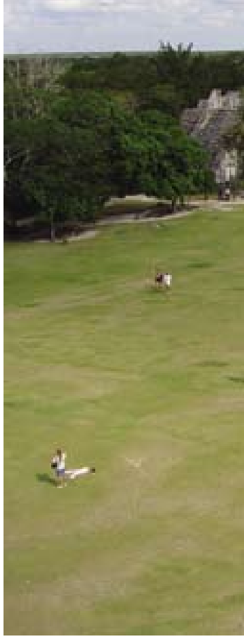
Build Your System i™ SOA Roadmap

By: Alex Nubla



Introduction

SOA Projects tend to be large initiatives with lots of risk and potential reward associated with them. The ROI (Return on Investment) is very hard to quantify and thus hard to sell, which brings about the need for a solid process and methodology to ensure success of the project.

SOA is a set of tools, technologies, frameworks, and best practices that enables the quick and easy implementation of services. In addition, the process of developing SOA uses a methodology for identifying reusable services in our applications and organization.

The architecture must focus on allowing us to identify, build, exchange and maintain our business processes as services rather than as large, monolithic applications. These applications are often termed as "Legacy Applications". SOA existed well before Web Services (remember CORBA.) The common misconception is that SOA must be implemented using Web Services technology. In reality, Web Services are only a small part of SOA. Web Services are one of the ways to provide access and invocation of some services in an SOA environment. Other services are better implemented with other technologies. Asynchronous services for instance, might be better served using asynchronous messaging like MQ or JMS. Heavy-duty services, with high volumes of data that need to be passed between consumer and provider, can use XML as a session initiation.

SOA promises to allow one system to interoperate seamlessly with another.

Gartner predicts that 20% of Fortune 1000 companies will deploy an enterprise information management (EIM) strategy in support of SOA. Gartner estimates that SOA will provide the basis for 80% of new development projects. SOA will also enable organization to increase code reuse by over 100% by 2008, saving developers time and energy. SOA will be the guiding principle when creating mission-critical applications and processes. Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises. By 2010, at least 65% of large organizations will have more than 35% of their application portfolio SOA-based, which is up from fewer than 5% of organizations in 2005.

SOA is targeted at reducing the complexities involved in software development. It addresses issues with multiple platforms, distributed software and application integration. SOA provides an application architecture in which we define processes as serves that have well-defined interfaces. The services can be dynamically invoked within the network. SOA enables faster time to delivery of business processes and cost reduction resulting from reductions in development and maintenance cost.

The Problem of SOA Complexity

IT complexity remains a huge burden for most enterprises. While SOA promises to enable business agility, "Out of control IT complexity often turns these systems into the key obstacle to business agility instead", according to Gartner (Applied SOA: Conquering IT complexity through software architecture). A number of factors have led to this complexity.

- The history of enterprise architecture can be seen as evolution rather than revolution. Instead of ripping and replacing existing systems, most enterprises have grafted new capabilities onto what they already have, resulting in a heterogeneous environment made up of both legacy applications and newer, service-based technologies.
- Shifting architecture paradigms continue to introduce complexity. For example, when outages occurred in the old client-server architecture, we could easily pinpoint a failure, and possible sources were limited to the handful of systems. In a service-enabled composite environment, the application server is a central point of coordination across multiple client-server systems, and a failure can occur anywhere along the transaction path.
- Business factors such as a change in organizational structure, additional business functions, or acquisition of new businesses can also increase complexity of enterprise architecture. Associates that previously were experts in specific areas of enterprise functionality can find themselves working in new departments, leaving behind knowledge gaps. As a result, additional application functionality tends to be "glued on," making the service more complex.
- The ability to rapidly deploy new business functions, or versions, makes it difficult to accurately map out the application architecture. Almost as soon as everything is

TM: System i is a trademark of IBM Corporation

Alex Nubla works as the Enterprise Architect for Health Net. He used to write technical articles for Powerbuilder Magazine, Midrange-Computing Magazine and iSeries News Magazine (even way back in News 3/x days).

Alex lives in Southern California now, after relocating from Charlotte, NC where he worked for Bank of America as their V.P. in-charge of the Technical Support for System i. Alex has been gone from writing for the past 9 years because of high demand on consulting work. He has written numerous utilities and always makes it available for free in the System i Community.

Alex has always been a great supporter of the System i. He has butt-heads with other industry folks who refers the System i as old technology. "I don't want to be portrayed as an Old Bigot who loves System i, but the truth is - System i is here to stay. And please, don't call it a Legacy System."

mapped, the chart becomes out of date. In this example of rapidly evolving environments, how can someone troubleshooting a single transaction and know which service was deployed at the time of error?

SOA promises to allow one system to interoperate seamlessly with another. Traditionally, information systems have operated independently, managing their own data, and remained blissfully unaware of other systems within the organization. The advent of Web-based technology and the requirement to share data from disparate sources led to the creation of point-to-point communication between applications. SOA replaces hard-coded links with a loosely coupled approach. This boils down to the fact that more attention must be paid to the integration point.

Developers of one service typically don't know much about another service, other than the information provided — for example by WSDL. Therefore, they have no means of diagnosing problems caused by a service, and they may not be able to identify which service caused the issue. This lack of visibility is one of the biggest challenges with SOA implementation. An effective service management solution is important in the design of our SOA enterprise.

Managing performance also becomes challenging when a particular component or service is shared among several applications. Various applications have different load requirements. New applications can be deployed anytime, and the service administrator may not be alerted regarding the increase in usage. The varying application usages make it hard to plan for and predict service load stresses.

As SOA becomes our standard for all application environments, the difficulty of managing them and the need for transaction visibility grows exponentially. The very fact that SOA makes it easier to integrate java-based and legacy systems, means that management tools must be able to reach across and within all of our systems to discover relationships and dependencies.

The 5 Stages of Software Processes (Maturity)

Stage 1. Initial Services

This is the current state of most organizations. There is no separation of architecture from projects. Typical projects are designed by project leader, with the help of Line of Business, working in Silo. Success in the organization depends on the competence and heroics of people in the organization and not on the use of proven processes. In spite of its challenges, this stage often produces products and services that work. However, it frequently exceeds budget, increases the cost of delivery and disrupts the project schedule. The quality of codes is typically not reusable and hard to maintain.

Stage 2. Repeatable / Architected Services

Stage 2 represents the initial learning and early phase of SOA adoption. It includes initial R&D activities testing of SOA technologies in "Sand-box" environments. Organization moves toward some form of EAI (Enterprise Application Integration). Projects are done to meet specific need to "implement functionality", while trying out specific technologies, recommended by the enterprise architect. The element of enterprise architecture emerges. Enterprise architects help put in place structure and foster communications between teams.

Project teams define reusable architecture that they use from one project to another. Organization componentized or modularizes major or critical part of application portfolio. The result is some level of reuse of architectural components. The benefits of reusable process are recognized. Software development and deployment are properly scheduled, and improve the overall quality of the software. The key business benefit of this stage is development and deployment cost reduction through the use of SOA standard infrastructure and components as to using older technologies and cost accumulated through multiple unique one-time projects. Services are exposed from consumption internally, not quite on a large scale, but acts as a service provider nonetheless.

Stage 3. Collaborative / Composite Services

The focus of this stage is the partnership between business organizations, information technology, and enterprise architects in order to assure that the use of SOA provides clear business responsiveness. It focuses on the implementation of internal and external business processes.

The standards, process descriptions, technology polices and procedures for a project are tailored to suit particular project, or organizational unit. Therefore are more consistent. At the end of the day, IT recognizes that they are serving a business.

Stage 4. Measured / Managed Services

At this stage, enterprise architects start defining path to SOA. Quantitative objectives are established and are used as criteria in managing process. Quantitative objectives are based on the needs of customer, end user, organization, and process implementers.

We can only be classified at this stage if our SOA initiatives seek buy-in from business. Governance and rewards policies should be defined. Support levels need to be established with clear understanding of whom to call, when and for what.

A framework for analysts to define services must be established. How does LOB identify potential services that it can expose? Who is responsible for building and maintaining this service? Who pays for it? These are some of the questions the SOA plan at this stage should answer.

Stage 5. Optimized Service

An organization continually improves its processes based on the quantitative measurements from the previous stage. It focuses on continually improving processes and performance through incremental and technological improvement.

A framework is in place for each team to consume and expose services. At this level, organizations truly explore the meaning of SOA. The organization starts to figure out how to exchange services with business partners, suppliers, and customers. Business service level reuse, not just technology component reuse, is the core to the architecture to achieve maximum agility.

A critical distinction between Stage 4 and 5 is the type of process addressed by business. At stage 4, the organization is concerned with addressing the cause of the process and the predictability of the result. At stage 5, the organization is concerned with the common cause of the process, changing the process to shift performance, and achieving the quantitative objective that meet or exceeds expectations.

The process of adopting SOA starts at Stage 1 (Initial Service). It gradually evolves to a technology adoption within a group that is engaging in the SOA style of projects, using tools, technologies, and methodologies that enable the creation of proofs of concept (POC). (See **Figure 1**.)

Assessments can be done during adoption. The functionality of the proposed services needs to be evaluated, validated, and verified by enterprise architects.

Once the technology adoption stage starts to achieve maturity, the results start to trickle into the line of business. LOB sees the value in sharing services across line of businesses to eliminate redundancy, and having a single point of access to existing functionality. This reduces cost and complexity, and increases flexibility.

Business starts using services to access common functionality, often starting at a

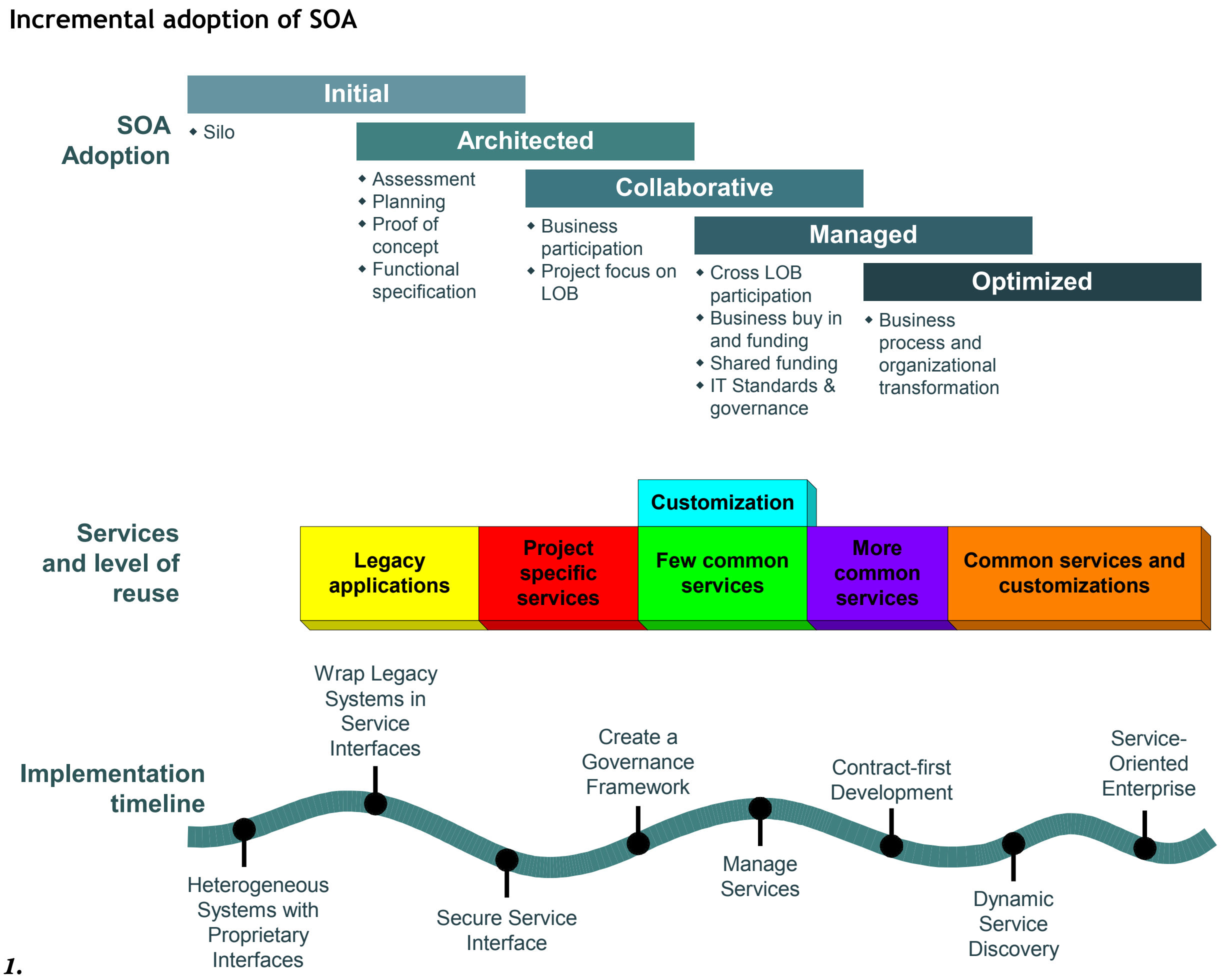

Incremental adoption of SOA
SOA Adoption
Initial
• Silo
Architected
• Assessment
• Planning
• Proof of concept
• Functional specification
Collaborative
• Business participation
• Project focus on LOB
Managed
• Cross LOB participation
• Business buy in and funding
• Shared funding
• IT Standards & governance
Optimized
• Business process and organizational transformation
Services and level of reuse
Legacy applications
Project specific services
Customization
Few common services
More common services
Common services and customizations
Implementation timeline
Heterogeneous Systems with Proprietary Interfaces
Wrap Legacy Systems in Service Interfaces
Secure Service Interface
Create a Governance Framework
Manage Services
Contract-first Development
Dynamic Service Discovery
Service-Oriented Enterprise

Figure 1.

technology level and then moving on to the business level (which provides greater value). Businesses can begin to partition business functions to eliminate redundancy. This comes to a certain degree of business buy-in as it evolves towards consolidation of business functions across lines of business.

At an enterprise level, standards and governance are adopted. Adoption will include protocols, tools, standards, and service models. These are extended across the organization.

Business Rules in SOA

It is at the higher stages (business rules, orchestration/manage, contract) where changes could ripple through other service consumers and producers. For example, a change in the business process could impact multiple components, such as the management of services that coordinates the interaction between multiple services, or the contract of services are impacted due to a change of the relationship mapped within the architecture. This begs the question regarding how we define "business rules" within SOA. Do we view them componentized in relational to the architecture? Or do we have a different perspective of the business process within the context of SOA?

Legacy Applications—Does SOA mean Save Our Assets?

All companies are under continual pressure to deliver up-to-the-minute business data to customers while keeping costs in line. Business users demand real time reports and metrics. Web based tools have been provided to show up-to-date information. These relentless imperatives have been placing new demands on legacy applications. But the term "legacy" itself has taken a negative connotation. And with the demands to adopt newer Web-based technologies, it is no surprise that organizations have been debating whether or not to "rip and replace" legacy applications in order to modernize IT infrastructure. However, "rip and replace" is not usually the right answer—especially when it comes to SOA strategy.

First of all, most legacy applications are mission-critical—they run the backbone of the business. The applications house data and business processes, and represent years of intellectual property. In addition, "rip and replace" projects are costly and prone to failure. However, legacy applications do have some limitations. They are often disconnected from the enterprise. They house silos of data that are difficult to integrate with other silos.

Let us consider the "preserve and extend" approach. It is less costly and less risky. By preserving and extending our legacy applications, we capitalized on longstanding strengths—reliability, security, and performance. "Preserve and extend" becomes the commitment to maintain the past and present, plus propels you into the future. However, we must never forget that legacy applications are often written in monolithic fashion, some of which have been around for decades. Can these applications be exposed easily using Web Services with a little point and click tooling? Or is it going to require significant reengineering of the applications before they can truly take part in the dynamic, federated, on demand future of SOA? The answer may both be "Yes."

HIGH AVAILABILITY / DISASTER RECOVERY

A dog is a dog no matter how you dress it.

At iTera, we focus on two things – improving reliability and decreasing complexity. That's why one out of three iTera sales actually replace competitor software. We're the best IBM System i5 disaster recovery and high availability solution around… period. Don't believe us? Call our customers. They know what works… and what doesn't (no matter how you dress it).

Read our case studies at www.iterainc.com

Reliable.
Easy-to-Use.
Affordable.

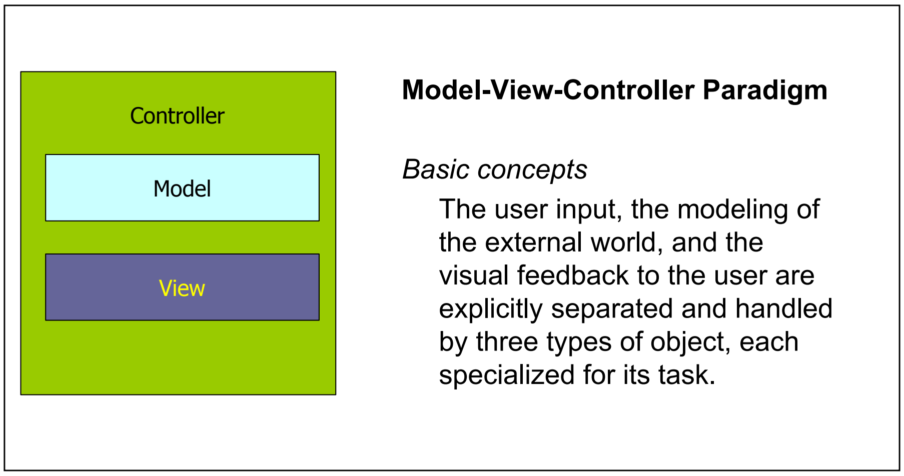
iTera
No Time for Downtime™

www.iterainc.com • 800-957-4511 • info@iterainc.com

iTera Systems Continuity	Asia Pacific	Europe/Middle East
© Copyright 2006, iTera Inc.	+852 3177 8211	+44 1 256 782 988
	infoap@iterainc.com	infoemea@iterainc.com

The monolithic nature of older legacy applications was in part due to the philosophy in the design (e.g. user interface), business logic, and data access logic—all of which are contained or built into the same program. This makes it virtually impossible to reuse portions of these applications or to integrate them into other applications. In some instances, companies included several different interfaces (Web, Java, green-screen), different data sources and data access layers, business logic contained in different modules, and lastly integrated process flow. Applications grew to be more monolithic, in the sense that they are contained within organizational silos, and interaction within other silos happened via messaging layers or file-transfer mechanisms.

The initial approach to take would be to map out where one system begins and another ends. We need to make the boundaries between silos manageable. We must identify where obscurity happens and develop elemental services to form the new basis of composite applications. We need to make these services available to existing legacy applications allowing the slow transition of reusable components. At the same time, we revisit our monolithic approach of coding and progressively transform them into more loosely coupled, reusable components that can be easily aggregated and integrated to deliver new services. Some basic elemental services remain distinct. These services will form the basis of the new "composite" applications.



Model-View-Controller Paradigm

Basic concepts

The user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object, each specialized for its task.

The next approach is to create these services as Web Services. Programs (or service programs / procedures) written in ILE RPG resides in the System i for example, and a Java Wrapper is created to invoke the RPG program—the resulting service is created as a WSDL. We should leverage the appropriate host integration tool (IBM Websphere, BEA Aqualogic or Microsoft .NET) so we can quickly abstract what was once a monolithic, terminal-based application into a set of services. The end result would be enhanced customer satisfaction, reduced cost, with better internal efficiencies and rapid development platforms in place for on-going projects.

Decoupling Legacy Applications

Partitioning applications improves code reuse, simplifies maintenance, and makes it easier to mix and match technologies. This allows us to leverage developer resources more effectively because different developers can work on different parts of a larger project. Developing a flexible application architecture is the goal—but delivering / leveraging the architecture to provide new user interfaces is an important secondary goal.

To accomplish this, we must uncouple business logic from user-interfaces within the current applications—making it possible to modify user interfaces without affecting business logic, and vice versa. Lay the groundwork so that we can later rebuild the user interface (e.g. JSP now, JavaServer Faces technology later). By decoupling the user interface, we can also accommodate a variety of user interfaces (e.g. Blackberry and PDA to full browser).

Model-View-Controller (MVC) is a framework widely being introduced in the System i SOA architecture. MVC separates the application into three functional parts: the model or business function, the view or the user interface, and the controller that handles the communication between the model and view. (See **Figure 2**.)

As we review our existing legacy applications, it is very important to keep the MVC approach in mind. It's tempting to couple the controller and model together too tightly. For example (in a Healthcare application for claims adjudication) to maintain the separation between the controller and model, we could have the controller launch a comprehensive "claims validation" in the model, rather than invoking the get claims information function directly. The claim validation model handles the task of getting the claims information as well as verifying all the header and detail information for the claim.

Because MVC uncouples the view from the controller and model, we can theoretically create the view using any technology we choose. This stage can gradually center on a browser-based view written in JSP for example. JavaBeans can be embedded in the JSP. By using servlets and JavaBean, we can easily maintain code components to a single task—which simplifies maintenance and creates opportunity for code reuse. We can easily support different user interfaces for the same application. For example, a claims application may support a JSP interface for internal claims adjusters, a portlet interface for external providers and a batch interface that accepts XML claims transaction using MQ.

We will need to invest in training to support that effort because for most System i developers, writing GUI components is a new skill. For example, for MVC application to succeed, a group of developers with Web design and JSP skills will create the view; another group with ILE RPG skills may write the model; and the third group with Java skills will write the controller and deploy the application.

IT Governance Framework

Governance is the ability to unite model and code so that it cannot drift apart.

JOIN BRAINS II FOR YOUR GROWTH AND SUCCESS (AND A $100,000 SIGNING REWARD)

Brains II is looking for Canada's best technology people to join our team. Because of our rapid growth, we need experienced technical and sales professionals who specialize in IBM and other leading manufacturers' solutions.

The Brains II Advantage
When you join Brains II, you join one of Canada's premier information technology companies. Benefit from our national presence, multi-platform/ multi-vendor product line and superior infrastructure (featuring progressive marketing, technical support and professional leadership).

Your Incentive to Join
We're offering a signing bonus to select energetic and ambitious IT professionals. The bonus may exceed $100,000 for leading performers. And because Brains II sells maintenance and support along with our technology solutions, you'll be able to service customers better and make more on every deal—giving you personal growth and earning potential like you never knew existed.

To learn more about these and other career opportunities, visit the careers section of our website at www.brainsii.com/careers or send your resume and cover letter to resumes@brainsii.com.

"I founded Brains II in 1979 to help customers throughout Canada solve business problems with information technology. As an accomplished entrepreneur, I also learned that people make this company successful. So, I am committed to giving you an environment in which your personal development is my top priority. That way, you and Brains II will provide the best service and grow more successful together."

Charles Hanna
CEO and Founder
Brains II

The Brains II brand is one of the most trusted names in the IT industry.
With the acquisitions of Memorex-Telex, DocuPartners, Circle Computers and many others, Brains II provides integrated solutions from leading manufacturers including IBM, HP, Sun, Xerox and many others for thousands of customers across Canada.

Brains II

IBM® Business Partner™

IT Governance is defined as "the decision rights and accountability framework for encouraging desirable behavior in the use of IT." IT Governance is seen as a framework that ensures that information technology decisions consider the business' goals and objectives—similar to ways in which corporate governance aids the firm in ensuring that key decisions are consistent with corporate vision, values and strategy. IT Governance ensures that IT-related decisions match companywide objectives. What is not IT Governance? IT Governance is not about what specific decisions are made. (That is management). Rather, it is about systematically determining who makes each decision (a decision right), who has input to a decision (an input right), and how these people (or groups) are held accountable for their roles.

Governance management should design IT governance around the company's objective and performance goals. It should involve senior executives taking the lead and allocating resources, attention, and support. It should take advantage of some of the mature business governance like an existing steering committee, project review committee, and resource management. Since governance requires IT to learn new roles, it will take time. It can be seen as a continuous process of aligning corporate and IT strategy. Remember that a change in governance is required with a change in objective—so don't forget that redesign is also required within your framework.

Accountability

Companies with the most effective IT Governance have more senior management involvement. IT Governance must have an owner and accountabilities. The CIO must be accountable for IT governance performance. A group of senior business and IT managers help design and implement IT Governance. This will be the whole committee in charge of IT governance.

The design of governance must not be done in isolation. Business Managers are expected to contribute to IT governance as they would contribute to the governance of financial and other key assets.

Transparency and Education

Make the IT governance mechanism transparent to all managers. The more IT decisions are made covertly, the less confidence people will have in the structure and less willing to participate in the rules. The clearer the IT governance, the more people will follow them. The more "special deals" are made, the less confidence there is in the process and the more workarounds are used. The less confidence there is in IT governance, the less willingness there is to play by the rules designed to increase corporate performance. NO SPECIAL DEALS!

Exception-Handling Process

Successful businesses continuously build new opportunities, some of which will not be supported by existing IT policies. To support these opportunities, IT governance must include a clearly stated exception handling process—to bring issues out into the open, allow debate, and foster organizational learning. Without an exception process, unauthorized exceptions will continue to occur with little enterprise-wide learning.

Who reads the TUG magazine?

Over 5,000 IT professionals in the GTA, and across the country.

Get inside their minds...

ADVERTISE!
in the TUG magazine

TUG

We are tightly focused on the midrange space.

Ron Campitelli 905-893-8217
Wende Boddy 905-607-2546

Formally approved exceptions offer benefits in addition to formalizing organizational learning. Exceptions serve as a release valve, releasing the enterprise built-up stress. Managers become frustrated if they are told they cannot do something even if they are sure it benefits the business. Stress increases and the exception process provides a transparent channel to release frustration without threatening IT Governance.

Contract-first Development

Business analysts work with users to define requirements into contracts. These contracts acts as marching orders for component development. These represent requirements as well as test plans that an analyst can execute to guarantee that the services meet the requirements.

Lessons Learned (or to be learned)

SOA adoption requires:

- Blend of strategic and tactical perspective
- Understanding of "real" business opportunity and value
- More than just an architecture—it requires a project profile, processes, organization and management

Why won't SOA be adopted?

- Lack of awareness at all levels (CEO and CIO levels) on how SOA may help companies become and stay competitive
- An IT culture that is resistant to change (CIO must drive SOA to ensure IT adoption)
- Certain business patterns that are not agreeable to SOA

What will be adopted instead?

- A continuation of what we have now
- More tightly-coupled and inflexible applications
- Point-to-point interfaces
- Lots of replicated and unmanaged data repositories

Make a note!

- Tools and technologies will not automatically give you SOA. Do not rush to deploy merely technological solutions.
- SOA without good data is doomed to failure. Do not ignore master data, quality of data, and security issues.
- SOA without governance will not realize its full potential.
- SOA is a LONG TERM cultural shift. Do not underestimate the cultural issues surrounding changes.
- Weight in your priorities. Business case = innovation + agility + total cost of ownership.
- Implement in increments. Start small -> Deliver -> in Phases -> Total delivery with Value. TUG

TUG Supports COMMON Education Foundation

By Vaughn Dragland

Education Foundation representatives Laura Ubelhor and Michelle August

During the Fall 2006 COMMON Conference in Miami Beach, Florida, TUG donated a couple of terrific items to the Silent Auction, in support of the COMMON Education Foundation[1].

Silent Auction table

The first item was a full-page advertisement in the TUG magazine (worth US$2500) which was picked up by **PlanetJ**, for a winning bid of US$751.

The second item was a full-conference registration at TEC '007, TUG's annual 3-day technical education conference, April 17–19, 2007 (worth US$825). **Doreen Hannon** from Guelph, Ontario, had the winning bid of US$320.

All of the proceeds will go to the COMMON Education Foundation. Thanks to all who participated in this worthy cause! TUG

Doreen Hannon - senior programmer/analyst - Information Systems, Homewood Health Centre Inc., Guelph, Ontario

[1]The COMMON Education Foundation promotes higher education in the information technology area. It does so in several ways including awarding tuition reimbursement scholarships to students attending accredited universities, and by providing scholarships to instructors at an Academic Initiative for System i college to attend COMMON conferences and IBM Summer School.

To fulfill these and other goals, the Foundation raises money through a variety of methods, one being the Silent Auction held at each COMMON conference.