on top of successful Web technologies such as virtualization, distributed computing, utility computing, Service Oriented Architecture (SOA) and software services [27]. The architecture of cloud platforms consists of hardware and software services stored on Web servers and accessible via the cloud. Cloud computing encompasses terms such SaaS, Infrastructure as a Service and Platform as a Service all aiming at cutting down the costs of buying, maintaining, supporting and updating IT infrastructures [2]. Services offered through the cloud can be broken down into three categories [5]:

1. Attached services usually contain applications that run on premises and provide extra services available in the cloud. Examples of applications that fall in this category are Windows Media Player, Microsoft's Exchange Hosted Services and Apple's iTunes.
2. Cloud platforms offer the framework to build applications that will run on the cloud. Examples of cloud computing platforms include Amazon Web Services, Google's AppEngine, Salesforce.com's Force.com and Microsoft Azure Community Technology Preview.
3. SaaS: These are applications designed to be accessible over the cloud and whose features are provided as services. As this is the bulk of this paper, it will be covered in depth in Section 4.

#### 3.2. Web Services

As defined by the World Wide Web Consortium [28], *a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL [Web Services Description Language]). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.* This definition only addresses a specific class of Web services: SOAP-based Web services.

In [20], SaaS services were considered as Web services from both macro level (services provided over the cloud) and micro level (leverage Web services technologies to support integration). They also described SaaS as a class of complex Web services. There are two types of Web services: SOAP-based Web services also called "Big" Web services in [17] and more recently Representational State Transfer (REST) Web services.

##### 3.2.1. SOAP-based Web services

The technology stack of this kind of Web services is filled with SOAP, WSDL, WS-Addressing, WS-ReliableMessaging, WS-Security and so forth aiming at interoperability for both Remote Procedure Call and messaging integration styles. SOAP-based web services work by linking servers (also called service providers) to clients (also known as requesters) via interfaces written in WSDL in order to trigger the communication. In its simplest form, SOAP-based Web services can be described as follows: the client first requires a service from a server and an XML message is wrapped in a SOAP envelop with the required headers and sent over the network to the server. The server gets back to the client by sending another XML message wrapped in a SOAP envelop. The wrapping of the XML message into a SOAP envelop is achieved by a SOAP engine. As XML and SOAP are both cross-platform languages so is the whole mechanism of exchanging messages across the network.

##### 3.2.2. RESTful Web services

REST is a term coined in [9] describing an architecture style of networked systems. RESTful Web services whose continuous popularity can be explained by their reliance on Web 2.0 services and also their alleged simplicity in being published and consumed [26]. Unlike Big Web services, RESTful Web services do not hinge on SOAP and WSDL and are instead designed to be used alongside HTTP. They are based on four HTTP methods: GET used to obtain the current state of a resource in a given representation, PUT to build a new resource, DELETE to get rid of an existing resource and finally POST to alter the state of a resource. According to [16], RESTful web services were made around four grounds illustrated in Table 1 :

**Table 1. RESTful Web Services principles**

Principles	Descriptions
Resource identification via URI	A bunch of resources needed by prospective clients is made available.
Interfaces	Resources are manipulated through GET, PUT, POST and DELETE
Messages	Resources are separated from their representations in order to be accessed in different forms.
Stateful communication via hyperlinks	The communication between computers is stateless hence request messages should be self-contained.

Despite their architectural disparity, it is possible to combine these two classes of Web services. This approach was tackled in [18]. The method works by accessing the functional components of any software located in a remote computer while using the power of distributing computing via a straightforward procedure.

### 3.3. Service Oriented Architecture

SOA is an architectural and design style for application development and integration aiming at achieving loose coupling among interacting software components or applications [13].

As reported in [14], SaaS and SOA are sometimes used interchangeably in the literature. This apparent confusion could lead to poorly designed applications. It should be noticed that these two tenets differ from one another by the fact that SaaS denotes a deployment model of applications whereas SOA is an architectural style used to develop distributed applications.

### 3.4. Software plus Services

Software-plus-service (S+S) is a mixture of SaaS, SOA and Web 2.0 that extends its platform into the cloud and allows combination of local and Web-based services with multiple device client and server software enabling a more flexible approach than software or service only tackles. Microsoft has set out inclusive strategic path by combining Web services with client and server software delivering exciting new capabilities. Through S+S, customers can benefit not only from SaaS or on-premises software but from the whole range of options enabled by both. Thus, S+S gives to its customers the best of both worlds.

The S+S approach allows developers to easily implement and support applications either based on Web applications by taking the advantage of opportunity provided by the Web or by taking the advantage of the rich experience that is available when software runs on a PC or device, providing the flexibility to extend the capabilities of server software or on-premises and hosted solutions [19].

### 3.5. Application Service Provider

Like SaaS, Application Service Provider (ASP) is a business model that gives to its customers the ability to use applications without actually owning them or being aware of the required infrastructure to run them, thus providing the services accessible over distributed servers on the Web with the convenience of the following benefits : no installation, accessibility, easy sharing of data with other users and co-workers [10]. Thus, leasing software applications ranges

from management software to document management systems and many others to businesses and consumers over the Web [11].

In the ASP business model, the service provider is also liable for application system updates, technical support and security for one application running for one customer on dedicated server and middleware stack at an ASP for a customer to be accessed via a network such as the Web [15]. Running one application for one customer on dedicated IT infrastructure, ASPs face the same problems as traditional on-premises software such as if a new customer wants to use the same application then it is only possible by setting up a new instance of application including the corresponding IT infrastructure [22].

Although most of the SaaS characteristics have been inherited from the ASP business model, the main difference between these two business models is the multi-tenant feature offered by SaaS. Indeed, several SaaS end-users share the same application instance and the application behaves for each tenant as if it was a separate instance of that application. Hence, whereas ASP concentrates on the provision of solution by providing a software application, SaaS on the other hand focuses on providing the entire support and management infrastructure [4].

## 4. An Overview of SaaS Applications

SaaS applications are deployed as hosted services and accessed over the cloud. The idea of outsourcing software has been first explored in [3] through several experiences. According to [23], the market share of SaaS applications is set to be more and more appealing as many companies are gradually considering investing on SaaS applications. SaaS big players include Microsoft, IBM, Google, Amazon as well as independent software vendors including Oracle. SaaS applications offer various services at a reduced cost (sometimes free) compared to on-premises applications. SaaS applications are particularly suitable for distributed applications with users living in remote locations.

There are also several other companies involved in the SaaS market. Some of them include Salesforce.com Inc. which was a pioneer in providing CRM SaaS applications, Taleo Corp. which uses human resources SaaS applications, SAP AG also uses a SaaS application called Business By-Design to offer to small-to-medium size businesses a suite of IT business services and RightNow Technologies Inc. is also involved in the CRM market.

Compared to its on-premises counterparts, SaaS presents several benefits including: reduced Total Cost of Ownership, lower deployment, maintenance and development costs, less financial risk, easier upgrades and no up-front investment. In [8], a game theoretical approach is used in order to work out the short- and long-term competition be-

tween SaaS and on-premises software providers. It is argued that although SaaS providers are liable for maintenance costs of their applications, these costs may affect the quality of SaaS applications.

#### 4.1. Benefits of SaaS over on-premises software

The increased popularity gained by the SaaS business model could be explained by the following benefits over on-premises software.

- **Security:** Because SaaS applications are run from providers' premises, vendors are liable for security and monitoring issues. Indeed SaaS providers' servers regularly back up customer's data in order to avert any data loss. The most important point to note here is the fact that all backup processes happen behind the scene and completely unbeknownst to customers;
- **Seamless installation and running processes:** On-premise software suffers from a serious setback related to their installation process. Indeed, that process can last for long depending on the actual size of the factory. On the other hand, SaaS applications only require a Web browser and an active internet connection;
- **Versatility:** Keeping track of all updates is essential to ensure the success of companies and also to make sure that they are equipped with the latest version of their tools. With the SaaS model, these updates are installed on the provider's computers. That process is usually completed within a very short amount of time which is a great improvement of the shortcomings of on-premises software;
- **Costs:** Because SaaS applications are paid on a pay-as-you-go basis, providers are more focused on the quality of service and support. Indeed, SaaS providers must always offer good products with good quality.

#### 4.2. Characteristics of SaaS applications

Unlike on-premises software which requires an up-front license fees, the pricing option of SaaS includes the license and hosting fees that are paid on a per-use basis by customers. SaaS applications are entirely hosted and maintained on its provider's data center. Therefore, customers are not required to purchase additional hardware to run the application. Table 2 summarizes SaaS characteristics.

#### 4.3. The multi-tenancy property of SaaS

Unlike ASP applications whose several instances were recorded at a given time, only a single instance of a SaaS

**Table 2. Characteristics of SaaS**

Features	Descriptions
Costs	License and hosting fees are alike.
Accessibility	The application must be accessible through a Web browser or a thin client. This ensures the seamless deployment and the ubiquity of the application.
Configurability	Tenants should be allowed to tailor the application in order to meet their needs.
Hosting	The application is fully hosted on the provider's infrastructure or data center.
Scalability	This brings about concurrency issues and the need of optimizing application resources including locking, network connections and database management.

application runs at a time and the application can accommodate several users that share a single database of their data. This feature is known as multi-tenancy.

SaaS applications must be by essence multi-tenant oriented as pointed in [6]. This means that some facilities must be transparently shared between many users and logically appeared to be unique to each user. For instance, as soon as a customer is granted access to his personal data through a SaaS CRM application, the particular instance of the application that the customer is hooked up to may also be simultaneously dealing with several other customers. The underlying mechanism must be hidden to customers. Hence, the architecture of the application must be robust and secure enough in order to deal with data belonging to different customers.

### 5. Relating SaaS to other technologies

SaaS applications have a better return on investment than their on-premises or ASP counterparts, therefore their services are provided for a nominal fee or sometimes free. In the case of free services, the provider usually recoups its investments from advertisement deals. ASP applications were not customizable. This implies that they are used to provide *one-size-fits-all* version of complex applications such as enterprise resource planing and CRM. SaaS unlike ASP applications are multi-tenant therefore, a single instance of a SaaS application can accommodate several end-users.

On the other hand, modern SaaS applications are fully customizable therefore their customers can change the user interface, settings and functionalities of the SaaS applica-

tion in order to meet their needs and wants. Although, ASP solutions were hosted by a third party company, users were required to purchase an up-front license fee similar to legacy on-premises applications.

Another noticeable difference between these two business models is their provision granularity. Indeed, while ASP aims at larger-grained and more standardized applications, SaaS champions the aggregation of fine-grained and customized services.

The relationship between SaaS and other cloud computing technologies is illustrated in Figure 2. As we can notice on that sketch, business models such as ASP and SaaS can be developed by making use of Web services (RESTful or SOAP-based) and architecture models such as SOA. In [24], the architecture of a SaaS application was presented that makes use of SOAP-based Web services and SOA that is the cornerstone of SOAP-based Web services.

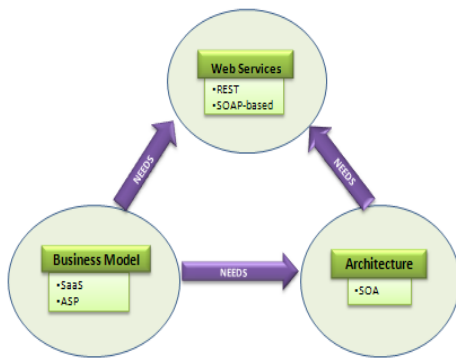


Figure 2. SaaS, ASP, Web services and SOA

## 6. Proposed high-level architecture

Application integration in our system will be addressed by the means of both RESTful and SOAP-based Web services whereas alert messages will be provided by the means of RSS or ATOM feeds. The core components of our system can be described as follows:

- Prospective users include the EPA, the HSE, local authorities, water treatment plant managers and their respective staff spread over Ireland;
- The presentation layer consisting of a security module that will be responsible of defining policies to handle users' authentication and authorization and access control to our system's vital resources, an interactive rich client application that will provide a domain-specific modeling language that in turns will help users assess the likelihood of failure of water treatment plants.

- An application layer that will provide the core mechanism of the overall system such as the actual EWS and the required tools to profile and monitor water treatment plant operations. The access control module will be liable of the policy to handle the specific role of users within an organization;
- A multitenant database that will store all information related to the operational process of water treatment plants, their possible causes of failure, the users' credentials. This layer will also be enhanced by a security mechanism whose aim is to protect users' data from being violated and corrupted.

Figure 3 illustrates the high-level architecture of our system.

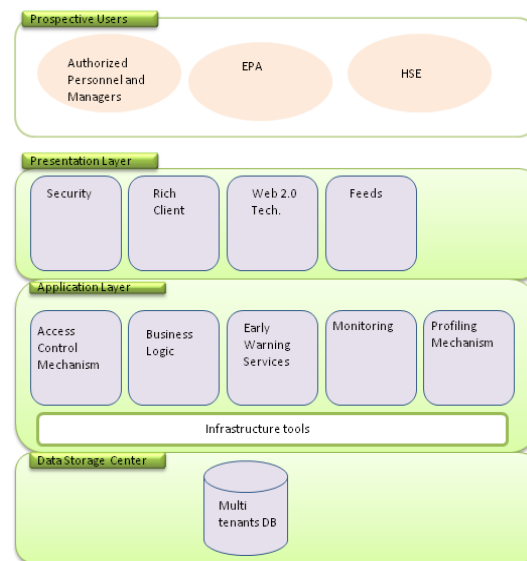


Figure 3. High level architecture

## 7. Research Issues

The research challenges in our project were recorded in many areas. These range from application integration to connecting all stakeholders of our system as well as putting together Web technologies and domain specific language tools in order to provide an EWS to monitor drinking water treatment plant operations in Ireland. Other issues include the multitenancy property of our application, configuration and customization and also data security.

- **Data integration:** The issue is highlighted by the need to provide a mechanism to link all actors and stakeholders of our system. From a developer perspective, this problem is raised by the necessity to build

a SaaS application with interfaces using various protocols, standards and formats.

- **Customization and configuration:** As reported in [21], software customization and configuration is not a new problem faced by software practitioners. They also noticed the lack of a complete study and guidelines on customization and configuration issues of SaaS that could help a prospective SaaS vendor to effectively schedule his plans to address these shortcomings. Our system should allow its end-users to change its settings as well as its interface in order to meet their needs;
- **Multitenancy:** As only one instance of our application will serve all users, addressing the security of users' data become a paramount concern. Indeed, distributed applications require their providers to protect sensitive information from being accessed by an unauthorized user. However the publicity of the Web makes things a lot harder to manage. As reported in [1], there are still many issues related to the implementation of a multi-tenant database;
- **Data ownership:** SaaS providers face a challenging task when it comes to clarify the ownership of customers' data. Indeed, the SaaS provider might be a third party company that actually rents some disk space to another company. Hence, the need to define a proper safety data protocol. Another important issue is the need to recover customers' files in case of an unexpected event such as an unwanted deletion of files.

## 8. Previous Works

Previous works related to our project were investigated in three directions: the applications of the SaaS business model in real world projects, any previous EWS related to drinking water management and finally any EWS deployed over the Web.

First of all, we found that the SaaS business model has been successfully applied to several applications including customer relationship management with Salesforce.com being considered as the most sterling example. The incentive behind our decision to choose the SaaS business model as a deployment model for our system was justified by its characteristics presented in Section 2.

Secondly, our literature review has identified only one representative example of an EWS in the domain of water treatment and supply. The WaterSentinel system [7] is the result of a program developed by the U.S. EPA for designing, deploying, and evaluating a model contamination warning system for timely detection and appropriate response

to drinking water contamination threats and incidents that would appear to have broad applications to drinking water utilities.

We argue that it is feasible to develop a novel type of EWS. Its main goal is to assess the possibility or the likelihood of occurrence of any operational problems and failures in a timely manner and through the use of IT and Web technologies to assess and inform stakeholders about any risk of a drinking water contamination incident. The novel characteristic of our approach is that, in order to solve the problem, we adapt a socio-technical perspective that includes:

- Technical components of the water utilities (e.g. software, hardware, electronic and mechanical systems);
- Actors (e.g. personnel and workers);
- Social elements (e.g. social rules, policies and procedures).

Given that our approach takes into consideration the need to monitor not only critical operational parameters in the water utility, but also critical parameters of the components of the wider socio-technical system in which such a utility is embedded. Contrary to the WaterSentinel system, the aim of our proposed service is to make the boundaries of safety performance of water utilities visible to the stakeholders and decision makers and to help them reduce the forces that drive water utilities beyond their safety boundaries.

## 9. Conclusion and Ongoing Work

Throughout this paper, we have described our innovative idea of using the SaaS business model to deploy an early warning service. We have also elucidated the differences between the ASP business model and its successor as well as the main advantages of the SaaS business model over on-premises software. We have also presented the architecture of our application that will be accessed over the cloud and whose functionalities will be made available to our various users as hosted services. A list of potential issues related to SaaS was also presented. Based on our literature review, we have realized that the usage of the REST technology in SaaS applications has not been discussed in detail. Thus, we decided to use the REST technology in our system in order to have a deep knowledge of the pros and cons of RESTful Web services in the SaaS business model. However, there are many other technical challenges standing on our way. At first, there is a necessity to select the right technologies and tools in order to develop our system. Secondly, implementing a multi-tenant database for our system is another issue in our project. Our next milestone will be achieved when all these issues are addressed.

## 10. Acknowledgments.

This paper is supported by the research project SCEWA (Grant No 2007-DRP-2-S5), funded by the Irish Environmental Protection Agency under the DERP grant scheme.

## References

- [1] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1195–1206, New York, NY, USA, 2008. ACM.
- [2] F. M. Aymerich, G. Fenu, and S. Surcis. An approach to a Cloud Computing network. In *First International Conference on the Applications of Digital Information and Web Technologies*, pages 113–118, Aug. 2008.
- [3] O. P. Brereton and D. Budgen. Component-Based Systems: A Classification of Issues. *Computer*, pages 54–62, 2000.
- [4] R. Butler. An analysis of service-oriented architecture (soa) to determine the impact on the activities performed by the external auditor of a soa service consumer. *Meditari Accountancy Research*, 16(2):13–30, 2008.
- [5] D. Chappell. A short introduction to cloud platform: an enterprise-oriented view. Available online at <http://www.davidchappell.com/CloudPlatforms-Chappell.pdf>, 2008. Accessed on 8 December 2008.
- [6] F. Chong and G. Carraro. Architecture Strategies for Catching the Long Tail. MSDN Library, Microsoft Corporation, 2006.
- [7] EPA. WaterSentinel Consequence Management Strategy. Technical Report EPA 817-D-05-004, US EPA, Water Security Division, Dec. 2005.
- [8] M. Fan, S. Kumar, and A. B. Whinston. Short-term and long-term competition between providers of Shrink-Wrap Software and Software as a Service. *European Journal of Operational Research*, 196(2):661–671, 2009.
- [9] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.
- [10] G. Flammia. Application Service Providers: Challenges and Opportunities. *IEEE Transactions on Intelligent Systems*, 16(1):22–23, 2001.
- [11] B. Furht, C. Phoenix, J. Yin, and Z. Aganovic. An Innovative Internet Architecture for Application Service Providers. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, page 6051, Washington, DC, USA, 2000. IEEE Computer Society.
- [12] P. H. Gleick. Water in Crisis: Paths to Sustainable Water Use. *Ecological Applications*, 8(3):571–579, 1998.
- [13] K. Holley, K. Channabasavaiah, and E. M. Tuggle. Migrating to a Service-Oriented Architecture. IBM DeveloperWorks, December 2003.
- [14] P. A. Laplante, J. Zhang, and J. Voas. Distinguishing between SaaS and SOA. *IT Professional*, 10(3):46–50, May 2008.
- [15] R. Mietzner. Using variability descriptors to describe customizable saas application templates. Technical Report 2008/01, Fakultät Informatik, Universität Stuttgart, January 2008.
- [16] C. Pautasso, O. Zimmermann, and F. Leymann. RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 805–814, New York, NY, USA, 2008. ACM.
- [17] L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly Media Inc., May 2007.
- [18] X. Shi. Sharing service semantics using SOAP-based and REST Web services. *IT Professional*, 8(2):18–24, March-April 2006.
- [19] H. Sirtl. Software plus Services: New IT- and Business Opportunities by Uniting SaaS, SOA and Web 2.0. *Enterprise Distributed Object Computing Conference, 2008. EDOC '08. 12th International IEEE*, pages xviii–xviii, Sept. 2008.
- [20] W. Sun, K. Zhang, S.-K. Chen, X. Zhang, and H. Liang. Software as a Service: An Integration Perspective. In *ICSOC '07: Proceedings of the 5th international conference on Service-Oriented Computing*, pages 558–569, Berlin, Heidelberg, 2007. Springer-Verlag.
- [21] W. Sun, X. Zhang, C. J. Guo, P. Sun, and H. Su. Software as a service: Configuration and customization perspectives. *Congress on Services Part II, 2008. SERVICES-2. IEEE*, pages 18–25, Sept. 2008.
- [22] L. Tao. Shifting paradigms with the application service provider model. *Computer*, 34(10):32–39, Oct 2001.
- [23] E. Tenwolde. A Preliminary Look at Delivery Model Performance. Technical Report IDC No. 206240, IDC, 2007.
- [24] M. Turner, D. Budgen, and P. Brereton. Turning Software into a Service. *Computer*, 36(10):38–44, 2003.
- [25] UN. The Millennium Development Goals Report, Aug. 2008.
- [26] S. Vinoski. Serendipitous reuse. *IEEE Transactions on Internet Computing*, 12(1):84–87, 2008.
- [27] M. A. Vouk. Cloud Computing - Issues, Research and Implementations. In *30th International Conference on Information Technology Interfaces*, pages 31–40, Cavtat, Croatia, June 2008.
- [28] W3C. Web services. Available online at <http://www.w3.org/TR/ws-gloss/>, 2009.
- [29] WHO. Water, Sanitation and Hygiene Links to Health, World Health Organization. Facts and Figures, Mar. 2004.