A COMPONENT-BASED APPROACH TO ERP DESIGN IN A DISTRIBUTED OBJECT ENVIRONMENT

Bonn-Oh Kim, Seattle University, bkim@seattleu.edu, Diane Lockwood, Seattle University, dianel@seattleu.edu, and Ted Lee, Memphis State University, elee@memphis.edu

ABSTRACT

ERP (Enterprise Resource Planning) vendors have seen a dramatic increase in their sales this decade. Even though several vendors are producing great products and making huge profits, there are some problems to be resolved to make ERP applications a continuous success in the next decades. Current ERP applications have the low reusability and inter-changeability of various modules among different vendors' packages. One of the main reasons for these shortfalls is a tight coupling of ERP domain knowledge with the particular implementation tools. Also, efforts in establishing and using the standards in specifications of ERP applications have been inconsequential. In this article, strategic steps to wield a dominant power in the future ERP market are discussed. These steps are 1. Knowledge Modeling: Abstraction of Domain Knowledge from Tools; 2. Componentization of Domain Knowledge; 3. Implementation of Componentized Domain Knowledge; 4. Marketing Strategies for Domain Knowledge Components.

Keywords: ERP, Component-Based System, Object-Oriented System, Distributed Objects

INTRODUCTION

Since the early 1990s, a notion of business reengineering has been very popular in many companies, especially in the USA. One of the contributions of business reengineering is that corporate information systems should be viewed as an enabler to transform the business processes and consequently organizational structures. To fulfill the mission of an enabler of business transformation, corporate executives found that corporate information systems should be planned, designed and implemented from an enterprise-wide perspective. A collection of

islands of software located in various divisions of an organization could not satisfy the new needs of large corporations.

ERP (Enterprise Resource Planning) vendors promise to deliver an integrated set of software systems for various functions of a company, including accounting, manufacturing, logistics and others. Recently, ERP vendors such as SAP, Baan, PeopleSoft, Oracle and J. D. Edwards have seen their sales growing exponentially. Analysts also state that 70 percent of Fortune 1,000 firms currently have or will soon install ERP systems and the ERP market is predicted to grow from $15 billion to $50 billion over the next five years (1). Behind the successful stories of ERP, however, there are several issues to be dealt with in order to adapt to the ever-changing computing environment and to maintain the competitive advantages.

Borrowing the idea from the industrial manufacturing, software components built based on standard specifications can be a building block for resolving the current problems in designing ERP applications. This component-based software development can benefit buyers too. From a buyers' viewpoint, the componentization of business application system makes it easier for them to create a "best of both worlds" scenario: the ability to mix and match applications from their primary ERP providers along with applications from other providers that address site- or process-specific requirements (2).

In this article, problems of current ERP applications are discussed and core competencies of ERP vendors are reviewed from a perspective of overall computing architectures. Then, as a precursor for building the components for ERP applications, modeling of domain knowledge abstracted from the implementation tools is discussed. In addition, a four-step approach to component-based ERP design based on the modeling of domain knowledge is proposed to wield a dominant power in the future ERP market.

CURRENT DESIGN OF ERP SYSTEMS

Currently, each ERP vendor has been developing its own proprietary systems in various domain areas. Since ERP customers prefer the seamless systems across their business functions, ERP

vendors are continuously expanding into new domain areas. However, one vendor does not necessarily produce superior ERP packages across all business functions. Each vendor maintains superiority in some functional domains, e.g., PeopleSoft for human resource management.

From a perspective of ERP customers, they have to opt for using all ERP applications primarily from one vendor or selecting many packages from different vendors. If customers can choose the best from different ERP vendors without worrying about the compatibility among different vendors' ERP packages, they can maximize the productivity gains by installing the best ERP applications in their organization. From an ERP vendor's perspective, it is very difficult to specialize in any particular domain functions (e.g., manufacturing, financials, etc.) because many customers want a smorgasbord of ERP packages from one vendor. If ERP packages from different vendors are interchangeable or compatible, some problems aforementioned can be somewhat resolved.

There have been overlaps in efforts developing virtually the same type of applications (e.g., accounting packages) by many different vendors. Reinventing a wheel is a last thing we need to do. Current ERP designs in industry lack reusability and interchangeability of domain application components. To develop a successful and dominant company in the ERP market, a strategic move to a component-based ERP design and marketing will be required. ERP vendors should be in a business of specifying the ERP components as well as building them. Once the design of specifications for ERP components is produced, manufacturing of each component can be outsourced to third-party developers.

CORE COMPETENCIES OF ERP VENDORS

Currently, ERP vendors' core competency appears to reside in its conceptualizations of application domain knowledge in financials, manufacturing, distribution and others rather than the application development tools (e.g., OneWorld from J. D. Edwards). Even though they are making profits by selling the ERP software on different machines, there will be even more profitable and huge markets for specifying and producing various components of each application. Readers are reminded that automobile companies make huge profits by specifying

and selling the automobile parts (or components). For example, GM controls an automobile business by specifying how the third-party manufacturers produce the parts for GM cars and trucks. Controlling the standards for specifications of parts endows an intrinsic dominant power to GM. GM does not produce all the parts. GM basically controls the specifications of parts.

By breaking down a huge complex application package into many independently packaged components, we can sell each component to all types of customers. Customers of ERP products do not have to be a mid-sized company wishing to have the financial applications installed. We can expand the market to software developers and end-users as well as our traditional mid- to large-sized companies. For example, if we package the accounts receivable application as a separate independent product using DCOM (Distributed Component Object Model) standard in Microsoft Windows 98/NT environment, potential profits could be immense.

COMPONENT-BASED ERP DESIGN

What is important is that we need to rethink how we develop the applications. Software design should be more or less like designing and manufacturing automobiles. GM and Ford make money by specifying components and assembling cars as well as by manufacturing parts. ERP vendors should be prepared to design and sell components of applications as well as the final whole ERP solution. Packaging of each component needs to be done using the industry standards. Once we conceptualize and build each component, we can package it using Microsoft DCOM, OMG's (Object Management Group) CORBA (Common Object Request Broker Architecture), SUN's JavaBeans or whatever. ERP vendors need not to be in a business of setting the standards for packaging. Instead, Their strength should be in conceptualization and specification of components and packaging them in various forms.

Currently, most of conceptualizations of knowledge in application domains are already available in the forms of computer code and some high level designs. Unfortunately, however, they are frequently hidden and dormant. They are tightly coupled with the tools (e.g., OneWorld). What we need to do is to abstract the knowledge from the tools and to specify each knowledge component independently of any tools. Then, each knowledge component can be manufactured

using whatever tools in a massively distributed environment. Thereby, we can give a new profitable life to this latent asset of ERP vendors. We need to recreate and repackage the knowledge. For effective packaging and distribution, it is very important to adopt a distributed object-oriented approach in software development and to normalize the database systems.

More specifically, the following steps need to be done:

Knowledge Modeling: Abstraction of Domain Knowledge from Tools

When designing software systems, we need to think about what is constantly changing and what is not. Invariant parts of the system should be separated from the variant parts in order to make a whole system adaptable to a new environment. In the ERP market, the domain knowledge in accounting or manufacturing does not change much over time while implementation tools are almost constantly changing. When there are new implementation tools available, the domain knowledge should be easily ported to a new tool environment.

Knowledge models in various domains should be independent of any lower-level abstractions, including implementation tools, middle-ware and others. Knowledge modeling has been a research topic in artificial intelligence for a long time and there are many models available now. For the ERP knowledge, however, two modeling tools can be most effective: i.e., object-oriented modeling for business activities and entity-relationship modeling for persistent data. These object models and entity-relationship models should be independent of any particular tools or technical environment. Depending on a market situation, we should be able to implement the knowledge models in almost any programming language and hardware environment.

Componentization of Domain Knowledge

One of the most important characteristics of components is the separation of "what" from "how". Each component should have a clearly defined interface specifying "what" it does while hiding "how" it does. Using this interface, each component can communicate with other components. A collection of objects will constitute a various grain size of knowledge in each domain

application as patterns or frameworks of objects. IBM's San Francisco project can provide a good reference model. For more details, visit the following Web site: http://www.ibm.com/Java/Sanfrancisco/.

Implementation of Componentized Domain Knowledge

As we have seen over many years, technical environments are changing at a very fast speed. ERP vendors should not be in the component packaging business. Currently, there are several packaging standards available, including Microsoft's DCOM (Distributed Component Object Model), OMG's (Object Management Group) CORBA (Common Object Request Broker Architecture) or SUN's JavaBeans. Knowledge components specified should be packaged using whatever standards popular. If we design and specify the domain knowledge independently of any particular packaging standard, we should be able to repackage the domain knowledge components as dictated by the market.

Marketing Strategies for Domain Knowledge Components

To become and stay a dominant power in the ERP market, an ERP vendor needs to control and own the standards for ERP components and allow others to manufacture the approved components. Open Applications Group's work can be a good place to see what is going on in the area of open standards. For more details, readers are referred to the following Web site: http://www.openapplications.org/. Microsoft practically controls the microcomputer market by owning a standard in the operating systems while many other companies build software based on Microsoft's standard. Each ERP vendor does not have to manufacture all the components.

Once an ERP vendor possesses the standards, it should be an owner of components catalogue. The catalogue of ERP components should enable a market place where software builders can shop to build or customize their own applications. An ERP vendor should be more than just a component builder. To be successful, ERP vendors should create and control the market for business software components. Readers are reminded that NYSE (New York Stock Exchange) has become a very profitable venture by creating a market for stock exchanges and by

controlling how stocks should be exchanged. Vendors controlling the software components marketplace should be able to set the rules for the ERP market.

CONCLUDING REMARKS

As in the industrial sectors of the USA economy, there will be more profits in specifying and designing software packages rather than just manufacturing them even in the software business in the near future. These days, manufacturing of software can be achieved less costly by exporting it to countries like India. In the ERP market, we need to think more like Nike. Designing Nike shoes is a lot more profitable than just manufacturing them. Activities involved in designing the specifications for the ERP components are quite distinct from manufacturing them. Once an ERP vendor controls the specifications of knowledge component for the ERP domains, it can be in a strategic position to dominate the ERP market with an absolute competitive advantage in the 21st century.

REFERENCES

1. Bingi, P., Sharma, M.K., and Godla, J.K. (1999). Critical Issues Affecting an ERP Implementation, Information Systems Management, Vol.16, No. 3, Summer. 7-14.

2. Caruso, David. (1998). Get a Backbone, Intelligent Enterprise, Vol. 1, No. 3 December, pp. 18-20.