Coordination of a Service Oriented Architecture

Jason Nichols
Arizona State University, USA

Andrew Chen
Arizona State University, USA

INTRODUCTION

As e-commerce models and applications have been widely employed in today's business environment, a new movement to so-called *dynamic e-business* has been urged to advance e-commerce applications to the next level: simplifying business interaction over the Web through effective and widely accepted messaging and data encapsulation standards (Chen, Chen, & Shao, 2003). Gisolfi (2001) defined *dynamic e-business* as the next generation of e-business focusing on the integration and infrastructure complexities by leveraging the benefits of Internet standards and common infrastructure to produce optimal efficiencies for intra- and inter-enterprise computing.

Infrastructure for both inter- and intra-organizational computing has undergone a significant maturation process from centralized mainframe computing to early distributed client/server environments, and most recently taking on a service orientation (Roure, 2003). Service-oriented architecture (SOA) represents the framework for the latest generation of service-based computing where once proprietary and monolithic applications are broken down into components and exposed through open standards for use by both internal and external enterprise partners. The SOA paradigm is argued to include in its list of benefits a higher return on investment, increased software reuse, and the capability to support dynamic service assembly (Stevens, 2005).

An increased return on investment is achieved through the componentization of application capabilities. The argument goes that the usefulness of a component (defined here as bounded by its functional capabilities to one distinct business domain) outlives the usefulness of an application (since applications are developed to support a subset of processes in a domain while a component is not constrained, by definition, to any particular process set).

Within the SOA paradigm, the development of applications to support a set of business processes is replaced with the connecting of components from distinct business domains in order to address the computational needs of a particular process. It is clear, then, that SOA has a positive impact on software reuse as components are leveraged across many configurations to address the specific computational needs of many different processes. To this end, one can map the reusability of components in an SOA context to the third argued benefit—dynamic service assembly.

Dynamic service assembly means that components are not developed with the complete set of application scenarios in mind. Instead, components are created to exemplify the information and computational contribution of a specific business domain. The choice of how these components are used later on is therefore not limited to assumptions of usage made at the development stage. Indeed, it is possible that the most valuable use for any given component may not exist at the time of component development. As business processes evolve dynamically over time and business needs for information and computational support change, a service orientation leveraging components that are developed in the absence of constraints for how they might be utilized allows for dynamic reconfiguration of services in order to adapt to changes in the business processes themselves. This ability to reconfigure increases reuse and extends the lifetime (from a value perspective) of the components that are developed. This, in turn, feeds back to an increased return on the investment in software development which is typically the primary motivation for buy-in to the SOA paradigm.

Similar to the shift from a mainframe to a client/server architecture (Malone & Smith, 1988), however, the shift to a service-oriented architecture requires consideration of costs associated with coordinating activities in this new environment. Management of these coordination costs will be necessary in order to preserve the purported increases in return on investment. Put simply, if the return on investments in software development increases but the costs associated with leveraging the developed information technology artifacts for business value also increases, then it is possible that the value created will be diminished or even overrun by the operational expense of coordinating use. In order to ensure that this is not the case, this article leverages a coordination theory approach to first understand the impact that a shift to service-oriented architecture will have on the cost of

coordinating activity both within and across the firm, and second to make recommendations for how these coordination costs can be addressed to preserve the return on investment from a shift to service-oriented architecture.

BACKGROUND

Client/Server and Associated Shifts in Cost

In order to understand the impact of a fundamental shift in information technology architecture, one can observe an historical shift of this type in that of mainframe computing to client/server models. The client/server computing paradigm came about as a response to the need for computational flexibility to support changing environments (Kavan et al., 1999). In this sense, the historical goals motivating a shift to client server map well to the contemporary goals that motivate a shift to SOA. In the client/server context, it was noted early on that the purported cost savings in terms of infrastructure were frequently overrun by management, support, and training costs (Diamond, 1995). History has shown, however, that the increased management costs were characteristic of early client/server adoptions, while best practices for management of the new architectural paradigm were being discovered in real time (Borthick & Roth, 1994).

Critical to the understanding of how a shift such as this impacts costs and organizational effectiveness, coordination theory has much to say regarding the movement from mainframe to client/server architectures (Malone, 1987; Malone & Smith, 1988; Malone & Crowston, 1994; Shin, 1997). In essence, mainframe computing flourished in the era where computational power was expensive and should therefore be conserved and used efficiently. By centralizing computational power, demand for computation was aggregated and benefited from a smoothing effect due to the pooled variance of computational requests. This allowed firms to run their computational resources efficiently by making capacity decisions that minimized unused processing power, since the demand for processing held relatively stable.

The trade-off to a centralization of computational power preserving on computing costs was an increase in the cost of coordinating computation and communicating across the organization. Examples of these costs include the cost of transmitting data and processing requests to a distant centralized processor, latency in the time between request submission and receipt of computational results, overhead related to scheduling and planning the execution of processing requests, and the existence of centralized computer centers to oversee scheduling, governance, and maintenance of a large and centralized computing platform. The emergence of the personal computer signaled a dramatic decrease in the per-unit cost of processing. Suddenly, the importance of efficiently managing processing capacity in order to minimize processing costs became relatively insignificant. Organizations found themselves in a new environment where processing power was cheap and could be spread across the firm to support the computational needs of individuals with reduced latency for the execution of a processing need.

But early adopters, as mentioned above, suffered from cost *increases* with the switch to a client/server model. Coordination theory suggests that the process of driving out such cost increases began with the realization that focus must shift from processing costs to coordination costs. Management practices and organizational design and governance must adapt along with the architecture in order to balance the trade-offs between processing and coordination costs. As processing became less expensive, focus turned towards the coordination of processing, and firms came to realize the value potential for a client/server model by adjusting their management and governance structures and policies to economize on coordination costs.

The Emergence of SOA

Service-oriented architecture represents the next major architectural shift for organizations currently existing in a client/server model of computation. In order to understand the cost, management, and governance implications of such a shift, it is first necessary to understand the current state of SOA as an emerging technology. Papazoglou and Georgakopoulos (2003) identify a set of service layers which can be considered a foundation for the SOA stack, as depicted in Figure 1.

Web services have become the foundational technology for the service-oriented architecture paradigm

Figure 1. SOA technology stack