



SERVICE ORIENTED ARCHITECTURE AND WHAT IT MEANS TO CAPACITY MANAGEMENT

CONTENTS

Introduction	2
SOA Basics	3
SOA Basic Structure	3
What SOA Means to Capacity Management	4
What Changes.....	4
What Stays the Same	4
Capacity Management Challenges	4
Conclusion	5

INTRODUCTION

Service Oriented Architecture, or SOA for short, is probably the most discussed application development topic today. Where and how it originated is a matter of opinion. Some say it was initially devised by two Gartner Group analysts. Some say Sun Microsystems developed it to support their JINI project. Some say it is an offshoot of Component Based Design or CBD and others say it is a natural evolution from Structured Programming and Object Oriented Design.

Besides the variety of claims, there are a multitude of “SOA” solutions being offered, usually in a form that touts a vendor’s own proprietary hardware and software solutions; actions somewhat in conflict with SOA’s goal of transparent interoperability.

Standards organizations generally lag behind technologies and architectures, usually waiting until general acceptance is clear before committing precious resources to set a standard. SOA is no exception. Recently OASIS (Organization for the Advancement of Structured Information Standards) issued a standard called [“OASIS Reference Model for Service Oriented Architecture,”](#) however it provides basic definitions that are more suited to guiding vendors in developing solutions than IT organizations implementing SOA. OASIS continues its work to better define the different facets of SOA but we feel it will be some time before work can be completed and agreed upon.

Because of the wide variety of offerings and minimal standards, SOA means a lot of different things to a lot of different people. In this paper, we will attempt to provide a simplistic view of SOA based upon its goal of a reusable, interoperable, platform independent architecture, then discuss how this new architecture impacts capacity management organizations.

SOA BASICS

It is important to remember that SOA, as the name implies, is only an architecture. It is not a set of processes, procedures or best practices; however those are needed if you are to effectively and efficiently operate applications within the architectural framework.

SOA provides a structure where applications or application components (i.e. services) more closely resemble business processes, where services are easily reused and interoperate well with each other. The architecture provides a framework by which services interface with one another and interoperate independent of how they are implemented and independent of whether they are internally or externally sourced.

When done correctly, additions and changes to the application portfolio can be accomplished quickly and easily, permitting the business to capitalize on opportunities while keeping IT infrastructure costs low, improving competitive advantages.

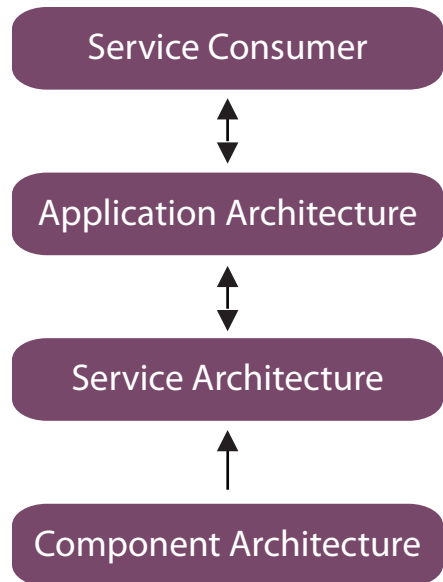
Common IT management reasons for implementing SOA:

- **Develop architecture that can support and sustain shifting business requirements**
- **Use technology-neutral components and services**
- **Speed time to market of new applications and functionality enhancements**
- **Employ component reuse techniques to reduce development times**
- **Leverage reusable components to reduce application inventories, eliminate duplication of effort thus lowering ongoing support costs**
- **Increase transparency between internally and externally sourced components and services**
- **Simplify interoperability across the enterprise with a consistent approach to application design**

- **Use IT infrastructure assets more efficiently and effectively**

The architecture is comprised of components, services, and processes. Although the solutions differ depending upon vendor products, there appears to be a common theme thus we have chosen a 4-tier approach to help visualize SOA. ¹

- **Component – An individual executable module that performs a single, measurable task**
- **Service – A collection of components that provide a unique business service such as an order processing service or a general ledger service. The architecture permits services to be provisioned internally or externally with equal ease and transparent to the service consumer.**
- **Processes – A collection of components and services that are invoked in a particular sequence in order to mirror business processes.**



SOA BASIC STRUCTURE

Service Consumers are the people who exploit IT services to perform business functions. They are responsible for using services as agreed upon in Service Level Agreements (SLAs). They usually pay for the service, either directly or indirectly. They interface with IT to

determine new functions, changes to existing applications, or sunsetting applications no longer needed.

Application Architecture defines business templates or scripts that determine which services and/or components are invoked, in what order, and when they are invoked. This area of SOA is responsible for understanding changes in business processes, then using the knowledge to modify or enhance existing IT processes to satisfy them. The applications teams will integrate other existing services into the processes and, where gaps exist, develop or procure new ones. When using SOA, applications teams become integrators more than application coders.

COMMONLY USED TECHNOLOGIES

- HTTP
- SOAP
- Web Services
- XML
- J2EE

Service Architecture addresses the structure of services and how they are sourced. Some of the more common solutions to support this architecture are:

Middleware is the technology and protocols that permit the transparent movement of data and information, usually in increments called messages, across the IT enterprise infrastructure. Middleware also permits seamless integration of external services into internal processes.

Workflow utilities facilitate consistent execution of business processes. During construction of a workflow, discrete pieces of work, the order in which they are completed, and the people or business teams responsible for completing them are identified. A script is developed in the workflow utility. As work is assigned, the workflow utility manages the assigning and completion of the individual steps. Where steps are not completed within specified time periods, the workflow utility invokes escalation procedures to manage the delays.

Converters, sometimes called **mediators**, may be used to normalize data between diverse infrastructure, such as normalizing data between EBCDIC format on IBM Mainframes and ASCII format on other platforms. In video format terms, converters would permit European PAL format to be displayed on North American NTSC devices or vice versa.

Employing solutions such as those previously mentioned permits the application teams to seamlessly integrate components and services into scripts or flows that resemble business processes. When implemented correctly, components and services can be added, changed or removed without any impact to the service consumers short of any needed training.

TEAMQUEST TIP

SOA Interoperability Concept – Microsoft Plug and Play

Users plug technology components into their PC such as CD-ROM drives, speakers, or a mouse which are automatically detected, installed and communicate with the PC. The operating system architecture is transparent to the user although from an engineering perspective, the technologies involved to interact may be entirely different.

Component Architecture defines the make-up of components themselves and the infrastructure upon which they run. Components usually perform a single, discrete function such as “get customer record” or “compute interest rate.” Components observe architecture protocols for sending and receiving data and information between other components and services. XML is a commonly employed format for sharing data.

WHAT SOA MEANS TO CAPACITY MANAGEMENT

WHAT CHANGES?

Capacity management’s individual platform view is extended. End-to-end

application and business process views, which may include both internally and externally sourced components, are needed to support management reporting needs. Capacity plans must be determined looking across all affected infrastructure components to ensure service quality across business processes.

The need for modeling becomes more urgent because although there are fewer components to deal with, there is greater service quality risk due to application interdependencies of component sharing.

More involvement with the business early in design cycles, as the reuse of common components permits more precise estimation of new application implementation and ongoing support costs. Using SOA’s modular approach, it is easier to predict new application performance. Since in many cases, components from existing inventory will be used to satisfy requirements, capacity management can extract performance data and employ modeling tools to accurately predict the impacts of the additional usage on the IT infrastructure.

Reporting scope has to be expanded. Because of the shared environments, detailed reporting may not be as meaningful to business and IT management. A more meaningful approach will be to track the performance of each business process from end-to-end or an enterprise view.

SOA requires more capacity management involvement when designing and scaling the enterprise architecture. Technology components are just as important as the application development techniques. Quickly developed applications are of little use if the infrastructure cannot react to growth just as quickly. Therefore it is important to have capacity management and infrastructure engineering involved in SOA architecture plans and decisions.

WHAT STAYS THE SAME?

Capacity management processes and the work required to execute them mostly remain the same.

Applications still consume server and network resources so the need remains to gather performance data, store it for historical purposes, and generate management and progress reports from it.

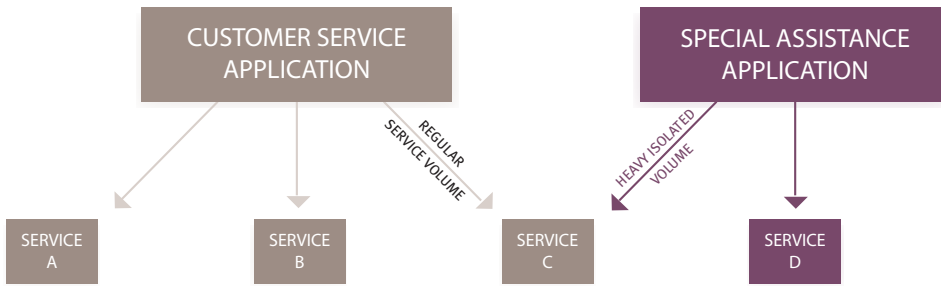
Business needs still need to be understood and translated into technology requirements. Those requirements still are needed to feed existing planning processes, interfacing with Finance to build, and then track financial plans supporting IT infrastructure growth.

There is still a need to tune applications to optimize the use of existing infrastructure assets, so the need to track trends and alert when potential problems may occur continue to exist. Bottlenecks still need to be identified before they negatively impact levels of service.

CAPACITY MANAGEMENT CHALLENGES

The same application component or service may be used across many business processes. Each process may generate work at different rates, depending upon needs. Therefore it may be necessary to find ways to identify and segregate business process resource usage internal to the components.

For example, if we look at a customer service application, the day-to-day cross section of customer types calling is varied enough that you can accurately predict future growth using average call metrics. Heavy query volumes could skew the average if you add a new service or process – using common modules and services – that gives special assistance to good customers who have large amounts of purchase history. If the different business processes grow at different rates, it could be difficult to accurately predict future resource requirements. Therefore it will be necessary to segregate usage in the common components by business process for more accurate predictions. See the image on the following page.



Provisioning or sourcing common services may be difficult. When sharing a particular service, you need to ensure

the lowest common denominator is satisfied. That lowest common denominator may require substantially different levels of service than the more frequent users of the service. For example, a mapping service used by customers to find stores may experience light usage but management prefers customers to find store locations any time of day or night. The warehouse application uses the same mapping service to optimize truck loading and delivery routes to the stores. Usage is heavy but the warehouse only uses the service weekdays between 8 a.m. and 5 p.m. As a result the common service could cost more since the lightest usage requires more expensive service options. Therefore one must understand the trade-offs between lower development costs by using a shared service and increased service costs by sourcing for the lowest common denominator.

During transition of legacy application functions to the new architecture, it may be difficult to accurately plan capacity positions. In many cases, the applications run in shared environments so it may not be clear as to what really goes away in the legacy environments.

Planning could further be complicated if legacy systems need to be available for historical purposes, such as regulatory reporting, for some undetermined period of time.

As an example, the legacy application could be converted to web services. Although the legacy piece would go away, some of the underlying OS support overhead may not and some utility programs may be used by other applications.

Recovering all resources consumed on a mainframe application is rare when it is moved to a different platform.

Larger, more comprehensive testing environments would be required to test all applications sharing common components because of interoperability complexity. This additional infrastructure could substantially increase testing expenses. As a result, capacity management may be called upon to employ modeling techniques to predict application performance based upon results produced in smaller test environments.

For outsourced services, capacity management will need to determine performance data requirements for inclusion in business process performance reporting. Obtaining data can be difficult if performance data accessibility is not addressed when the contracts are written. Capacity management must be involved in contract discussions to ensure needed data is available to ensure vendor service levels are met and business processes perform as expected.

Business processes may have durations measured in hours or days whereas IT process durations are usually measured in seconds or minutes. Capacity

management may need to devise ways to associate reporting of IT events to business durations.

CONCLUSIONS

As you can see, capacity management work to support SOA is not much different from today. The same processes prevail, with a change in scope from individual platforms to business processes.

Since the viewpoint changes, new reports need to be created to supplement those in place supporting existing platforms. Where external services are used, reporting needs to be put in place to ensure vendors deliver expected levels of service and when they do not, provide the data required to escalate and resolve the issues.

Modeling becomes even more necessary to ensure service quality, plan for the future, and find and correct potential bottlenecks prior to negative impacts to production service quality.

Testing rigor requires a larger scope, requiring more testing resources. Using shared components and services require test practices that span different business processes to ensure code and performance quality for all processes using shared components and services. More rigorous monitoring is needed and modeling may provide lower cost options than large testing infrastructures to assure application performance.

1 Bieberstein, Norbert, Bose Sanjay, Fiammante, Marc, Jones, Keith & Shah, Rawn (2006). *Service-Oriented Architecture Compass – Business Value, Planning and Enterprise Roadmap*. Upper Saddle River: Pearson.

Sprott, D, & Wilkes, L, (2004) *Understanding Service-Oriented Architecture*. Microsoft Architect Journal. Retrieved August 2006 from <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmaj/html/aj1soa.asp>.



TEAMQUEST CORPORATION

WWW.TEAMQUEST.COM

AMERICAS

ONE TEAMQUEST WAY
CLEAR LAKE, IOWA 50428
USA

+1 641 357-2700

+1 800 551-8326

INFO@TEAMQUEST.COM

EUROPE, MIDDLE EAST AND AFRICA

BOX 1125
405 23 GOTHENBURG
SWEDEN

+46 (0)31 80 95 00

UNITED KINGDOM

+44 1562 881 889

EMEA@TEAMQUEST.COM

ASIA PACIFIC

LEVEL 6, 170 QUEEN STREET
MELBOURNE, VIC 3000

AUSTRALIA

+61 3 9641 2288

ASIAPACIFIC@TEAMQUEST.COM

Copyright © 2006 TeamQuest Corporation
All Rights Reserved

TeamQuest and the TeamQuest logo are registered trademarks in the US, EU, and elsewhere. All other trademarks and service marks are the property of their respective owners. No use of a third-party mark is to be construed to mean such mark's owner endorses TeamQuest products or services.

The names, places and/or events used in this publication are purely fictitious and are not intended to correspond to any real individual, group, company or event. Any similarity or likeness to any real individual, company or event is purely coincidental and unintentional. NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a license agreement. The only warranties made, remedies given, and liability accepted by TeamQuest, if any, with respect to the products described in this document are set forth in such license agreement. TeamQuest cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions. U.S. Government Rights. All documents, product and related material provided to the U.S. Government are provided and delivered subject to the commercial license rights and restrictions described in the governing license agreement. All rights not expressly granted therein are reserved.