On Re-Architecting Legacy Software Systems: The Case of Systems at Umm Al-Qura University

Basem Y. Alkazemi
Department of Computer Science
Umm Al-Qura University, UQU
Makkah, Saudi Arabia
bykazemi@uqu.edu.sa

***Abstract*—This short paper describes our proposed architecture for the software systems at Umm Al-Qura University (UQU). We adopted the notion of SOA to derive the building block of the new architecture. The proposed solution is the first step towards migrating the legacy systems at UQU to new architecture that can respond seamlessly to the emerging e-government requirements.**

Keywords- legacy systems; SOA; e-government.

I. INTRODUCTION

Responding to rapidly changing IT markets - including expanding e-government applications - requires adopting a reliable, versatile and fully flexible system capable of accommodating recent and upcoming changes and modifications efficiently and smoothly while keeping old business needs intact [1-2]. In this day and age, such a system can be described as a mandatory rather than an optional when responding to ever increasing business needs.

Adaptable systems nevertheless are not always available to many large private companies, public organizations, government agencies, hospitals, municipalities, and universities in the Saudi Arabian context. These institutes in reality usually maintain their respective legacy systems as far as the systems provide the basic necessary functionality. However, these organizations are aware of the rapidly changing IT market and are duly planning to replace their old systems at some point in order to accommodate the growing new business requirements should finical resources become available. They may also consider the more cost-effective option of modernizing or re-architecting [3].

Many challenges are attributed to the nature of legacy systems which cannot be easily modified. Systems are usually treated as black boxes not because they lack documentations or because the source code is not available. Instead, the systems are poorly architected in the sense they can no longer cope with new business needs. Hence, become one of the key barriers to adopt any potential e-government business models.

Poorly architected legacy systems can encourage CEOs to replace them with entirely new ones. However, such a decision should be informed and well researched as it still has consequences. Legacy systems usually provide highly customized functionalities that none of the available solutions in the market may provide if purchased as is. For example, setting up new systems may require making huge modifications that can take up to several years to comply with old business needs within organizations while accommodating newer ones.

This research aims at investigating an architectural model to analyze the feasibility of re-architecting legacy systems in order to satisfy e-government business needs. The paper is organized in six sections. Section II presents the background work that set the context of our work. Section III introduces a conceptual system architecture model of SOA. Our proposed model is given in section IV. Section V describes some potential advantages of applying the notion of re-architecting as compared to purchasing new products. Finally, the conclusion and future works is given in section VI.

II. BACKGROUND WORK

We chose Umm Al - Qura University (UQU) [11] as a typical Saudi organization running a legacy system in need of urgent updating to act as our case study for applying and evaluating our proposed alternative model. The goal is to establish a fully integrated environment that supports e-government business. In other words, the institution needs a rigorous solution that promotes changes without interrupting their daily working activities. However, funding seems to be a major constraint that constantly influences the decision for adopting any major new development plans.

Umm Al-Qura University launched its information systems in early 2001 to serve around 3600 employees and just over 40,000 students at the time. It owns old fashioned systems based on Oracle 6i for forms and reports those are built entirely on client-server pattern [7]. The major systems include an in house built *Enterprise Resource Planning* (ERP), *Student Information System* (SIS), *Library Information System* (LIS), and *Healthcare Information System* (HIS). These systems are still used today to serve around 75000 students and more than 7000 employees, a much higher figure than in 2001, with minor improvement to the original functionality.

However, software systems at UQU still lack many capabilities that become core-requirement nowadays in terms of compatibility with different environments (e.g., Mobile devices) and the services provided to students and faculty members in the University. Moreover, with the pioneering of e-government movements in Saudi Arabia, organizations may need to apply major changes to their systems in order to accommodate these new requirements; one of which is process automation that solely requires splitting functional aspects of an application from the business aspects.

 ISBN: 978-1-61208-230-1

Current practices for the modifications to add features to any of the systems are done in an ad-hoc manner by which an application's code is modified to satisfy requirements. Specifically, business processes are implemented directly into the forms confusing the functional aspects of an application with the non-functional ones. As a result, the complexity of UQU systems is building up rapidly in a manner that will become very hard to manage in the near future.

III. CONCEPTUAL SYSTEM ARCHITECTURE

One key driver for establishing our framework is the representation of workflow within a software system. Currently many systems develop their business processes hardcoded into the source code. So, whenever new business processes are required to be implemented the overall code must be modified. Moreover, applications are integrated in a one-to-one manner by writing glue code to establish the integration. This glue code is usually written as a mediator between two applications. Although this approach might look simple to some developers, it causes process design to become totally confused and mixed. In some cases glue code is injected into one of the applications themselves. This is the worst scenario as it will result in very tangled code that cannot be managed over the years especially when developers are dealing with an enterprise solution.

The described framework considers SOA [8] as an integration facilitator mechanism and not as a service delivery mechanism. The framework is composed of different layers that, based on our previous work for analyzing a number of systems [9], any enterprise solution in the market must satisfy in order to ensure flexibility and extensibility of their systems. Figure 1 presents our proposed architecture for an enterprise solution.

Each layer is independent of the other surrounding layers in terms of their main functionality. The description of these layers is as follows:

- Data Access Layer: this layer is responsible for managing the interaction between application and database and hiding the databases used in the organization. So, if different database technologies are used (e.g., MYSQL, Oracle), this layer will manage the connectivity with the corresponding source.
- Application Layer: this layer is responsible for executing the basic functionality that represents an organization's business needs. In the context of an ERP solution, this layer represents the fundamental modules offered by the solution such as HR, Finance, Projects, and Sales. Every one of these modules must be a standalone application that is not aware of any other modules.
- Packaging Layer: this layer is responsible for wrapping the available applications from the application layers into standard component model [12]. All applications are therefore decoupled from their underlying environment and made available through request-response interaction mode.

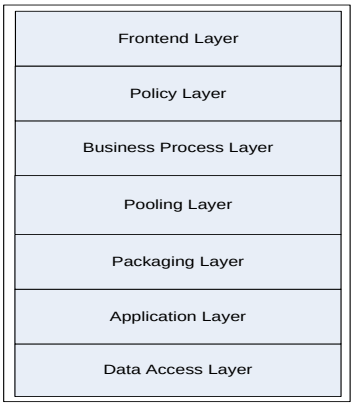

Figure 1. Architectural Layers for Enterprise Solutions

- Pooling Layer: this layer is responsible for hosting the different packaged components and make them ready to be used in a business. In addition, the layer is responsible for establishing the communication pattern and routing protocols that enable service discovery and interaction. It defines the policies that comply with the standards adopted by vendors. For example, web services interact by exchanging SOAP messages over HTTP protocol. Any interaction between services must be accomplished through this layer. This is usually referred to as the Enterprise Service Bus (ESB) layer.
- Business Process Layer: this layer defines the workflows that are employed by an organization. It is responsible for establishing the sequence by which services are going to be invoked to satisfy business requirements. For example, an attendance service might need to issue a request to a finance service to deduct a certain amount from an employee salary.
- Policy Layer: this layer is responsible for defining the privileges for accessing services. A different level of access rights can therefore be granted at this layer according to the defined policy.
- Frontend Layer: this layer is responsible for exposing services for different types of devices and technologies (e.g., web applications, mobile application, cloud computing).

This layered architecture is technology neutral and designed partially utilizing the concept of the SOA pattern. The identified layers are not interchangeable as they must build up in a bottom-up manner. So, for example, a database can be established and tables created for an ERP system. Then, a number of standalone applications are developed on top of these tables to utilize the data in the tables. These applications must then be exposed in a standard manner in order to facilitate their integration with other applications to achieve new business needs. So, the new exposed interfaces are pooled and made ready for requests. Workflows can then be defined on top of the available pool of services in order to integrate different applications seamlessly without affecting

 ISBN: 978-1-61208-230-1

each application's concern. In fact, a workflow defines the design of a system where different components can be executed in a pre-defined sequence. Once all the business requirements are established (i.e., all functionality is implemented), there should be privileges assigned to personnel who are authorized to execute certain processes in the system.

IV. PROPOSED ARCHITECTURE FOR UQU SYSTEMS

UQU is moving steadily and progressively toward providing e-government services which goes in line with the university's technological ambitions. A main objective from the university's website indicates fully automating all its internal processes and establish rigorous infrastructure capable of supporting internal and external exchange of data. In fact, the organization dedicates huge resources and funds in order to achieve this objective.

This requires a comprehensive architecture to be established in order to facilitate harmonious integration of different systems. UQU systems currently operate in three different environments, SharePoint 2010, PHP (codeigniter), and Oracle 6i. Our proposed architecture is meant to integrate all systems regardless of technology in a rather neutral manner. The proposed architecture model is given below.

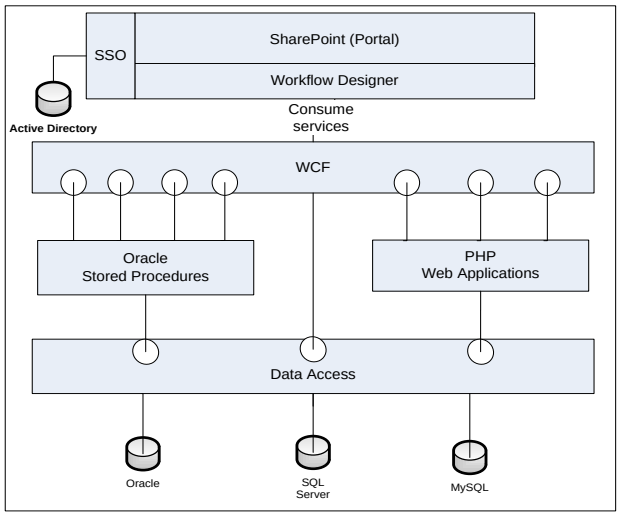

Figure 2. UQU Proposed System Architecture

Figure 2 illustrates the proposed architecture for facilitating the adoption of the emerging business need of UQU based on the resources that are currently available to the university. The main objective of this solution is to promote fully integrated environment that facilitates internal and external data exchange, in addition to promote scalability for future development. UQU currently own full package of SharePoint 2010, in-house built Oracle ERP solution, website and a number of services in PHP, and an Internet Information Server (IIS).

In our proposed solution, SharePoint is utilized to play two main roles; the web presence and the service orchestration layer where business processes are defined through windows workflow foundations (WWF) provided by the SharePoint workflow engine. Services are exposed to SharePoint through the Microsoft-IIS layer where web services are defined. As a result, every application must be wrapped and exposed as a standalone web service that can be consumed by SharePoint. This capability simulates the basic functionality of the well-known Enterprise Service Buss (ESB) pattern for service integration and management which represent the communication layer for integrating the various applications in an organization. SharePoint 2010 must work only on SQL server, hence, in this solution; we propose to use the SQL server for document flow management purposes without interfering with the university database by any means.

This architecture proposes a flexible solution for the ERP system within the UQU and also establishes rigorous platform for any potential ECM functionality required by the university. SharePoint 2010 together with Microsoft-IIS provides the necessary pool and management of services. They facilitate services orchestration in order to enable the interaction between the different services of the system. Any new service can be exposed into this layer and then composed with other services by defining a workflow that corresponds to a predefined business process model.

Ideally, the resulted architecture should promote high degree of extensibility and flexibility where different business processes within or between departments become composable and fully automated. The first step toward that ultimate goal, as far as system structure is concerned, is re-architecting of the legacy system in order to increase the flexibility of IT within UQU. Re-architecting UQU legacy systems with SOA concepts in mind allows for quick response to changing market needs, can implement IT systems that can quickly adapt to changing markets, shifting customer requirements, and new business opportunities.

V. POTENTIAL BENEFITS OF RE-ARCHITECTING

Legacy systems' re-architecting is a cost-effective modernization alternative to the creation of software systems in an organization when a new market or business need arises. Given that purchasing new software has huge cost implications to the organization, it is better for organizations to improve their own proven legacy systems to address their specific emerging business needs. The huge costs of purchasing new software from the market come from the actual price of the software, rollout cost, and training costs. In fact, being new, it might cost higher to maintain in the initial days because it might involve vendor intervention from time to time [4]. More so, this approach promotes low operating costs, with the software built on existing hardware and other systems [5].

The re-architecting approach is low risk modernization option in the sense that existing software is already tested and proven to work in addressing existing business needs. This is more favorable as opposed to software systems an organization purchases and have pilot tests done before establishing if it actually addresses the organization's needs [4]. It allows an organization to transform its existing legacy applications to meet the current market demands without

 ISBN: 978-1-61208-230-1

overhauling the entire system. This, in turn, minimizes the loss of existing IT systems' investments in which the legacy systems hold crucial information and data that is required in the daily operations of the organization.

Another advantage of this approach is that re-architecting encourages the development of a custom software system architecture that is based on the organization's demands and capabilities. This is because this process allows the re-architecting team to survey and understand not only the requirements of the new system, but also the overall capabilities of the organization in managing the new system [6]. Consequently, this enables the development of a system that the organization will use and manage comfortably. During the re-architecting of legacy systems, highlights that re-architecting legacy systems gives the development team an opportunity to transform the current system user interface to the popular web-based user interface if it is not in place already. This helps the users interact with the new system in a friendly manner and, thus, enable the usual operations of the organizations to run with minimum delays.

Software system re-architecting approach permits customization in the training of the system users and maintenance teams. This is realized through re-architecting experts who train the users at each stage of the development process, thus, enabling them to understand the new system with much ease. This is achievable since re-architecting targets certain system enhancements concerning the central part of the solution which aims at handling given business needs.

Improving legacy systems helps sustain an organization's reputation because it principally helps minimize any interruption to routine business operations. This means that customers, too, will feel minimal negative impact, if any. This is critical in businesses where reputation is very important, particularly due to competition. Consider a situation where rolling out a new system takes days to realize. Business operations would have to stop until the system is working as expected but clients may not be that patient and, therefore, consider the organization insensitive to their needs.

Finally, this approach grants an organization the opportunity to employ modern technical architecture such as Service Oriented Architecture and Cloud Computing Architecture. These have tested and proven levels of flexibility to accommodate future technological changes. For instance, when SOA is implemented to support business intelligence (BI), it allows a flawless technology integration to form a consistent BI environment that makes the delivery of data straightforward while simplifying low latency diagnostics at the same time.

VI. CONCLUSION AND FUTURE WORK

This paper accounts for a major obstacle that challenges the decision to adopt e-business solution in any given organization which is the lack of standard architecture of the legacy systems. The paper proposed that re-architecting legacy system can be beneficial to some organizations in improving the architecture of their systems without affecting their underlying business logics. We summarized the main advantages of re-architecting in the following points:

- Re-architecting connects legacy business logic with modern technologies and concepts.
- Re-architecting can evolve legacy applications into SOA-based deployments.
- The new system will require less time spent coding when modifying or developing logic.
- By being based on SOA concepts and built on an advanced framework, the new system will be flexible, transparent, and reliable.
- The new system will be expandable without the danger of a 'spaghetti architecture' emerging.

The next step in this work is to utilize one of the tools available in the market such as the BAZ [10] tool performs the conversion of 6i forms into ADF [13] compatible components. Then, components will be exposed as web services and deployed into the IIS for business process utilizations. Also, we will apply this model to some other universities within the region in order to evaluate its applicability to a wider range of cases.

REFERENCES

[1] C. Holland, and B. Light, "A Critical Success Factors Model for ERP Implementation", Software IEEE, vol. 16, no. 3, 1999, pp. 30 – 36.

[2] K. Bennett; M. Ramage, and M. Munro, M. Decision "Model for Legacy Systems", Software, IEE Proceedings, vol. 146, no. 3, 1999, pp. 153 – 159.

[3] R. C, Seacord, D. Plakosh & G. A. Lewis, Modernnising Legacy Systems: Software Technologies, Engineering Processes, and Business Practices. 2003, Boston: Pearson Education.

[4] D. Reeves, "Legacy systems re-engineering: leveraging your existing assets", Revenue Solutions, Inc. 2009, Available from: http://www.taxadmin.org/fta/meet/09tech/Tech%2009%20papers/Reeves-Legacy.pdf [retrieved: March 2012].

[5] A. Umar and A. Zordan, "Reengineering for service oriented architectures: a strategic decision model for integration versus migration", Journal of Systems and Software, vol. 82 vo. 3, 2008, pp. 56 – 64.

[6] D. Quah, 2005. Thesis on 'Case Study on Re-Architecting of Established Enterprise Software Product: Major Challenges Encountered and SDM Prescriptions from Lessons Learned.' Massachusetts Institute of Technology, pp. 1-122.

[7] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Software Architecture Volume 1: a System of Patterns, 1996.

[8] L. Grace, M. Edwin, S. Dennis, and S. Soumya. SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment, In CMU/SEI-2008-TN-008. Software Engineering Institute, Carnegie Mellon University, 2008.

[9] B. Y. Alkazemi, ,"A Conceptual Framework to Analyze Enterprise Business Solutions from a Software Architecture Perspective ", in the IJCSI, vol. 9, no. 3, 2012.

[10] SmartDeveloper Co., http://www.sd4it.com/Baz.html, [retrieved: Jan 2012]

[11] Umm Al-Qura University, http://www.uqu.edu.sa, [retrieved: Nov 2012].

[12] K. Lau, Z. Wang, "Software Component Models", IEEE Transaction on Software Engineering, vol. 33, no. 10, 2007.

[13] Oracle Co, Oracle Application Development Framework, Available from:http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html [retrieved: June 2012]

 ISBN: 978-1-61208-230-1