On Evaluating the Architecture of ERP Systems

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

Abdullah Baz  
Department of Computer Engineering  
Umm Al-Qura University  
Makkah  
Saudi Arabia  
aobaz01@uqu.edu.sa

Grami M. Grami  
Department of European Languages  
King Abdulaziz University  
Jeddah  
Saudi Arabia  
ggrami@kau.edu.sa

Yasir A. Khayat  
Department of Civil Engineering  
Umm Al-Qura University  
Makkah  
Saudi Arabia  
yakhayat@uqu.edu.sa

ABSTRACT

The architectural aspects of software systems are not typically explained to customers when a product is presented by respective vendors. One immediate downfall customers are likely to encounter emerges when new business needs necessitate modification of the core business processes within their organizations. In some cases, executives are forced to replace the entire systems or re-architect old ones to comply with new architectural standards. This paper describes a proposed architectural framework for an ideal ERP system. It applies the proposed framework to evaluate commonly recognized competitors of ERP solutions in the Saudi market. This study should assist organizations envisage a comprehensive view of vendor’s solutions in order to help CEOs make informed decisions about the optimal solution that fits their current and future business needs.

Keywords: Software Architecture, SOA, ERP, Business Process.

1. INTRODUCTION

Software vendors tend to label their products with glorified terms which are not immediately intelligible to prospective customers. The purpose may be to sell what customers believe they need. In most cases, only functional aspects of the systems are described. However, architectural details that demonstrate how the system is structured are not clearly and explicitly defined, intentionally or in the belief customers may not comprehend them.

It is possible customers lack the proper knowledge of software architecture and its impact on business needs. In fact, it is not common to find an organization that has an existing plan for adopting an extensible architecture for their software systems which dictates the procurement of products in the market. Most purchase decisions are usually taken according to the prevailing conditions at the moment. For instance, customers select vendors that satisfy their business needs within the timeframe and budget available, usually without considering how these systems are going to be built and what the potential consequences of adopting a specific vendor’s technology might be.

Our observation of various organizations across Saudi Arabia concluded that many employ different technologies within their respective systems to satisfy common business needs. For example, some may use Oracle E-Business Suite for their employment management systems and Microsoft SharePoint to run their website. Others may use PHP for websites and SharePoint for the intranet applications in addition to Oracle forms for financial and warehousing applications. Although the diversity of technologies within an organization is usually demanding as far as management is concerned, this variety might be useful in the sense it increases the flexibility and extensibility of the business needs. However, it would not be feasible to conceive this advantage in real life unless the organization relies on a solid architecture that describes different layers where every aspect of functionality may fit.

This paper is designed to draw the attention of Saudi Arabian organizations towards the need for planning IT projects from an architectural perspective. This should be carried out in addition to the business needs which seem to be missing in many IT projects in the region. It argues that understanding architectural specifications is equally important as the functional ones, especially in cases where organizations need to ensure flexibility, extensibility and consistency of their systems.

Furthermore, it describes a proposed architecture for a typical enterprise system and uses the architecture as a framework to evaluate some common enterprise information systems in the Saudi Market. We selected Enterprise Resource Planning (ERP) [4] as a system to against which our framework will be evaluated, mostly from an architectural perspective. ERP is widely used as a software system that manages the different business applications within organizations. Relevant studies in the literature target attributes such as functional capabilities, usability, cost, technology used, and customer satisfaction rather than architectural features. Moreover, this paper establishes the basis for achieving comprehensive alignment between business improvement and software
architecture activities that is always lacking among enterprises [17].

The remainder of this paper is organized as follows. Section 2 presents a background discussion about software architecture to set up the context of the work. Section 3 describes the key quality attributes from a systems’ perspective. Section 4 presents the proposed architectural layer of an enterprise information system. Section 5 discusses the main features of a number of ERP systems in the market. Discussion of the analysis presented in the paper is given in section 6. Finally, the conclusion and practical recommendations are given in section 7.

2. BACKGROUND REVIEW ON SOFTWARE ARCHITECTURE

Rudimentary understanding of the term ‘architecture’ conjures up images of the actual physical construction of a building; the external shape, and the layout of the rooms within that building. In software however, the word ‘architecture’ has other connotations. Both interpretations nonetheless involve creating a product (a software system in the latter case) from a number of selected components rather than a single monolithic example. The way components are incorporated, the order in which they are placed, and the mechanism of interaction between them, are all parts of what a system architecture describes. Bas et al. [7] defines software architecture as the structure of a system that comprises software elements, their external visible characteristics, and the relationship between them. IEEE 1471 [8] similarly defines software architecture as “the fundamental organization of a system embodied in its components, their relationship to each other’s and the environment, and the principles guiding its design and evolution”. Jones [9] also defines architecture as the structure that is composed of components and rules that establish the basis for the interaction between them. All the definitions agree that architecture is concerned with the constituting parts of a system and the relationship between them.

In the literature, many other studies attempted to establish the significance of considering architecture in software systems. One reason in favor of software architecture is to help our understanding of complex software systems. Shaw and Garlan [10] suggest that architecture can be used to define the overall design of a system. Garlan and Perry [11] identify the benefits of considering software architecture in software development as providing support for reusing, evolving, analyzing, and managing software. Budgen [12] believes software architecture to be a tool of describing the constructional aspects of a software system at a high level of abstraction (e.g. design stage). Allen [13] further explains that architecture is the vehicle of communication between the requirement and the implementation stages. Szyzerski et al. [14] also suggest that architecture is important for establishing a context for software systems representing standards and platform requirements.

Garlan et al. [15] identifies a number of architectural characteristics that could result in a mismatch in terms of component interaction within a system. These characteristics are:

- The infrastructure that a component is primarily built on.
- Control issues of whether a component can generate a control signal or not.
- The data type manipulated by a system and the way it is transferred between components.
- The pattern of interaction between components.
- The sequence that components must be instantiated and invoked with.

Furthermore, Yakimovitch et al. [16] interpreting the work of Garlan identifies five variables that describe assumptions of components’ interactions, namely packaging, control, information flow, synchronization, and binding. Their objective was to establish a mapping between architectural assumptions and a number of problem domains that conform to certain standard architectural types. They demonstrated that the defined variables can be used to theoretically classify different software architectures. The previous work emphasizes the importance of considering software architecture as a vehicle to fully understand the different parts of a system. This understanding should help organizations fulfill their business needs. It also helps identify whether or not a functional component can be integrated into their system seamlessly without interrupting the daily work.

As business needs are likely to grow, certain attributes must also be achieved to ensure the readiness of the system. The next section discusses key elements of quality control that establish the context for evaluating a ERP vendor’s solutions.

3. SYSTEM QUALITY ATTRIBUTES

In the context of software engineering best practices, an enterprise software system must satisfy a number of key quality attributes that will ensure its readiness to accommodate new business needs without affecting its
overall software architecture. So, a system adhering to these attributes can be considered a healthy system to accommodate emerging business needs. These attributes include:

- **Reference schema**: tables in the database must be prioritized based on the main business objectives of the organization. For example, the human resources (HR) schema is usually the primary asset in most organizations. So any application must be linked to this schema in order to provide services to the corresponding employee.

- **Applications decoupling**: every application must provide only its basic functionality without mixing its concern with other application business. In addition, applications must not be aware of any other applications in the system. Their main task is to receive requests, process them, and provide results. So, any hardcoded links between applications must be eliminated.

- **Application architecture**: applications must be well structured in the sense that their composing components can be identified and the relationship between them is defined. The architecture of the application can then be utilized to identify the computational components from the data and control exchange components.

- **Separation of concerns**: the functional components of an application must be distinct in the sense that their business logics are not interleaved. For instance, credential check functionality must not be mixed with data retrieval or computation algorithm functionality. Every concern must be separated in a modular way (i.e. component) so it cannot be confused with other functionalities of an application.

- **Standardization of interfaces**: software applications must be wrapped in a way that complies with the standard interface used across the various systems within an organization. The interface usually defines the standard data exchanging model and control topology that is common to all systems.

- **Dynamic binding**: this attribute needs to be satisfied in enterprise systems where software applications can be used differently as per process design. In fact, this feature promotes a wider level of integration between different systems that conform to a standard interface.

- **Integration mechanism**: applications need to expose their standard interfaces in a layer within the overall environment where reaching them can be facilitated. This is usually referred to as a mediator platform where requests can be managed in terms of scheduling, routing, and finding of applications, among other things.

- **Authority matrix**: a system might be accessed by many users, and everyone has their own privileges to execute specific functionality. This is a mandatory attribute that any enterprise system must effectively handle and manage.

- **Data warehouse**: some organizations may have multiple databases for different types of application. This may increase the administration and maintenance overheads. Moreover, this may conflict with the strategy adopted by the organization that needs to integrate their scattered systems. A single unified data source must therefore be employed that wraps all the different databases and exposes a single interface to the applications. This approach is advantageous in the case of having various database Systems (e.g. Oracle, MS-SQL, MySql).

To the best of our knowledge, these attributes are the most significant ones that organizations must consider when defining their system architecture. The identified attributes are the main driver for establishing our proposed architectural framework, which is given in the next section.

### 4. PROPOSED ENTERPRISE SYSTEM ARCHITECTURE

A key driver for developing the framework is the representation of workflow within a software system. Currently many systems develop their business processes hardcoded into the source code. In other words, whenever new business processes are to be implemented, the code must be modified accordingly. 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 like a quick fix, it actually makes the design extremely complicated and confusing. In some cases, glue code is injected into one of the applications themselves. This is the worst scenario as it results in a tangled code that gets worse over time.

Our proposed framework considers SOA [5] as an integration facilitator mechanism and not as a service delivery vehicle. The framework is composed of different layers that, we believe, 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 from the surrounding ones in terms of 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. SQL, Oracle), this layer will manage the connectivity with the corresponding source.

- **Business 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.

- **Exposure Layer**: this layer is responsible for exposing the available applications from the application layers into services (e.g. web services, com+ components, EJB). All applications are therefore decoupled from their underlying environment and made available through request-response interaction mode.

- **Communication Layer**: the communication layer is responsible for establishing the necessary patterns 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.

- **Orchestration Layer**: this layer defines the business processes 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. This process is defined at the orchestration layer to feed into the payroll system.

- **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. So, users can work on a pre-defined set of services assigned to them as per their roles in the overall organizational structure.

The layers are not interchangeable and should follow a bottom-up design. 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 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. The new exposed interfaces are pooled for requests. Workflows can then be defined on top of the available pool of services in order to integrate different applications seamlessly without affecting 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 authorized to execute certain processes in the system.

5. ERP SYSTEMS ANALYSIS

A number of well-known ERP systems are currently available in the market. Oracle, for instance, is among the prominent vendors advertising their Oracle Apps, or the E-Business suite (EBS) [1]. Oracle ERP is a three-tier system that is composed of four basic modules, namely Human Resources, Project Management, Finance, and Asset Management. These modules are built on top of a unified Oracle database. The interaction between these modules is achieved via the Business Event System (BES) that triggers message creation or consumption of any registered parties. Oracle currently offers an additional package,
namely the SOA suite, which can be integrated with the E-Business suite in order to promote enhanced scalability. ERP applications can therefore be exposed on the Oracle Service Bus (OSB) as services. These services then interact with each other through a business process design defined in BPEL. Recently, the key features of the SOA suite became an integral part of the Oracle E-business suite R12.1 package with the inclusion of the Oracle EBS adapter which exposes PL/SQL as services. As the privilege of using these added features requires purchasing separate licenses, the consequences, especially in governmental organizations, can be very expensive.

Microsoft offers a selection of ERP solutions to suit various customer needs, one is known as a comprehensive solution is Dynamics AX [2]. It employs the three-tier architectural pattern, namely, client tier, Application Object Server (AOS) tier, and database tier. The client contains forms and reports code. AOS is used to execute application objects such as classes and queries. The database is normally used to store data for the ERP. Microsoft Dynamics AX utilizes the Application Integration Framework (AIF) to facilitate the integration of application-to-application and also business-to-business. AIF supports the creation of generic web services and also document services; it also facilitates the consumption of external web services from within Dynamic AX. Another ERP solution provided by Microsoft is the Dynamics GP, which is also based on a three-tier architectural pattern. The application tier is composed of three main components: the Dexterity tool and runtime, Dynamics Application Dictionary, and SQL server. The Dexterity tool is used to build the forms and also to attach scripting code using sanScript to applications. The Dexterity runtime environment is used to enable the execution of a functioning application to end-users. This tool is therefore responsible for the development and the execution of the application interfaces. The Dynamics Application Dictionary (DAD) is responsible for storing the business logics in common component architecture such as COM+ and DCOM [6], so other distributed applications can use them as service providers. The main design consideration of this dictionary is to separate the presentation logic from the actual business logic of an application, so services can be accessed independently of any form or application of the presentation layer. The workflow engine is not part of the overall structure but Dynamics utilizes SharePoint to provide this feature.

SAP ERP [3], known as SAP R/3, is another prominent solution in the market. It is primarily based on a three-tier architectural style: the presentation layer, the application layer, and the database layer. The presentation layer represents a tiny application, namely sapgui.exe that is usually installed on the client's machine. The application servers, namely SAP Netweaver, host different SAP services that execute code written in APAB/4 language. A messaging server is responsible for routing requests between applications and establishing a means of interaction between them. The main modules exhibited by SAP ERP are: Financials and Controlling (FICO), Human Resources (HR), Materials Management (MM), Sales & Distribution (SD), and Production Planning (PP). A comparison matrix of the three solutions is given in Table 1 below.

### 6. DISCUSSION

ERP systems provide similar functionalities and share a common three-tier architectural pattern. The three-tier architectural pattern can satisfy, to some extent, the scalability requirement we described earlier; however, it is not very efficient in terms of integrating services or applications. Currently, the business logics are implemented in the application tier in all the ERP systems. In Oracle ERP, some business logics are stored in a database as well. In other words, when there is a need to simultaneously integrate two or more applications / services, one can either modify part of the application's code or write an additional mediator application that establishes the link between the corresponding parties. The proposed solution of integrating workflow business in the context can tackle this problem and solidify the application layer. Moreover, it can satisfy the scalability and integration requirements.

The Oracle ERP does not adopt the principle of data source where different types of databases are used because it is restricted to its own technology platform. This is not the case in Microsoft Dynamics and SAP R/3 as the database link layer is developed to manage interconnectivity with any type of database servers. None of the ERP products support the idea of services where they decouple business logics from the underlying environment. Currently every application must be written...
in a specific programming language that sticks to certain architectural specification. This adds extra overheads when there is a future need for potential development. All the three ERP systems lack a well-defined integration mechanism. The same can be said for the communication layer responsible for managing interactions and finding services. Microsoft Dynamics has a workflow engine that defines how documents must be transferred within the organization. However, the workflow engine is not designed to facilitate the orchestration and integration of applications or services.

It is obvious that all the ERP solutions focus mainly on the functional part to satisfy business needs. However, an architectural arrangement to support scalability and flexibility is not considered in the original building block of the system. These additional capabilities can cost a considerable sum of money even though they play a significant role in enhancing the scalability and flexibility of software systems within organizations.

7. CONCLUSION

We attempted to present our proposed framework of evaluating ERPs as one common example of enterprise information systems in the market. The framework is based on the concept of SOA to help define different architectural layers that we believe any enterprise solution must fulfill. A survey of common ERP systems in the Saudi Market has been conducted to evaluate their solutions. Although the survey was limited to three ERP systems, these solutions are nonetheless the most commonly used. The paper laid the foundations on which organizations can invest when building blocks of vendors’ solution and not only rely on the functional aspects of their system. One important advantage of this approach is defining a reference model which incorporates different technologies in the market and tailors them to meet the business needs of an organization. Effectively, the organizations should no longer be controlled by vendors due to less than well-informed decisions made when initially purchasing systems. The next step in this research is to implement the proposed architectural model on representative organizations in Saudi Arabia and find out how the theory measures up when it is put into real life situations.

8. REFERENCES