# Implementation of novel faculty e-services for workflow automatization

Dimitar Kitanovski, Aleksandar Stojmenski, Kostadin Mishev,
Ivan Chorbev, Vesna Dimitrova
”Ss. Cyril and Methodius” University in Skopje
Faculty of Computer Science and Engineering
”Rugjer Boshkovikj” 16, 1000 Skopje, Republic of Macedonia

*Abstract*—**This paper presents a brief overview of the concepts for collaboration between various systems developed for the Faculty of Computer Science and Engineering in Skopje. Web technologies such as the HTTP, originally designed for human-to-machine communication, is utilized for machine-to-machine communication, more specifically for transferring machine-readable data in web service formats such as JSON. By using this kind of web technology and communication we can create various software applications suitable for various needs.This kind of web based software applications enable automatization and drastically eased and accelerated the entire procedure whose initial steps in the past was manually.**

**This paper gives a brief overview of two novel systems which are integrated in the faculty software architecture. Software's for master thesis submission and student surveys are integrated as a part of the core systems. The system's network is collaborating using web services, central authentication services and data sharing which is based on cross-platform interfaces.**

*Keywords*: faculty systems; collaboration; systems integration; web services; administration; cross-platform

## I. Introduction

Workflows automatization is a complex process that integrates process automation tools to replace manual and paper-based processes. Workflow Automation refers to the design, execution, and automation of processes based on workflow rules where human tasks, data or files are routed between people or systems based on pre-defined business rules. This paper describes two software applications that are used to improve the work processes of the faculty. In order to develop this kind of software applications that are easy to integrate into existing system architecture and maintenance, novel software design practices are required. For this purpose, different design patterns are used to facilitate the communication between systems and integration to the current system architecture.
Different problems and their possible solutions are presented, regarding systems lifecycle, architecture, process, interface, synchronization and security. Each application endpoint exports OAuth2 security protocol functionalities for system authentication and authorization. Following the principles of the OAuth2 protocol, each server authenticates the users using bearer tokens. Furthermore, the communication protocol adopts the JSON data format as a primary exchanging throughput over a HTTP communication channel. [1]

## II. Background work

Although a lot of work and progress has already been done in the area of web services in the past years, efforts have been mostly focused on service description models and languages, and on automated service discovery and composition[2]. The term Web services is used frequently nowadays, although sometimes it is very ambiguous. Existing definitions of the terms vary from generic to specific and restrictive. One definition is that a Web service is seen as an application accessible to other applications over the Web [3]. This is a very open definition meaning that anything with a URL address is a Web service. It can include a CGI script or refer to a program accessible over the Web with a stable API, published with additional descriptive information on some service directory. A more precise definition is provided by the UDDI consortium, which characterizes Web services as “self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces” [4]. This definition is more detailed, placing the emphasis on the need for being compliant with Internet standards. In addition, it requires the service to be open, which essentially means that it has a published interface that can be invoked across the Internet. In spite of this clarification, the definition is still not precise enough. For instance, it is not clear what it is meant by a modular, self contained business application. A step further in refining the definition of Web services is the one provided by the World Wide Web consortium (W3C), and specifically the group involved in the Web Service Activity: “a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols”. The W3C definition is quite accurate and also hints at how Web services should work. The definition stresses that Web services should be capable of being “defined, described, and discovered,” thereby clarifying the meaning of “accessible” and making more concrete the notion of “Internet-oriented, standards-based interfaces.” [5] [6] It also states that Web services should be “services” similar to those in conventional middleware. Not only they should be “up and running,” but they should be described and advertised so that it is possible to write clients that bind and interact with them. In other words, Web services are components that can be integrated into more complex distributed applications. The W3C also states that XML is part of the solution. Indeed, XML is so popular and widely used today that, just like HTTP and Web servers, it can be considered as being part of Web technology. There is little doubt that XML will be the data format used for many Web-based interactions. Note that even more specific definitions exist. For example, in the online technical dictionary Webopedia, a Web service is defined as "a standardized way of integrating Web-based applications using the XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available" [7]. Specific standards that could be used for performing binding and for interacting with a Web service are mentioned here. These are the leading standards today in Web services. As a matter of fact, many applications that are "made accessible to other applications" do so through SOAP, WSDL, UDDI, and other Web standards. However, these standards do not constitute the essence of Web services technology: the problems underlying Web services are the same regardless of the standards used. This is why, keeping the above observations in mind, we can adopt the W3C definition and proceed toward detailing what Web services really are and what they imply.

Web services were developed as a solution to (or at least as a simplification of) the system integration problem[8]. The main benefit they bring is that of standardization, in terms of data format (JSON), interface definition language (WSDL), transport mechanism (SOAP) and many other interoperability aspects. Standardization reduces heterogeneity and makes it therefore easier to develop business logic that integrates different (Web service-based) applications. Web services also represent the most promising technologies for the realization of service-oriented architectures (SOAs), not only within, but also outside companies' boundaries, as they are designed to enable loosely-coupled, distributed interaction [9]. While standardization makes interoperability easier, it does not remove the need for design patterns that include adapters and mediators. Different Web services may still support different interfaces and protocols. For example, although two map or driving direction services may support JSON or XML and use SOAP over HTTP as transport mechanism, they may still provide operations that have different names, different parameters, and different business logic or protocols. In addition, other opportunities enabled by Web services have an implication in terms of adaptation needs. In fact, having loosely-coupled and B2B interactions imply that services are not designed having interoperability with a particular client in mind (as it was often the case with CORBA-style integration) [10]. They are designed to be open and possibly without knowledge, at development time, about the type and number of clients that will access them, which can be very large. The possible interactions that a Web service can support are specified at design time, using what is called a business protocol or conversation protocol. A business protocol specifies message exchange sequences that are supported by the service, for example expressed in terms of constraints on the order in which service operations should be invoked. Another studied solution is to make system integration with ActiveXML which utilities peer-to-peer interaction between nodes and specifies special data design and ActiveXML web services[11].

## III. Systems architecture

The Faculty of Computer Science and Engineering continues the development of e-platform for student and staff services by providing new e-services and their adaptation with machine interfaces to the central data repository. Such services provide simplification and acceleration of the Faculty administrative workflows by providing easy-to-use interfaces avoiding congestion and bottle-neck scenarios. The system architecture that is discussed in this paper consists of several different subsystems which work as a part of the architecture provided in [12]. That means that the core of the subsystems is a common part which ties the entities as soft links providing scalable and reusable patterns for the purpose of interoperable services.

### *A. Architecture Core*

The core of the service architecture is implemented in Microsoft .NET MVC technology. The authentication process is handled by the Central Authentication Service (CAS) which is implemented in Java. The CAS service involves a back-end service, that does not have its own HTTP interface, but communicates with a web application. The Service manager implements a protocol which is platform independent (JSON based). All applications and services in the system are communicating and synchronizing using this protocol. The service manager is implemented in C Web Api and is used as a mediator for control messages exchange, storing permission access rules, identifying the status of the services (running, failed, blocked...) and enabling intra service communication. As a result of successful authentication, the user obtains JWT token which is passed as authentication header, providing stateless communication between the client browser and the server. The JWT token is used as a key reference for the user credentials. Users identity management is handled by Active Directory. Active Directory is interconnected with the CAS service and serves information about the user credentials. CAS service queries the AD to enable user single sign on authentication for multiple third-party services. User authorization i.e. the role of each user for specific service is handled by each service individually. That means that each services implements its own many-to-many relationship which stores the information about the grant tickets associated to each user in the application. If the user does not contain any grant to the application, he will not be able to access it. The authorization is handled immediately after successful authentication to the CAS service. It is provided authentication by the user group. That means that is the user is a part of the group students, he will obtain different grant that the user from the group professors. This properties can be overridden by specifying the grant for each user individually. The grant with higher weight i.e. with stronger permissions wins the authorization process. The intercommunication among this services is realized with REST JSON-based web services by using the ASP.NET Web API Framework. Such core enables development of software applications that will facilitate the workflow among students, administrative and teaching staff in the faculty with implementation of several use-case scenarios. The current system architecture contained systems for student request service, consultation management service, absence workflow automatization, diploma thesis submission.... Two new systems are modulary added to the architecture, namely systems for master studies submission and student surveys.

## B. *Master thesis submission*

This system is developed in order to facilitate the whole process of master thesis submission, approval and status tracking. Having in mind that the problem is complex, the system itself contains number of workflows in order to cover all the possible scenarios.

In order to implement the process of master thesis submission, we identified the following steps in forementioned workflow:

- The first step of the master thesis submission process is actually a set of several sub-processes conducted by the student and his supervisor. First of all, the student chooses a supervisor and delivers documents which are necessary for the master thesis application. Then the supervisor approves the proposed documents and assigns a committee for the particular master thesis.
- Then the documents are being verified by the faculty student affairs, the secretary and the vice-dean for academic affairs. If some of the documents is missing or is not in the appropriate format, it can be resubmitted by the student, taking him few steps back. In every step, all of the involved roles have access to the documents which are uploaded or changed in the system.
- Next is uploading the draft version of the thesis and its validation by the mentor and the faculty administration. If the student hasn't uploaded a draft version within a year, the attaching master thesis status automatically closes and the whole process is pushed back to the beginning. Otherwize, the validation takes place from the mentor,members of the committee, the secretary of faculty and the member of Teaching Scientific Commission. In this chain of validation, if something is not valid, the student is obliged to make the requested changes in the proposal.
- The last step of the master thesis submission process is submitting the final text of the thesis and determining a date for public defence. In this step, the committee members, can create notes for the completeness of the thesis. After the comments, the secretary approves again and the public defence is being scheduled.

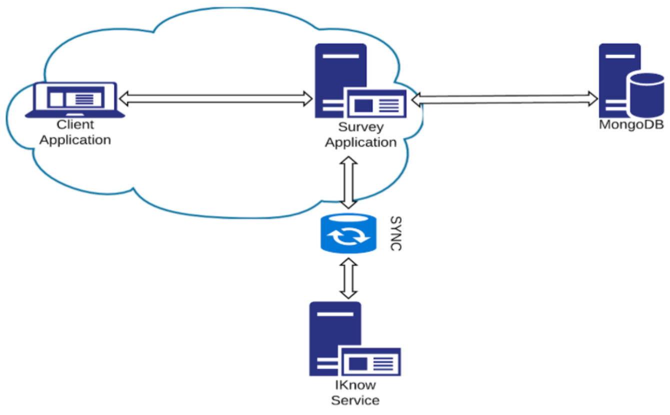
## C. *Student surveys*

The faculty framework used to have this kind of system, but due to legacy frameworks and migrating to IDP authorization and authentication [13], the application was rebuilt using modern state-of-art frameworks such as Angular 6 and ASP.NET Core.[14] [15]

*1) Workflow description:* After completion of whole semester, the Faculty of Computer Science and Engineering opens schedule on the system for surveys, which allows all students to evaluate the professors who is teaching to them.The software requires from students login with their IKnow account,so if the login pass successfully, the user receives his own identifier (token) by the IKnow system. When each student logs in, the survey system performed synchronization with IKnow system which results by taking the appropriate courses. The appropriate courses are courses that the student listened in past semester for which the faculty has created a schedule. Once the whole process is carried out, the student has the opportunity to evaluate the professors for the relevant subjects. When a student chooses a particular subject for which he wants to evaluate the professors, the student gets a form he student gets a form to select the appropriate teaching assistants (if the subject is taught by more teaching assistants), while the professors are taken directly from IKnow system. Once the student has finished the process, he has the opportunity to answer a couple of questions about the teaching process that he follow ie to evaluate his professors and teaching assistants. After this step, the student has the opportunity to save his answers. When the student saves the answers,the database on which is connected this system, created appropriate record. The record contains data about the list of grades that the student filled in form for the appropriate course. In the end, the student is redirected to the home page where the courses of the semester for which he has created a schedule is displayed, and the corresponding subject for which the student answered is removed from that list.

There are two main building blocks which are shown on Fig. 1:

Fig. 1. Application architecture



*2) System architecture:* The implemented system architecture follows the interoperability standards considering the heterogeneity of the external services which are used for data harvesting. IKnow synchronization is made by using RESTful services. Login services are implemented by using Shibboleth protocol. The core architecture of the system is presented in Fig. represents a very simple and flexible coupled solution which is effortless to maintain and expand. The survey application is made by using ASP.NET Core technology which is intended for microservices architecture development. In the implementation of the front-end application, Angular 6 is used. Non-relational database MongoDb, is used as data storage. The reason that we decided to user MongoDB is its performances in write operations and large-scale abilities. The communication between client application and survey application is made by using RESTful services which rises the separability and functionality tier. We use the following synchronization processes:

- Students synchronization with iKnow per semester
- Courses synchronization with iKnow per student
- Teachers synchronization with iKnow per course

All database writes are anonymous. We do not keep track of the users which have filled the surveys in order to keep privacy. In the other side, we keep information about the user that he has completed the answering of the survey. This architecture provides the concept of federalization of survey services for all faculties in the University. The management and support are centralized, placed in FCSE, providing efficient control of software upgrades and improvements.

## IV. Conclusion

Faculty of Computer Science and Engineering continues the development of e-platform for student and staff services by providing new e-services and their adaptation with machine interfaces to the central data repository. Such services provide simplification and acceleration of the Faculty administrative workflows by providing easy-to-use interfaces avoiding congestion and bottle-neck scenarios.In this paper we present novel services implemented in FCSE and UKIM as well. After the success implementation of the system for management of the process for diploma thesis defense, FCSE decides to implement an online solution for workflow automatization of the process for master thesis defense. The implementation of such process is more complex, but facilitates the user interactions. The main goal in implementation is adaptation of the same software and offering as a functional component to the other faculties in UKIM. Also, we develop a software for student survey answering considering the new legislations in teaching staff assessment. The main idea behind implementation of this software is to improve the concept of survey answering among the students and gathering the general opinion about the quality of the curricula and education globally at the faculty. All of the novel services provide scalable architecture. Also, inter-service communication is improved by adding microservice components in implementation of the new one. Central authorization, user management, the concept of single point of responsibility and interoperability improve the quality of e-services and facilitate the future upgrades with novel services.

## References

[1] Beatriz Plaza. Google analytics for measuring website performance.

[2] Fabio Casati Daniela Grigori Hamid R. Motahari Nezhad Benatallah, Boualem and Farouk Toumani. Developing adapters for web services integration.

[3] M.P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions.

[4] UDDI Consortium. Uddi executive white paper, nov. 2001.

[5] Microsoft sql server. https://www.microsoft.com/en-us/server-cloud/products/sql-server/.

[6] Microsoft web api. http://www.asp.net/web-api.

[7] E. Al-Masri and Q.H. Mahmoud. Investigating web services on the world wide web.

[8] H. Kuno V. Machiraju G. Alonso, F. Casati. Web services: Concepts, architectures, and applications.

[9] F. Casati B. Benatallah and F. Toumani. Web services conversation modeling: A cornerstone for ebusiness automation. ieee internet computing, 8(1), 2004.

[10] E. Pimentel J. Troya A. Vallecillo C. Canal, L. Fuentes. L. bordeaux et al. when are two web services compatible?. vldb tes'04. toronto, canada. 2004.

[11] Omar Benjellourn Ioana Manolescu Tova Milo Abitrboul, Serge and Roger Weber. "active xml: Peer-to-peer data and web services integration." inproceedings of the 28th international conference on very large data bases, pp. 1087-1090.

[12] E. Pimentel J. Troya A. Vallecillo C. Canal, L. Fuentes. L. bordeaux et al. when are two web services compatible?. vldb tes'04. toronto, canada. 2004.

[13] I. Dimitrovski V. Dimitrova K. Mishev, A. Stojmenski and I. Chorbev. Cloud services for faculty workflow automatization.

[14] J Lowy. Programming wcf services.

[15] B. Green and S Seshadri. Angularjs.