Refactoring Legacy Enterprise Information Systems to Service Oriented Architecture - The Faculty of Technical Sciences Case Study

Petar Bjeljac, Branko Perišić, Igor Zečević, Danijel Venus
Chair of Informatics
Faculty of Technical Sciences
Novi Sad, Serbia
pbjeljac@uns.ac.rs, perisic@uns.ac.rs, igor.zecevic@uns.ac.rs, danijelvenus@gmail.com

Abstract—**The rise of information system complexity, as well as continuous changes in requirements and technology, turn the legacy enterprise information systems (LEIS) development process into a constant reengineering process. A vast majority of LEIS are based on client-server architecture which, in the era of service oriented approaches, becomes a challenge for reengineering. Due to the underlining black-box architecture of ERP-s, the reengineering complexity shifts to service discovery in contemporary client-server applications. In this paper a generic framework for service discovery based on a formal specification of client-server application architecture is presented. The design constraints and transformation layers of this generic framework are verified on the Faculty of Technical Sciences LEIS.**

Keywords-SOA;enterprise information systems;refactoring

I. INTRODUCTION

Information systems (IS) have undertaken a huge change from their emergence to current days. Rise of IS complexity and rapidly changing IT environment, requirements and technology require reliable, yet flexible solutions in order to support old and current functions of information systems, as well as accommodating future applications. A large number of enterprises already own or use an IS to support their business processes. However, these applications are often outdated legacy systems, with problems regarding maintenance and further system upgrades.

On the other hand, adaptable solutions for specific enterprises are not readily available, and could be too risky or too costly for implementation. This is the reason why enterprises keep running their legacy systems as long as they provide support for basic functionalities.

The Faculty of Technical Sciences is one of the largest educational institutions in Serbia. It possesses all characteristics of large enterprises, and has a functional legacy enterprise information system, developed and maintained by the Faculty of Technical Sciences Software development team. This LEIS supports different business processes, from human resources to student activities. However, as the complexity of this IS rises, and the need for new functions and supported platforms emerges, system maintenance requires a rising amount of effort and resources (developers, time etc.).

This paper brings an insight to the existing LEIS at the Faculty of Technical Sciences (FTS), proposes an adaptable solution and migration process.

In the second section, the FTS LEIS is presented and existing difficulties in development and maintenance are analyzed. The third section describes the service oriented architecture proposed. The fourth section describes different migration approaches. The fifth section describes the initial FTN LEIS migration idea. The sixth section presents related work, and the seventh chapter gives a conclusion and a course for future work.

II. THE LEGACY ENTERPRISE INFORMATION SYSTEM

In [1] legacy information systems are described as programs that have been developed with technologies that have become outdated. In [2] they are defined as “any system that significantly resists modification and evolution”. Sneed [3] states that any user organization which has been using IT for more than five years has a legacy software problem.

Legacy information systems bring multiple problems in terms of software maintenance, requirement changes and expansion. However, several issues still remain in making the decision to conduct migration. Stehle et al. [4] state that the reason why a piece of software persists in an organization for a long time is the fact that this is software that works well, supports an organizations needs and provides support in a dependable manner. Furthermore, due to years of use, this software can be thought of as well tested. According to Khadka et al. [5], legacy systems are still vitally important to enterprises, as they support complex core business processes, and on the other hand, the documentation, skilled man power and resources needed to evolve these legacy systems are scarce.

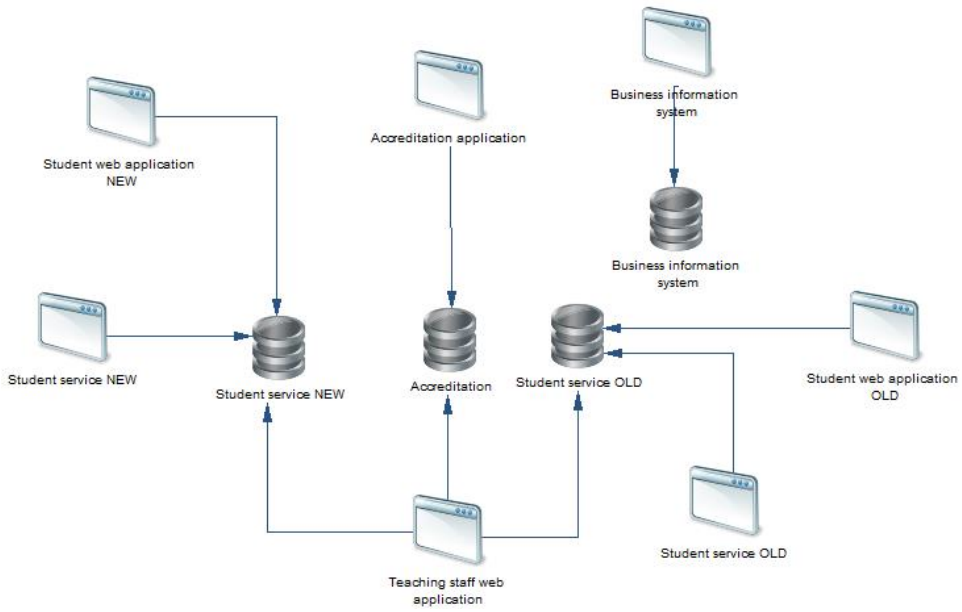

Figure 1. Faculty of Technical Sciences Information System Architecture

A legacy information system, possessing all these characteristics, is in use at the Faculty of Technical Sciences. Figure 1 presents the architecture of this LEIS.

The IS consists of several independent applications, which in general use several same databases. These applications can be divided in three groups:

- Business IS – human resources, financial etc.
- Accreditation IS – Accreditation application and part of the Teaching staff web application
- Student/Teaching IS – Student web applications, student service desktop applications and part of the teaching staff web application

These applications have been developed in different time intervals of the IS life cycle, using different technologies and languages and based on different architecture types. Desktop applications are monolithic (single tier) applications developed in Visual Basic (Business IS) and Java (all other desktop applications), while web applications are client-server applications implemented using Java (student web applications) or PHP (teaching staff web application).

The structural analysis of these systems reveals several issues:

- Different systems use same sets of data and conduct similar functions, implemented in their respectful programming languages independently
- Data migration between systems is based either on manually running database transactions on certain time intervals, or redundant manual data input
- Access control is implemented directly in all applications, making it difficult to adapt access control policies to management requirements.

Recently, the need for keeping pace with new technologies (mobile, tablet applications), as well as the requests for delivering data (statistics on students, salaries etc.) to state institutions bring new issues to keeping the LEIS active in its current form. Due to all issues which have been observed, a migration to service oriented architecture is proposed, as an effort of minimizing future problems in maintenance and adapting to new requirements.

III. SERVICE ORIENTED ARCHITECTURE

According to [6], service oriented architecture (SOA) is an architectural style that supports service orientation. Service orientation is a way of thinking in terms of services and service-based development and the outcome of services. It is an architectural approach that seeks to align business processes with service protocols and the underlying software components and legacy applications that implement them [7]. Papazoglou [8] states that SOA is to structure the information system of an enterprise as a set of services that expose their functional interface and communicate via messages.

Authors in [9] outline that, using this approach, services can be shared and reused across several business processes, resulting in a highly adaptive environment, with lower costs for application development, improved integration and quicker deployments.

Many different definitions of services exist. Brown et al. [10] define a service as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled

(often asynchronous), message-based communication mode. In [11] services are simply defined as a well summarized business utility with an apparent uniqueness and programmatic available interfaces.

In [12] the authors give the main characteristics of services: each service is supposed to be a process with open interface, self-containedness and coarse granularity for business. This means that services have interfaces, through which they are accessed by external entities, hiding platform and implementation specific details. Services should be able to be executed independent from other services.

Finally, a service is a coarse-grained process which can be a business construct by itself, of integrated to achieve coarser-grained services.

All these characteristics are agreed by the authors of this paper as important for the migration of LEIS to SOA, in terms of extracting service candidates.

IV. LEIS TO SOA MIGRATION

Legacy system migration represents a complex task, which subjects the system to a thorough analysis from different aspects. Most authors single out two main aspects in this process: the migration strategy selection and service candidate extraction.

A. Migration Strategy Clasification

A number of authors have discussed different approaches to migrating legacy systems into SOA, and defined different classifications.

Bisbal et al. [13] classify migration approaches into three categories:

- **Redevelopment** – rewriting existing applications
- **Wrapping** – wrapping an existing component into a new, more accessible component
- **Migration** – a combination of the who, transforming the system into a more flexible one, while keeping the original data and functionality

Erradi et al. [14] classify approaches into

- **invasive** – approaches which need the code to be changed
- **non invasive** – adding a wrapping layer that hides internal complexities and gives a service interface, with no changes to existing code

Almonaies et al. [15] propose a classification similar to [13], but add a fourth category – the **replacement**, consisting of completely retiring the application and replacing it either with an on-the-shelf package, or rewriting it from scratch.

After analyzing these approaches, we have come across a two category classification, which can be thought of as a generalization of these categorizations: **redevelopment** and **wrapping**. All other categories can be thought of as special cases or a combination of these two.

B. Choosing Service Candidates

. This process involves the analysis of the system from different aspects, in order to choose the right granularity of services, in order to keep the services independent and self contained, reduce calls which can slow down the performance due to network limitations, but also keep functions wrapped logically, with reference to the existing business processes.

Matos et al. [16] state that it is common to find code fragments concerned with database access, business logic, presentation aspects and exception handling, making it difficult to find service candidates.

The authors in [17] propose a solution, called SMART, for assistance in analyzing legacy capabilities for use as services in an SOA. Smart consists of five activities:

- **Establish stakeholder context** – employing SMIG (Service Migration Interview Guide) to solicit information about stakeholders
- **Describe existing capability** – obtaining descriptive data about the components of the legacy system
- **Describe the SOA state** – gathering evidence about potential services that can be created from legacy components and gathering sufficient detail about the target SOA, appropriate services and their interaction
- **Analyze the gap** – identifying the gap between current state and future state
- **Develop migration strategy** – selecting a suitable migration strategy, described in the previous subsection

The Erradi decision making process [14] explores the organizational factors which influence the strategy selection definition using a simple matrix in three phases:

- **Legacy modernization business context** – identifying key business goals and key issues in the current system in terms of achieving these goals
- **Legacy portfolio assessment** – assessing the current system portfolio in terms of size, interfaces with other systems and technologies used
- **Choosing legacy modernization options** – making the decision based on previously gathered data and a proposed best-match matrix

Authors in [18] build on these approaches composed of:

- **An assessment process** – mainly following the SMART approach
- **A decision making process** – mainly based on Erradi.

What all authors agree on is the necessity of including different types of staff (management, developers, end users etc.) into the decision making process.

The next section describes the proposed FTS LEIS decomposition and migration approach.

V. FTS LEIS MIGRATION

The process of choosing candidates for converting into services is highly complex. The solution proposes a complete reverse engineering process to be executed in order to fully understand the business processes and their extents. However, some aspects of the current LEIS can be of assistance during this task.

The FTS LEIS has, from its early development stages, been implemented with care given to process decoupling throughout the whole system. However, service orientation has not been put to implementation.

The data intensive nature of our LEIS makes it fairly easy to conduct partial automation of application implementation. With this in mind, in the past years, the Software development team at the FTS has developed code generators based on database schemas. However, the code generation is limited to standard forms, which are generic forms for conducting single table operations (CRUDS - Create, Read, Update, Delete, Search). Nevertheless, in the process of database design, tables required in order to support different business processes, have been divided into packages and sub-packages in the Power Designer conceptual database model. The package structure of the Student services system is presented in Figure 2. The packages represent potential metadata to be used in the reverse engineering process, in order to semi-automatically create a business process model as a starting point in the analysis of candidates for service transformation.

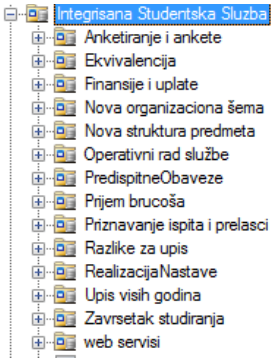

Figure 3. Student Service Package Structure

Another useful aspect of the current LEIS is that all desktop applications used are structured in a manner discussed in [19], where an application possesses a main form, representing the entry point for the application, and containing a menu, where different submenus represent different aspects of application use (subsystems), making them service candidates. This is also data which could be used in the reverse engineering process, and could aid communication with application users, which can be of great assistance during the decomposition process. An example of menu organization is presented in Figure 3. The structure of a menu, representing an exam organization subsystem, with all its sub-processes is presented.

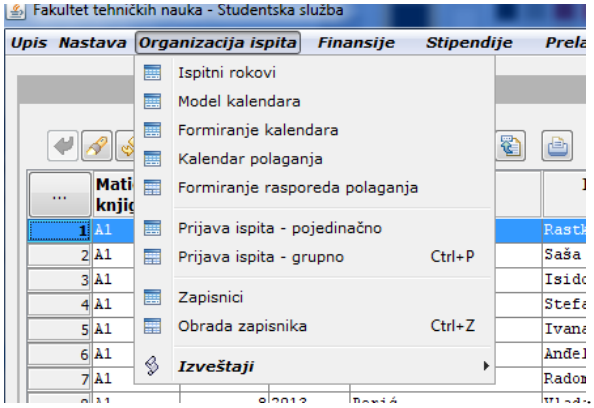

Figure 2. Exam Organisation Submenu

Using these approaches, coarse-granularity services can be singled out and later ranked by several different characteristics (complexity, importance to the system, number of different applications which use them etc.) in order to organize the order of their migration.

Going deeper into the structure of our applications, we find that our low grained services can be further decomposed into three types of fine-grained services:

- **Single table services** – services providing basic (CRUDS) operations on one table
- **Multiple table (transaction) services** – services which change data in multiple tables in a transactional manner
- **Reporting services** – services which are used for gathering result sets from several tables for different types of reports

Another aspect of our proposed SOA system is the security and access control aspect, and its details will be the topic for future work. Two types of access control are proposed:

- **Access control for authenticated application users** – necessary for conducting management based access control policies
- **Access control for authenticated systems** – necessary for authorizing communication between different services

Further analysis also needs to be made in order to choose the best approach for legacy system parts integrations – using wrapping where applicable and conducting redevelopment where necessary.

VI. RELATED WORK

An overview and comparison of migration techniques is given in [1]. In [5] the migration process of one part of a bank information system and lessons learned are presented, with focus given on forming teams, consisted of staff from different company departments, in order to improve the service extraction and migration process. Authors in [20] discuss a model driven approach to the migration of legacy systems. In

[21] the SOA development process based on form types, a concept similar to our standard forms, is presented. A migration of an educational institution LEIS to SOA is presented in [22], with attention given to the use of documentation extraction tools.

Security aspects, which will be the topic of our future research, are discussed in [23] and [24].

VII. CONCLUSION AND FUTURE WORK

Legacy information systems cannot keep pace with changes in operating environment and requirements. Due to difficult maintenance, the costs of keeping these systems rise continuously. Service oriented architecture emerges as an approach to cope with these problems. However, the migration process, if not well conducted, could potentially increase costs, or even lead to complete information service failure.

The Faculty of Technical Sciences legacy information system fails to handle emerging requirements, thus raising the issue of migration. In this paper we have proposed the starting point for legacy system decomposition and migration, and given an overview of several methods and techniques from which to choose. However, these approaches need to be taken into further consideration on multiple organizational layers, including management, system architects, users and current and past developers.

Proposed future work aims at several directions. Firstly, a reverse engineering process needs to be conducted in order to document all existing business processes in the system. The next faze might be the implementation of different abstraction level models to aid the service implementation process. Further, a pilot service needs to be migrated as a proof of concept. Finally, before putting the new migrated system into function, security issues need to be addressed by implementing access control services in order to securely wrap functional services.

REFERENCES

[1] Ali, Seridi, and S. Abdelhak-Djamel. "Evolution approaches towards a Service oriented architecture." Multimedia Computing and Systems (ICMCS), 2012 International Conference on. IEEE, 2012

[2] Brodie, Michael L., and Michael Stonebraker. Migrating legacy systems: gateways, interfaces & the incremental approach. Morgan Kaufmann Publishers Inc., 1995.

[3] Sneed, Harry M. "Integrating legacy software into a service oriented architecture." Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on. IEEE, 2006.

[4] Stehle, Edward, et al. "Migration of Legacy Software to Service Oriented Architecture." Department of Computer Science Drexel University Philadelphia, PA 19104 (2008): 2-5.

[5] Khadka, Ravi, et al. "Migrating a large scale legacy application to SOA: Challenges and lessons learned." Reverse Engineering (WCRE), 2013 20th Working Conference on. IEEE, 2013.

[6] http://www.opengroup.org/soa/source-book/soa/soa.htm#soa_definition, consulted February 28, 2014

[7] www.omg.org/attachments/pdf/OMG-and-the-SOA.pdf, consulted February 28, 2014

[8] Papazoglou, Mike P. "Service-oriented computing: Concepts, characteristics and directions." Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on. IEEE, 2003.

[9] How service-oriented architecture (SOA) impacts your IT infrastructure, IBM Global Technology Services, January 2008

[10] Brown, Alan, Simon Johnston, and Kevin Kelly. "Using service-oriented architecture and component-based development to build web service applications." Rational Software Corporation (2002).

[11] Goyal, Vinay, and Amit Jain. "Role of Web Services in migration of Legacy System to Service-Oriented Architecture."

[12] Nakamur, Masahide, et al. "Identifying Services in Procedural Programs for Migrating Legacy System to Service Oriented Architecture." *International Journal of Information Systems in the Service Sector (IJISSS)* 3.4 (2011): 54-72.

[13] Bisbal, Jesús, et al. "Legacy information systems: Issues and directions."*IEEE software* 16.5 (1999): 103-111.

[14] Erradi, Abdelkarim, Sriram Anand, and Naveen Kulkarni. "Evaluation of strategies for integrating legacy applications as services in a service oriented architecture." *Services Computing, 2006. SCC'06. IEEE International Conference on*. IEEE, 2006.

[15] Almonaies, A., James R. Cordy, and Thomas R. Dean. "Legacy system evolution towards service-oriented architecture." *International Workshop on SOA Migration and Evolution*. 2010.

[16] Matos, Carlos, and Reiko Heckel. "Migrating legacy systems to service-oriented architectures." *Electronic Communications of the EASST* 16 (2009).

[17] Lewis, Grace, et al. *SMART: The service-oriented migration and reuse technique*. No. CMU/SEI-2005-TN-029. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2005.

[18] Aly, Sherif G., and Rafik Amir. "Automated Selection of Legacy Systems SOA Modernization Strategies using Decision Theory."

[19] Milosavljević, Gordana, and Branko Perišić. "A method and a tool for rapid prototyping of large-scale business information systems." *Computer Science and Information Systems* 1.2 (2004): 57-82.

[20] Velmurugan, K., and MA Maluk Mohamed. "A Model driven Approach for Migrating from Legacy Software Systems to Web Services."

[21] Knežević, Marko, Salaheddin Elheshk, Vladimir Ivančević and Ivan Luković. "An Approach to Developing Information Systems with Service Orientation using Form Types" Proceedings of the International Conference on Applied Internet and Information Technologies (ICAIIT), Zrenjanin, Serbia, October 2013

[22] Alkazemi, Basem Y., Abdullah Baz, and Grami M. Grami. "Refactoring Legacy Software Systems into SOA Compatible Style to Support e-Business Development in Enterprise Organizations: The Case of Umm Al-Qura University's Systems."

[23] Chetty, Jacqui, and Marijke Coetzee. "Towards an information security framework for service-oriented architecture." *Information Security for* South *Africa (ISSA), 2010*. IEEE, 2010.

[24] Millham, Richard, Evans Dogbe, and Prenitha Singh. "Role and Data-Based Constraints of Data Access Control in a Legacy System Migration to a Service-Oriented Environment." *International Proceedings of Computer Science & Information Technology* 29 (2012).