Revista Brasileira de Computação Aplicada, April, 2019

DOI: 10.5335/rbca.v11i1.8051
Vol. 11, Nº 1, pp. 2–13
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ORIGINAL PAPER

A case study on a service-based information systems integration in academic environment

Ramon Abilio[1], Cristiano M. Garcia[1,2], Flavio Morais[1] e Antonio E. R. Neto[1]

[1]Information Technology Department, Federal University of Lavras (UFLA) - Brazil e [2]Graduate Program in Systems and Automation Engineering, Federal University of Lavras (UFLA) - Brazil

*{ramon.abilio,cristiano.garcia,flavio,antonio}@dgti.ufla.br

Received: 2018-04-25. **Revised:** 2018-11-05. **Accepted:** 2019-01-17.

Abstract

In corporate environments, we can find various information systems (IS), which need to communicate to each other to share and maintain data consistency. Academic environments (AE) are even more complex than corporate environments because they have several IS to help manage different aspects, such as restaurant and library, which need to have consistent data to work properly. Therefore, it is necessary to encounter a form to develop an integration among them and share common, trustworthy data. We present a case study on SOA-based architecture for IS integration within AE to keep data consistent through the systems, to monitor the communication, and to make the integration safe and manageable. We applied the proposal, and the results show that we can integrate, monitor, and manage different software systems and network services and permissions. The main contribution is a useful integration architecture for AE that must share trustworthy data among several, heterogeneous IS and network systems. In addition, a small team can implement and maintain this proposed architecture.

Key words: Academic Environments; Information Systems Integration; Service-Oriented Architecture; Web Services.

Resumo

Em ambientes corporativos, pode-se encontrar diversos sistemas de informação (SI), que precisam comunicar entre si para compartilhar e manter dados consistentes. Ambientes acadêmicos conseguem ser ainda mais complexos que ambientes corporativos, pois, especialmente em universidades federais, há diversos aspectos a serem gerenciados, como restaurante universitário e biblioteca. Tais setores também precisam de dados consistentes para funcionar corretamente e não haver perda financeira. Então, é necessário encontrar um meio de desenvolver uma integração entre os SI e compartilhar dados comuns de maneira confiável e consistente. Neste trabalho, é apresentado um estudo de caso sobre uma arquitetura orientada a serviços para integração entre SI em ambientes acadêmicos para manter os dados consistentes através dos sistemas, monitorar a comunicação entre tais sistemas e tornar a integração segura e consistente. A proposta foi aplicada e os resultados mostram que é possível integrar, monitorar e gerenciar permissões de integração entre diferentes SI e serviços de rede. A principal contribuição é uma arquitetura de integração para ambientes acadêmicos que precisam compartilhar dados confiáveis através de diversos e heterogêneos SI's e serviços de rede. Além disso, uma equipe pequena de TI consegue implementar e manter a arquitetura proposta, já que não há necessidade de alteração de código de sistemas e serviços já existentes.

Palavras-Chave: Ambientes acadêmicos; Integração de Sistemas de Informação; Arquitetura orientada a serviço; Serviços Web.

1 Introduction

Information Systems Integration constitutes a common problem in corporate environments, as well as in academic environments. Studies have pointed out that, concerning academic environments, several Universities around the globe in fact struggle with integrating information systems (Andersson et al.; 2014; Costa et al.; 2010; Cunha et al.; 2008; Garcia et al.; 2015; Fu and Yuan; 2011; Suryawan; 2014; Bessa et al.; 2016; Putro and Rosmansyah; 2017). Universities, in general, have different necessities regarding their information system since they have departments that deal with administrative activities and departments that deal with learning, research, and extension activities. Furthermore, those information systems have to be integrated to share data among them and among network services composing an heterogeneous environment due to the diversities concerning, for example, technologies, DBMS, and platforms.

Corporate environments, in general, have different information systems to handle strategic, tactic, and operational information (Turban et al.; 2004). Besides, data may be shared among systems in different levels (vertical integration), and among different systems in the same level (horizontal integration). There is a number of approaches for Information Systems Integration (ISI), which can be classified, for instance, by technique-centric approach, such as "Centralized Information" (Martins; 2005), or by focusing the organization itself and its processes, for example "Business to Business (B2B)" (Hohpe and Woolf; 2003).

Concerning ISI, there are three important dimension of issues (Hasselbring; 2000): distribution, heterogeneity and autonomy. Distribution means the ability of a proxy service hide distribution and be presented as an only system, for example, by using Remote Procedures Calls (RPC). Heterogeneity concerns the different servers on which various systems, developed in a number of different languages, can execute. Finally, autonomy is related to the ability of a system decide how to communicate with external systems (Hasselbring; 2000).

Database features can assist at ISI by using direct connections to external views or, if distinct databases are on the same Database Management System (DBMS), the tools of the DBMS may be used, such as triggers, procedures, or even database engines, for example the Federated Storage Engine, present in MySQL DBMS. Other approaches, such as creating a unique global ID for entities (Suryawan; 2014) can be used, although it represents a big change in already settled / third-party databases. Simultaneously, other tools, such as Cron[1], may help by being set to trigger an update script.

Approaches such as Service-Oriented Architecture (SOA) can be used for ISI. Several technologies provide support for implementing SOA, such as J2EE, .Net Frameworks, Microservices (Dragoni et al.; 2017) and Web Services (He and Da Xu; 2014). When we use SOA specifically for ISI, we have Service-Oriented Integration (SOI) (Hensle et al.; 2010; OASIS; 2006). A way of implementing SOI is through Web Services, which represents a vision on distributed programming and resource offering, strongly linked to Internet (Technologies; 2001). Many patterns and technologies, such as Simple Object Access Protocol (SOAP), Representational State Transfer (REST) architecture, and Java API for Web Services (JAX-WS), provide support to develop Web Services.

ISI can be extended to academic environments as they may be even more complex and heterogeneous than corporate environments (Alkhanak and Mokhtar; 2009; Andersson et al.; 2014; Costa et al.; 2010; Fu and Yuan; 2011; Suryawan; 2014). Systems most commonly used in corporations, e.g. Enterprise Resource Planning, frequently contain modules that embraces the major activities in the company in an only system. On the other hand, universities have different departments and services, for instance restaurant, academic control department, wireless, e-learning platforms without evident relation among them. However, these systems and services have at least one thing in common: they do need trustworthy, consistent and updated data.

The Federal University of Lavras (UFLA) is an example of that academic environments that has complex and heterogeneous information system integration (Garcia et al.; 2015). UFLA has, approximately, 20 information systems (IS) maintained by the Information Systems Coordination (ISC), which is subordinated to the Board of Information Technology Management. Those information systems have peculiarities regarding their: goal (academic, administrative, and support); DBMS; platforms (desktop, web or mobile); and technologies. Furthermore, some of those IS have been developed by ISC and others have been developed by thirds (companies or professors with their students). In addition, the University has network services, such as e-mail and wireless internet available to the whole academic community via user authentication.

This diversity of IS and network services makes the integration challenging. Therefore, in this paper we present a case study on a SOA-based integration architecture designed to be scalable, flexible, monitorable, and to have mechanisms of security. A small team with 3 professionals (database managers and developers) implemented the proposed architecture at UFLA, making the integration more reliable and secure. After that, they could identify, for example, problems in the information systems integration using the management and monitoring system developed by them. The main contribution of this work is an integration architecture that can be implemented by small teams and can be used in academic environments that have a diversity of information systems and network services.

The rest of the paper is organized as follows. We briefly present concepts related to information systems integration and SOA in Section 2. In Section 3, we define the problem and, in Section 4, we present the integration architecture proposal, covering a number of aspects. We implemented the architecture in an University and we report and discuss the results in Section 5. The related work is presented in Section 2.3 and the Conclusion is presented in Section 6.

[1]https://help.ubuntu.com/community/CronHowto

2 Background

The company that uses Information Technology (IT) in an efficient way, integrating IT strategies to business strategies, is able to gain competitive advantage (Laurindo et al.; 2001). In these companies – specially in Universities –, it is common to find scenarios with many information systems, in which data must be shared in order the preserve the consistency among all IS and network services and eventually to constitute an Information System Integration (ISI). One of the most popular techniques for ISI is by using SOA. This section briefly presents approaches of information systems integration and concepts related to the Service-Oriented Architecture (SOA).

2.1 Information Systems Integration Approaches

Several reasons may instigate companies to contract or develop information systems integration solutions. These reasons may be (Degan; 2005): to extend from the existent technology, to reduce costs and time on implementation of new services; to allow integration with stakeholders, expanding the range of services; and integrate common information in different databases, outcomes of fusions, acquisitions or legacy systems. Information Systems Integration constitutes a risky task, as each organization has its own characteristics and integration needs (Martins; 2005).

An organization can integrate systems taking into account approaches with different focuses, such as implementation or on the company and its processes. Regarding implementation level, an information systems integration can be classified as (Martins; 2005): a) Composite Applications: applications integrated via API, that works as a connector between systems; b) Centralized Information: information systems have access to the same database sharing data and metadata; and c) Integrated Management Systems: closed systems and formed by independent, internal modules. This type of integration is implemented, commonly, in the source-code level.

Focusing on the organization and its processes, we have, for instance (Hohpe and Woolf; 2003): a) Data Replication: integration in the information level, having distributed, synchronized and updated databases; b) Business-to-Business Integration (B2B): surpass companies' boundaries. Represents functionalities offering between different organizations. Although the other concepts mentioned here may be applied to B2B, the utilization of external networks may arise new aspects to be analyzed (Hohpe and Woolf; 2003); and c) Service-Oriented Architecture (SOA): systems offer functionalities as services, that means, in computing science context, "function well-defined and universally available".

Although different focuses, the taxonomies have similarities among themselves (Hohpe and Woolf; 2003; Martins; 2005). For instance, the approaches Business-to-Business Integration (B2B) and Composite Applications may use API to sharing information and accomplish an integration among information systems. To realize the integration, we can use SOA implemented by using Simple Object Access Protocol (SOAP) or Representational State Transfer (REST), and Extended Markup Language (XML) or Javascript Object Notation (JSON).

2.2 Service-Oriented Architecture

Software architecture represents a structure that comprises software components, their externally visible properties and the relation between both (Pressman; 2001). Service-Oriented Architecture (SOA) have been considered one of the principal paradigms in distributed systems design leading to a ramification in Software Engineering, so-called Service Software Engineering (van den Heuvel et al.; 2009). SOA constitutes a paradigm that aims to organize and utilize resources, that may be under control of different owners, by providing a uniform means of offering, discovering, and interacting with functionalities used to produce the desired and consistent effects.

This paradigm can offer several benefits, such as control of systems growth, global-scale services offer and utilization and cost reducing in business-to-business cooperation (Valipour et al.; 2009). By using SOA as an integration means, there is the Service-Oriented Integration (SOI), which principal aim is to create an integration among multiple systems, changing little or nothing their implementations (Hensle et al.; 2010). This technique exposes data, functionalities and processes to be consumed by systems of integration. There are several approaches for SOI focusing on existent systems (Hensle et al.; 2010): a) Service: it uses a service layer between existent systems and service consumers; b) Process Integration: to integrate processes in a corporate environment, suggesting the utilization in the integration of small processes within big processes, having human interaction or not; and c) Data Integration: it regards an approach to manage data models' complexity in different applications.

Web Services (WS) constitute a means of implementing SOI providing a service interface that allows consumers to interact with service providers (Coulouris et al.; 2013). The most common manner to implement WS is by using Simple Object Access Protocol (SOAP) as protocol or Representational State Transfer (REST) as architecture of communication, also mentioned as solution for processes integration among organizations (Zur Muehlen et al.; 2005).

SOAP is a protocol that uses Web Services Description Language (WSDL), an XML-based document, to describe functionalities offered by a WS (Zur Muehlen et al.; 2005). SOAP provides a basic standard of communication, in which each operation is represented by its terminal, described in the XML sent in the request, instead of a method HTTP as in REST architecture (Zur Muehlen et al.; 2005).

REST is an architecture (Fielding and Taylor; 2002) which constitutes an abstraction of principles that makes World Wide Web (WWW) scalable (Zur Muehlen et al.; 2005). It allows offering services identified by a Uniform Resource Identifier (URI), for example http://www.mysite.com/companies. HTTP methods such as GET, POST, DELETE and PUT, define the operation to be executed on a record or a set of records. For example, GET obtains, POST inserts,

DELETE removes and PUT updates data (Fielding and Taylor; 2002).

By using REST the responses can be returned in accordance with the application requirements, for instance, by using different types of response or formatting, such as Javascript Object Notation (JSON) or XML. Conversely, SOAP defines a response structure that must be respected (Gorski et al.; 2014).

2.3 Related Work

Information Systems Integration in academic environments has been studied along the years and a number of proposals have been made (Alkhanak and Mokhtar; 2009; Andersson et al.; 2014; Costa et al.; 2010; Fu and Yuan; 2011; Suryawan; 2014; Putro and Rosmansyah; 2017).

In the Açores University (Portugal), a set of Web Services was developed aiming to optimize critical tasks, involving financial and strategic information (Costa et al.; 2010). Risky administrative tasks, such as scholarship distribution and discounts for flight tickets (relevant here as the principal means of reaching Açores is by aircraft) - that required information correctness and velocity, - used to take several days to be checked and validated, as the university's employees resorted to fax, telephone, mail and electronic mail to perform these tasks (Costa et al.; 2010).

In the Federal University of Pelotas (UFPel - Brazil), it was developed and deployed a set of Web Services, named Cobalto Webservice, to keep information consistency among systems and services, such as university's restaurant, e-learning environment and wireless (Andersson et al.; 2014). This project, by the time of publishing, was still ongoing, as 60% out of all systems in UFPel already used Cobalto Webservice, and the other 40% of systems were scheduled to be migrated (Andersson et al.; 2014).

An ISI solution was deployed in an Indonesian University, based on a global unique ID, that identifies a person in the Institution and is used as a synchronization key (Suryawan; 2014). This key is copied among the tables in the databases of the integrated systems. One of the applications is defined as a Single Source of Truth (SSOT) and the synchronization is scheduled by using Cron. The model does not present resources to recover from failures during synchronization. The integration script by accessing Web Services or direct access to database, the author affirms that Web Services are utilized only for simple and occasional tasks. For more expensive tasks, the update is done by using direct access to database, as in case of network failure, the instruction keeps running on the DBMS. Yet according to the author, this is a low-cost solution and it is considered a palliative solution, until the development of a new definitive, robust solution (Suryawan; 2014).

In a University in China, an ISI platform has been developed for 6 years in order to improve and safely ensure infrastructure, account, application, privileges, processes and data integration, and a case study is presented (Fu and Yuan; 2011). This platform has solved a number of ISI issues, such as a better control access and authentication management (Fu and Yuan; 2011). Although it seems interesting, there is a lack of important details, as it presents an entire platform but little part is shown in the case study.

In a Malaysian University, it was developed a set of Web Services in order to convert some applications into services and make data available (Alkhanak and Mokhtar; 2009). In this mentioned work, it was shown the importance of integration and data availability to the academic community through SOA. At first, the students were facing problems in using data and services of the institution. With the adoption of integration of the services, students could rely more on data provided by the colleges, with increased speed and low costs implementations (Alkhanak and Mokhtar; 2009).

The work developed in Portugal (Costa et al.; 2010) presents a very specific, limited integration, as it seemed to fit only their needs. Another work, developed in Brazil (Andersson et al.; 2014), presents a generic Web Service, intending to integrate the various systems. However, in the work it is not mentioned if they count on security methods, or a monitoring system, for example. The work performed in Indonesia (Suryawan; 2014) presents a low-cost database integration, which can be very expensive while changing data models, specially in third-party databases. The integration is very simple, and the author emphasizes that his approach it is not a definitive one, and thus, a more robust, which would take longer and more knowledge, should be developed. Our work provides a generic, scalable manner to integrate a number of systems, in a safe and monitorable form.

Putro and Rosmansyah (2017) presented a proposal of a enterprise service bus (ESB) to deal with smart educational services. Their proposal is based on SOA, web services e microservices technologies and its main functionality is to route, transform protocols, and transform messages or data due to compatibility among services and information technology resources. Our architecture can be used with smart educational services acting as an ESB and also dealing with data integration of other software systems and network services.

Garcia et al. (2015) reported in their paper how the software system integration was performed at UFLA before the adoption of a SOA-based integration. In June/2015, a mobile application was developed and integrated to the other software systems using SOAP and XML in the communication. However, it was necessary the development of a REST layer to reduce data processing on the mobile application. (Garcia and Abilio; 2017) studied the impact of that layer regarding time response of the requests. This work differs from Garcia et al. (2015) and (Garcia and Abilio; 2017) since it presents a consolidated architecture used at UFLA that has been studied and improved since 2014. In this work, we present an abstraction of our architecture proposal that can be used as a reference by other IT teams.

3 Problem Definition

In general, Universities are based on teaching, researching, and extension activities. To support them, universities have departments to deal with

administrative tasks, e.g. human resource, and university planning and management. Moreover, they provide services for their students, professors, and employees, such as restaurant, library, e-learning environment, and academic e-mail. In addition, there are several software systems to support the activities and services in the university. These software systems are commonly developed by different companies, storing data in distinct Data Base Management Systems, running on different Operating Systems.

Therefore, we have a number of different processes and information systems (IS) to support them. Those IS have to be integrated to eliminate, for instance, duplicated work among the departments and inconsistency data among the IS. In this integration, some IS connect in other systems to authenticate an user, obtaining few data in response, or periodically synchronize more than thousands of registers in each connection (batch update).

We can identify the following characteristics of this scenario: a) IS used by different business areas (academic, administrative, and support); b) IS and network services based on different technologies and hosted in different servers; c) IS of different companies; and d) The need of IS for sharing different amount of data.

We also have to consider aspects, such as: a) Security: only authorized applications can have access to the data because personal data of thousands of people may be handled; b) Maintainability, Flexibility and Scalability: new or different information systems and network services may be added, changed, or removed, for example, due to business rules and information technology changes; and c) Monitorability: it is necessary to monitor the integration to detect, for instance, problems in the network connection or unnecessary data synchronization.

This scenario is real for universities in Indonesia (Suryawan; 2014) and it is present in Brazil (Andersson et al.; 2014; Cunha et al.; 2008; Garcia et al.; 2015) and Açores (Costa et al.; 2010). Our proposal aims to deal with different systems / technologies, running on different servers, due to the heterogeneity present in this sort of environment.

4 Proposed Architecture

We propose an integration architecture addressing the characteristics and aspects that we identified in the Problem Definition. This integration architecture can be classified regarding the implementation-centric vision as a "Composite Applications" approach (Martins; 2005) due to the integration by using SOAP. Regarding the strategies, it focuses on the organization and its processes using a Service-Oriented Architecture and Data Replication (Hohpe and Woolf; 2003) because the systems provide functionalities by means of services, that can be consumed by clients. Those functionalities may deal directly with the databases involved in the integration. In this Section, we discuss the main aspects of the proposed architecture.

4.1 Maintainability, Flexibility and Scalability

Regarding maintainability, flexibility, and scalability, we have to analyze the services and providers granularity. A balance between coarse-grained (when it is involved too much data) and fine-grained (impacting in round-trip requests to maintain a single record, for instance) granularity is recommended in order to prevent a number of problems, such as service duplication, hard maintainability, and SOA principles breaking (Kulkarni and Dwivedi; 2008).

Therefore, we discussed the question "Is it better only one provider containing all services, or some providers containing some services?". We selected the second option due to scalability, isolating, and grouping the services by specific systems. This may improve the governance, auditability, and maintainability of services (Kulkarni and Dwivedi; 2008). With different providers grouping related services, the maintainability and flexibility are increased because when a provider or its services need to be created or changed, the provider can be initiated or stopped without interfering on the other providers.

4.2 Security

The architecture uses SOAP as communication protocol and JSON as response protocol. SOAP had been chosen because the client has to know the operations of the provider. SOAP counts on specifications such as WS-Security, which aims to improve SOAP messages, providing three principal mechanisms (Lawrence et al.; 2006): i) sending security tokens as part of the message; ii) message integrity; and iii) message confidentiality. We used JSON because we can transform objects into text in the response, and at the same time not mixing SOAP structure and the structure of the response, as it would happen if both would use XML. In the proposed architecture, the responses must contain a JSON object with four attributes:

i. ID: the ID of the message with the ID of the Provider, in order to improve the debug of eventual failures;
ii. MESSAGE: it contains a textual description or an object of the outcome for the action that was requested;
iii. TYPE: it indicates the type of message, for instance, if occurred a "SUCCESS" or an "ERROR_DB";
iv. SYSTEM: this attribute brings the name of the requested provider.

Figure 1 shows an example of a JSON object with the four attributes filled. In this example, the attributes have the following values: 1) ID: 700_1.1; 2) MESSAGE: records requested by the consumer in the requirements elicitation; 3) TYPE: SUCCESS; and 4) SYSTEM: PRG was the provider requested by the consumer.

In order to prevent unauthorized access, an authentication approach using a Token and IP address can be used. That is, each authorized system has a

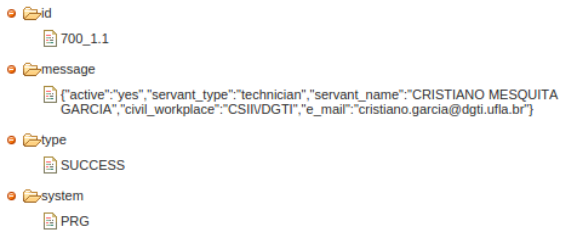

Figure 1: JSON-formatted response of the Web Service

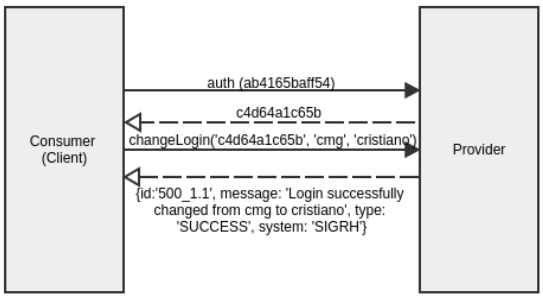

Figure 2: Example of interaction between a Private Consumer and a Provider

Token and each token has a list of IP addresses or complete subnets from which the requests can origin. Therefore, when a system initiate its authentication, it provides its Token and the provider checks the IP of the client. If both match, the system is authenticated and can consume the services.

Figure 2 presents an example of the process in which a consumer needs to change a user login. The consumer starts calling "auth" service and providing its token ("ab4165baff54"). The provider verifies if the request came from an authorized IP and returns a session key ("c4d64a1c65b"). A request with session key is utilized to prevent unneeded accesses to database. The services that the token can access are loaded into the session and retrieved using a session key. After that, the consumer calls "changeLogin" service providing the session key, the current login and the new login. As a result, the provider returns a JSON object with the four attributes aforementioned.

To deal with a situation in which we have an integration among internal information systems (managed by the Information Technology team of the University) and external systems (software systems developed by thirds), we divide the providers in two categories named: Private and Public. Private providers are the ones that receive requests exclusively from internal systems or contains services that perform UPDATE or DELETE operations. On the other hand, Public providers contain single services that perform only SELECT operations consuming private services. In this way, we have only one public interface and we encapsulate private services protecting them from unauthorized access.

Figure 3 illustrates the situation in which an external client needs to change the user login and this update needs to be propagated to more than one system. Consumer calls "changeLogin" service from the Public Provider providing its token and the Public Provider checks the token and the IP of the Consumer. After that, the Public Provider acts like a Private Consumer (Figure 2) having its own token, authenticating itself in the respective Private Provider, and consuming the services. Therefore, the Public Provider receives requests and maps the requested services to the respective private services. This approach allow the encapsulation of services and provides only one public interface. That is, if the structure of the private providers change, or a new private provider is included, or an existing one is remove, the consumer will not be affected. The Public Provider can log and/or send an e-mail to the system administrator in case of failure.

4.3 Data Integration

It is common that legacy systems do not provide mechanisms to integrate with other systems. Therefore, an integration architecture can work as a connector among systems providing, consuming, and translating data. The proposed integration architecture provides, consumes, and translates data among systems by using SOA-based providers and consumers.

In the proposed integration architecture, we can have providers and consumers that act directly in the databases of the integrated systems or that provide and consume SOA-based services, when at least one system provides them. When the service have to change data directly in the database, it is needed to gather the requirements and study the metadata involved, contacting the software vendors in order to detect possible and undesirable side effects at updating data directly on the database. When the systems provide an integration interface, the negative impacts are minimized.

Providers and consumers can be developed for each software system and the execution of the consumers can be scheduled. Figure 4 shows five software systems with their providers and the respective consumers executed by Cron.

In order to maximize the data consistency among the systems, there is a database table where the data that should be propagated to the other systems must be inserted. This database table works as a buffer. The client of each system queries this table, in a predefined frequency, in order to obtain the data and to consume the provider of the same system, sending the data. Thus, both systems tend to be consistent among each other.

4.4 Monitorability

The integration should be monitorable in order to detect failures, such as network or database failures. It is also necessary a manner to compare the amount of requests among systems to detect the time of the day that the requests happen the most, and to detect undesirable requests. It is necessary a graphical data in order to be rapidly turned into information.

A Management System (MS) can be developed for monitoring and managing the providers, services, tokens, and requests. In its main window, MS should have a dashboard showing graphics that allow the

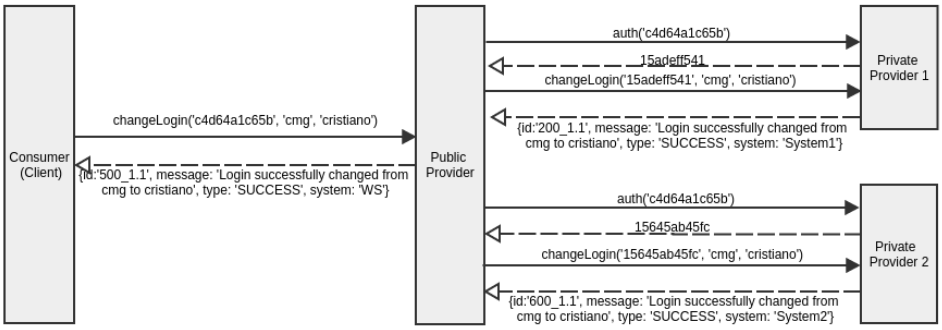

Figure 3: Example of interaction between Public and Private providers

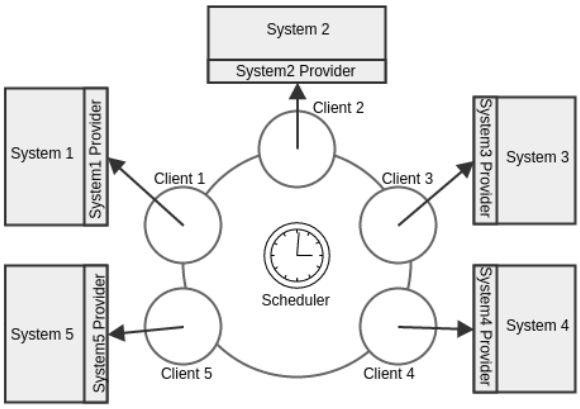

Figure 4: Integration Architecture

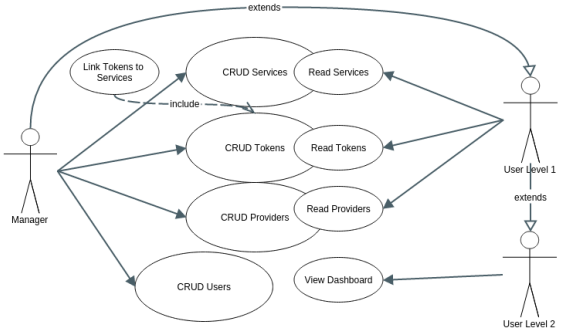

Figure 5: Use Case Diagram of the Management System

comparison of number of requests, success/error in the processing of requests, number of requests between current and last week, time of the last request, among others. MS should have CRUD (Create, Read, Update, and Delete) functions for tokens, and IP / subnet addresses from which a request can be origin (consumer's IP address).

MS has to provide a flexible management of the authorized systems avoiding configuration files to configure tokens and their IPs. In MS, the administrator can enable or disable a service in a graphical way. Furthermore, we can generate reports on the stored data related to the services and the requests. Services that check the status of communication between clients and providers are important. Additionally, when facing for instance a timeout, the service may send warning e-mails to users recorded in MS.

The Use Case Diagram (Figure 5) shows the MS actors and functionalities. MS has three actors (levels of access): 1) User Level 2 can only view the Dashboard; 2) User Level 1 inherits the permissions from User Level 2, and can also read Services, Tokens and Providers, but cannot update or delete them; and 3) Manager can maintain Services, Tokens, Providers, Users, view Dashboard and link Tokens to Services.

5 Applying the Proposed Architecture

The Federal University of Lavras (UFLA) has been passing through massive changes, as Brazilian government has encouraged the creation of new courses in Federal Universities and, thus, increasing the number of students and civil servants, including professors. Therefore, new needs started to emerge regarding information systems. UFLA used to have isolated software systems in each department, and due to its growth, it was detected the need for sharing information automatically among that information systems. Therefore, in this section, we present the scenarios before and after the development and deployment of the proposed integration architecture.

5.1 Scenario before the Proposed Architecture

Due to the growth of the University, in 2006, the first spark of ISI at UFLA appeared when data started to be shared among Cin-Cadastro, Zimbra, and LDAP, by using UDF to execute external scripts (Garcia et al.; 2015). UDF stands for User-Defined Function, functions programmed by a user, used as native function in MySQL, developed in C and compiled internally in the DBMS (MYSQL; 2005). A number of problems were identified when using UDF, such as database overload, as well as a problem related to a MySQL version update from 4 to 5.

Cin-Cadastro is a software system that allow the management of users of e-mail, wireless, and the generation of the Institutional Card. In 2008, ISC started the development of SIG-UFLA aiming the centralization of data of civil servants and students and providing functions to manage students and professors academic life (e.g., courses, disciplines, online enrollment, and reports). As SIG-UFLA was

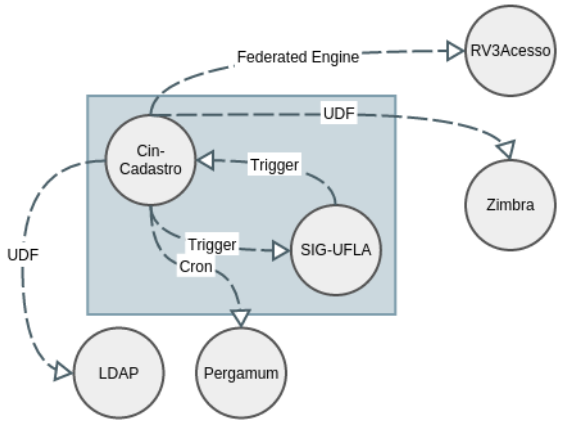

Figure 6: Integrated Software Systems before the Proposed Architecture

going to have its own base of users, it was decided to integrate SIG-UFLA and Cin-Cadastro avoiding duplicated work registering users in both systems. Although Cin-Cadastro centralizes the majority of data used by other systems, SIG-UFLA has the data of civil servants, students, and other people that have relation with the University.

Before implementing the proposed approach, only 4 systems and 2 networking services participated in the ISI (Figure 6): Cin-Cadastro, RV3Acesso, Zimbra, SIG-UFLA, Pergamum, and LDAP. The integration of the information systems and network services was implemented by using scripts scheduled in Cron and DBMS tools, such as triggers, UDF, and Federated Engine of MySQL (Figure 6).

Figure 6 shows that, in the integration of SIG-UFLA and Cin-Cadastro, it was used triggers, i.e., when data of a user are inserted or updated, an event is dispatched and Cin-Cadastro is updated. Pergamum and Cin-Cadastro were integrated using PHP scripts executed by Cron at scheduled time. Cin-Cadastro, Zimbra, and LDAP were integrated using UDF, but this approach was very difficult to maintain, as it is developed in C language programming and compiled in the DBMS. Furthermore, as the update volume was growing, the longer the update took to be concluded overloading the database server (Garcia et al.; 2015).

The integration approach presented in Figure 6 was difficult to manage and scale because: 1) when other system needed data, database views were made available for them; 2) it lacked a mechanism to monitor the integration; 3) there were several mechanisms to integrate the systems; and 4) UDF, trigger, and Federated Engine are dependent on the MYSQL version.

Afterwards, to reduce the negative impacts of UDF, Cron was utilized to execute external scripts, at a scheduled time. Although not simultaneously, the updates started to be executed without overloading the database. However, the scripts scheduled on Cron did not have any pattern or similarities among them, characteristic that makes the scripts difficult to maintain (Garcia et al.; 2015). In this scenario, the only form to make data available to external systems was through database access, mostly offering access to views.

5.2 Scenario after deploying the Proposed Architecture

In 2014, we studied the historic of the integration (Garcia et al.; 2015), and it allowed us to propose and test the architecture. In April 2015, we already had 12 software systems and 2 network services integrated using the proposed architecture. Table 1 presents the software systems with their programming language, DBMS, and platforms. For instance, the system SIG-UFLA was developed by using PHP, uses DBMS MySQL, and its platform is Web.

Table 1: Software Systems and Technologies

Systems	Prog. Lang. / Platform	DBMS
Cin-Cadastro	PHP/Web	MySQL
HCS	-/Desktop	MySQL
Merengue	Java/Web	MySQL
MinhaUFLA	Java/Mobile	-
Pergamum	Delphi/Desktop	SQL Server
	Java/Web	
PROEC	PHP/Web	MySQL
PRP	PHP/Web	MySQL
RV3Acesso	-/Desktop	MySQL
SIG-UFLA	PHP/Web	MySQL
SIGAA	Java/Web	PostgreSQL
SIGADMIN	Java/Web	PostgreSQL
SIGRH	Java/Web	PostgreSQL
SIP	PHP/Web	MySQL
SIPAC	Java/Web	PostgreSQL

Information Systems Coordination (ISC) has developed SIG-UFLA, MinhaUFLA and Cin-Cadastro, and there has been licenses purchases to use Pergamum, HCS and RV3Acesso. SIGAA, SIGRH, SIPAC and SIGADMIN compose a family of systems. They are developed and maintained by other Federal University and that University makes them available for deployment in other Universities. Those systems are under gradual deployment, but they provide and consume data from other systems. Merengue, SIP, PRP, and PROEC are software systems developed at UFLA, but we considered them as an external software systems because ISC does not maintain them.

Another software started to be integrated to the integration architecture from June/2015: a mobile application for institutional purposes (at first offering functionalities for undergraduate students), named MinhaUFLA. In order to reduce data processing on the mobile application, a REST layer had to be developed. This layer works by communicating with the MinhaUFLA Provider (that uses SOAP), translating data, converting the requests/responses from/to JSON before sending it back to the mobile application (Garcia and Abilio; 2017).

To implement the architecture in the University, PHP was chosen as programming language. PHP has native functions for SOAP requests, general database-handler classes, and has good perspectives

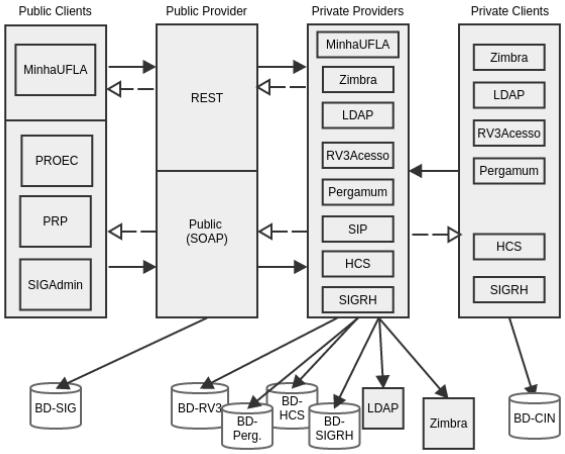

Figure 7: Communication among Public Clients (external systems), Public Provider, Private Providers and Private Clients (internal systems)

for future as its version 7 should be released in 2015, bringing significant advantages comparing to early versions, including performance. Furthermore, 80% out of the websites in 2017 (James; 2017) were developed in PHP, which means there are several options regarding servers to host web systems/sites that use this technology. Furthermore, PHP is cross-platform, and has, as of its version 5.1, PDO (PHP Data Object), a data-access abstraction layer that ease data access and future changes on data sources.

Figure 7 presents the implemented architecture. For example, Public Clients can request Public Providers, and Public Providers can request only one or multiple Private Providers, that access the databases and return the requested data. Private Clients can request Private Providers and they can perform the SELECT operation in the database of the Cin-Cadastro seeking for updates in records that will be propagated to other systems.

In order to trigger the clients with the need of recurring update, we used Cron configured to execute them at each 10 minutes. Additionally, to bring security, we utilized TLS, that is a protocol based on public key cryptography. This protocol, as well as SSL, is widely accepted as a protocol that can provide secure HTTP for internet transactions (Microsoft; 2003).

To manage and monitor the integration, we developed the Management and Monitoring System (MMS). MMS was developed by using PHP and has the uses cases described in the proposal (Figure 5). Figure 8 shows the MMS main window, in which we have a dashboard with 5 charts:

i. Percentage of Success and Failures: observing this chart we can notice that are occurring failures and then we can investigate their causes;
ii. Comparison of This and Last Week: with this chart, we can follow the use of the services;
iii. Proportion of Requests to Providers: this chart compares the amount of requests of each service;
iv. Top 5 Requested Services: observing the services more requested, we can try to optimize them seeking to minimize, for instance, the response time;
v. Requests in the last 24 hours: the chart allow us identify peaks of requests of a service and investigate why they occurred.

Figure 9 presents the status of services, together with the time of last request. When a Provider that was supposed to be updated in some frequency does not do that, after 1 hour from last request to checking time, the status of the Provider is signaled to red and the MMS starts sending warning e-mails to the responsible staff. Additionally, Figure 9 shows the time of last request (date and time) and if it is in production (globe icon) or development (PC icon) environment. The statuses that can appear in this window are online (green), offline (red), and disabled (gray). For example, HCS Provider is in production, and its last request occurred at 17:00:01 on 2015-04-27 and WS Provider is also in production, but its last request happened at 16:16:16 on 2015-04-27. The latter is signed as red because the last request to it occurred longer than one hour before checking, meaning that something went wrong with the client. In addition, LOG Provider (a hypothetical system, showed for example purposes), is an example of disabled provider, signaled as gray.

MMS has a use case to register Tokens and their IPs following the proposal of preventing unauthorized access by using an authentication approach that takes into account a Token and an IP addresses (Figure 10). For example, in Figure 10, the token "a5b075fea11b1f9afdf0f5011" can use the IP addresses "187.145.16.27" and "179.151.20.102" to access the Providers. There may also be registered complete subnets, such as the Token named "Department 1 Client" and subnet "177.150.48.%", where % means any number from 0 to 254.

After selecting a token in the list (Figure 10), we can allow its access to any Provider, since a link is recorded in MMS.

5.3 Comparing the Scenarios

After implementing the architecture in October 2014, we can manage ISI easily through the MMS. In addition, the implementation and deployment of new providers and services take some hours or a few days, whilst in the former architecture, the IT team spent weeks or months. As a result related to Flexibility and Scalability, in April 2015, less than one year after implementation, we had 13 systems in the integration; in April 2018, 20 systems were integrated, and this number is going to grow due to the adherence of new systems at UFLA's system environment.

The security was one of the main points to regard during developing the architecture. Before, when making views available to clients, some actions should be taken: creating a new user, setting the IP address (or range of IP addresses) that could access that view (in MySQL, the relation user:address is 1:1, meaning that, for each new IP address or range of IP addresses to access the view, a new user should be created). Furthermore, when a client, for any reason, stopped using the view without any notification, the permission was still recorded, meaning it was difficult to manage and to have control on which

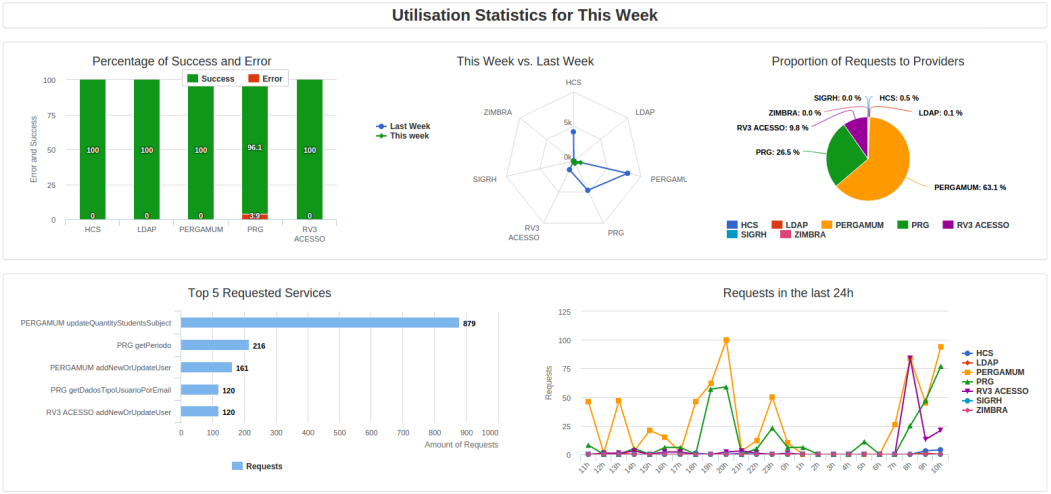

Figure 8: Management and Monitoring System (MMS) Main Window

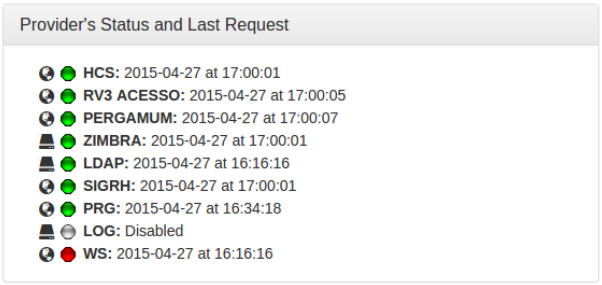

Figure 9: Provider's Status and Last Request

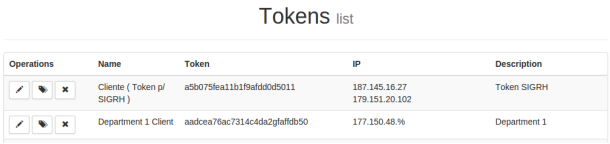

Figure 10: Managing Tokens and authorized IPs

permissions are still in use. Notice that there was no central point of management. It was always dependent on the system and integration tool used.

Nowadays, with the new architecture, we have a central management system, the MMS, in which tokens and IP addresses that can access them can be managed. It offers enough data in order to support decisions, detect unused tokens and unauthorized trials of access. New controls and warnings of numerous access trials will be developed.

The approach, in which a unique global ID for entities shall be created (Suryawan; 2014), is expensive when dealing with several systems, including third-party systems. However, at UFLA we had an interesting characteristic: most of the databases had personal data stored, including CPF (individual registration), which could work as the unique global ID (Suryawan; 2014). For the systems that did not store CPF, such as Zimbra and LDAP, there was a field that, by politics at UFLA, is unique: user login. Therefore, the data consistency is verified using those fields without changing the systems.

Before the new architecture, it lacks a way for monitoring the integration. Most of the failures were detected when a user experienced problems in one of the systems due to the integration stopping. With the new architecture, we can monitor the integration observing in the dashboard: log system, checking services, and graphs of MMS. At the moment, MMS has graphs only for Private Providers and Services. As a future work, we intend to develop a graphical monitor for Public Provider and Services as well.

In general, those charts allow the identification of anomalous behaviours of services in terms of the amount of requests. For instance, we verified that a certain service was requested several times in a short period. Investigating the cause, we found unnecessary calls to that service, and then, we fixed the source-code in the Client with wrong behavior.

6 Conclusion

In this work, we have proposed an architecture for integrating information systems in academic environments, and described how it was implemented in a Brazilian Federal University (UFLA). A team with only 4 IT professionals was involved in the architecture design. However, only 3 professionals from the team are responsible for the database management and information systems integration. Therefore, a small team is responsible for managing the integration, developing consumers and providers, registering tokens and authorized IPs.

Before the implementation of the proposed integration architecture, only 4 software systems and 2 network services were integrated. In the integration, several techniques were used, such as, triggers, UDF, and Federated Engine, but those technologies depend on the DBMS version. With the new architecture, within less than one year, 13 software systems and 2 network services were integrated. This new architecture promoted positive changes because it is monitorable, manageable, secure, and scalable specially comparing to the former integration architecture. Heterogeneous systems, in this new architecture, can be integrated

in an uniform way.

The proposed architecture in Section 4 can be used as a reference by Universities and even by corporate environments, since the architecture is platform-independent and its implementation requires knowledge on web services and access to databases. Therefore, it is a low cost solution that can be developed and maintained by a small IT team and do not require licences of ERP third-party modules or a whole new system to manage the integration.

As future work, we suggest the implementation of a structure to manage and monitor scripts executed by schedulers, since a software system may have to perform batch updates using scripts that connect directly to the DBMS and most of those scripts are not managed and monitored; the development of a generic framework for information systems integration focused on Academic ISI.

References

Alkhanak, E. and Mokhtar, S. (2009). Using Services Oriented Architecture to Improve Efficient Web-Services for Postgraduate Students, *World Academy of Science, Engineering and Technology* **3**(8): 64–67.

Andersson, V., Santos, R., Tillmann, A. and Noguez, J. (2014). COBALTO Webservice: Solução para consistência de informações, *VIII Workshop de Tecnologia da Informação e Comunicação das IFES*, pp. 1–1.

Bessa, J., Branco, F., Costa, A., Martins, J. and Gonçalves, R. (2016). A Multidimensional Information System Architecture Proposal for Management Support in Portuguese Higher Education: The University of Tras-os-Montes and Alto Douro case study, *Proceedings of 11th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, pp. 1–7.

Costa, C., Melo, A., Fernandes, A., Gomes, L. and Guerra, H. (2010). Integração de Sistemas de Informação Universitários via Web Services, *Actas da 5ª Conferencia Ibérica de Sistemas y Tecnologías de Información*, pp. 290–295.

Coulouris, G., Dollimore, J., Kindberg, T. and Blair, G. (2013). *Sistemas Distribuídos: Conceitos e Projeto*, Bookman Editora.

Cunha, M., Souza Junior, M. and Dornelas, J. (2008). O Uso da Arquitetura SOA como Estratégia de Integração de Sistemas de Informação em uma Instituição Pública de Ensino, *Simpósio de Excelência em Gestão e Tecnologia (SEGeT)*, pp. 1–13.

Degan, J. (2005). *Integração de Dados Corporativos: Uma Proposta de Arquitetura Baseada em Serviços de Dados*, Master's thesis, Universidade Estadual de Campinas.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R. and Safina, L. (2017). Microservices: Yesterday, Today, and Tomorrow, *Present and Ulterior Software Engineering*, Springer, pp. 195–216.

Fielding, R. T. and Taylor, R. N. (2002). Principled Design of the Modern Web Architecture, *ACM Transactions on Internet Technology* **2**(2): 115–150.

Fu, Xiao-Long, L. Q.-X. and Yuan, F. (2011). Design and Implementation of University Level Unified Information System Integration Platform, *Computer Engineering and Design* **32**(6): 7–12.

Garcia, C. M. and Abilio, R. (2017). Integração entre Sistemas utilizando Web Services REST e SOAP: Um Relato Prático, *Revista de Sistemas de Informação da FSMA* (19): 34–41.

Garcia, C. M., Abilio, R. and Malheiros, N. (2015). Abordagens e Tecnologias para Integração de Sistemas: Um Estudo de Caso na Universidade Federal de Lavras, *Revista de Sistemas de Informação da FSMA* (15): 11–22.

Gorski, P. L., Iacono, L. L., Nguyen, H. V. and Torkian, D. B. (2014). Service Security Revisited, *2014 IEEE International Conference on Services Computing (SCC)*, IEEE, pp. 464–471.

Hasselbring, W. (2000). Information System Integration, *Communications of the ACM* **43**(6): 32–38.

He, W. and Da Xu, L. (2014). Integration of Distributed Enterprise Applications: A Survey, *IEEE Transactions on Industrial Informatics* **10**(1): 35–42.

Hensle, B., Booth, C., Chappelle, D., McDaniels, J., Wilkins, M. and Bennett, S. (2010). Oracle Reference Architecture - Service-Oriented Integration, Release 3.0, *Technical report*, Oracle.

Hohpe, G. and Woolf, B. (2003). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*, Addison-Wesley Professional.

James, H. (2017). *80% of the web powered by PHP*. Available at https://haydenjames.io/80-percent-web-powered-by-php/ (Accessed 2018 April 03).

Kulkarni, N. and Dwivedi, V. (2008). The Role of Service Granularity in a Successful SOA Realization - A Case Study, *IEEE Congress on Services-Part I*, IEEE, pp. 423–430.

Laurindo, F. J. B., Shimizu, T., Carvalho, M. M. d. and Rabechini Jr, R. (2001). O Papel da Tecnologia da Informação (TI) na Estratégia das Organizações, *Gestão & Produção* **8**(2): 160–179.

Lawrence, K., Kaler, C., Nadalin, A., Monzillo, R. and Hallam-Baker, P. (2006). Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), *OASIS Standard Specification* pp. 1–76. Available at https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (Accessed 15 September 2014).

Martins, V. M. M. (2005). *Integração de Sistemas de Informação: Perspectivas, Normas e Abordagens*, Master's thesis, Universidade do Minho, Guimarães, Portugal.

Microsoft (2003). *What is TLS/SSL?* Available at https://technet.microsoft.com/en-us/library/cc784450%28v=ws.10%29.aspx (Accessed on 27 April 2015).

MYSQL (2005). *Adding a New User-Defined Function*. Available at http://dev.mysql.com/doc/refman/5.1/en/adding-udf.html (Accessed 03 November 2014).

OASIS, O. (2006). *Service-Oriented Architecture*. Available at http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf (Accessed 15 September 2014).

Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach*, McGraw-Hill.

Putro, B. L. and Rosmansyah, Y. (2017). Functionality Design of Enterprise Service Bus (ESB) as Middleware on the Smart Educational Service Computing System Platform, *2017 International Conference on Information Technology Systems and Innovation (ICITSI)*, IEEE, pp. 355–360.

Suryawan, F. (2014). Inter-database Synchronization: A Low-cost Approach to Information System Integration, *International Conference on Engineering Technology and Industrial Application*, pp. 243–436.

Technologies, I. (2001). *Web Services Definition*. Available at http://www.w3.org/2001/03/WSWS-popa/paper13 (Accessed 21 October 2014).

Turban, E., Leidner, D., McLean, E. and Wetherbe, J. (2004). *Tecnologia da Informação para Gestão: Transformando os Negócios na Economia Digital*, 6 edn, Bookman.

Valipour, M. H., AmirZafari, B., Maleki, K. N. and Daneshpour, N. (2009). A Brief Survey of Software Architecture Concepts and Service-Oriented Architecture, *IEEE International Conference on Computer Science and Information Technology*, pp. 34–38.

van den Heuvel, W.-J., Zimmermann, O., Leymann, F., Lago, P., Schieferdecker, I., Zdun, U. and Avgeriou, P. (2009). Software Service Engineering: Tenets and Challenges, *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*, IEEE Computer Society, pp. 26–33.

Zur Muehlen, M., Nickerson, J. V. and Swenson, K. D. (2005). Developing Web Services Choreography Standards: The Case of REST vs. SOAP, *Decision Support Systems* **40**(1): 9–29.