Back to the Future with SOA: The Importance of Governance

By Axel Angeli and Lynton Grice

Editor's Note: *Our parents dreaded 1984 and the predicted appearance of "Big Brother". Our generation was more concerned with Y2K and 2001. What future event or technological advancement will our children worry about? At SAPtips, we can't claim to predict the future, but with the assistance of NetWeaver gurus Axel Angeli and Lynton Grice, we can give you a pretty good preview of the role of SOA in the years to come. In this "strap on your seat belts" white paper, our forward-thinking duo present their ideas about why the SOA architecture has experienced slow adoption, what has changed, and why they anticipate an SOA-driven future in ERP-land. What's the key to moving ahead with SOA? Axel and Lynton say that standards and governance will determine whether we see evolution or revolution.*

Introduction

In today's business environment change is inevitable. Heightened competition, mergers, and increasingly more complex technologies and products around every corner have increased the pressure on companies to become more agile. The Service-Oriented Architecture (SOA) is deemed to be the strategic approach to fulfil these ambitions, and you cannot but see it as inevitable for the future.

SOA will bring you away from centrally controlled "aristocracy" (like software application landscapes) and will transform IT into a marketplace of freely combinable internal and external components that will be offered to other applications as services. No longer will the potential requestor of a service need to beg for assistance, but it will be the provider of the service that will have to woo the consumer and keep its "service offer" in high demand.

By 2015 enterprise computing will be handled via Internet-based, outsourced Web services making classical centralized computing a niche market.

Service Oriented Architecture

The new buzzword "SOA" appears with increasing frequency in the headlines of technical and business publications. Few IT-focused forums or SIGs seem to be able to withstand the temptation to embellish their organization's banners with at least a mention of SOA. SOA – the Service Oriented Architecture – is, however, far more than just hype; it is a milestone in the evolution of computer architecture that will lift commercial computing from loosely connected computers to a "grown-up" automated communication network where software is no longer seen as rigid, inflexible technology components, but as an abundance of agile, self-adapting, and competing computer-based services.

A Service Oriented Architecture transforms a structure-less and randomly compiled computer network into a mature networked operating system that delivers the architectural grounds for agile, component-based software services in the form of a simple communication framework.

The SOA provides for the essential basic services, like data transformation, protocol conversation, transactional execution, guaranteed data delivery, safety and security, and central repositories that publish available services along with access and compatibility information. SOA liberates distributed software development from deviating and repeating tasks, and hence gradually introduces services that communicate through a small set of common (and easily

understood) language and protocol standards, while allowing highly sophisticated algorithms (behind the services' wall) using technology and programming aids of the developer's choice.

The effect of SOA on the computing world will be tremendous. The evidence of this can already be observed in the World Wide Web where under the humble name of "Web 2", dynamic Web sites are popping up everywhere. The Wikipedia Website introduced the wiki concept, that is, Web pages that can be edited directly from the browsers, and allow several users to make changes simultaneously to the same document. Amazon.com offers unlimited storage space to dump data to their data centers for backup purposes, or to be used as a message queue. Google teaches the giants of computing how a portal should indeed look; with the ability to arrange component placements on the fly (in the browser).

Seeing all this, it is no wonder that computing giants like SAP® AG, IBM, Oracle, Microsoft, and many others sometimes rush to market for fear of "missing the train". They know that their old concept of centralized computing services (that relies on adoption of roughly configured, but comprehensive ERP software) will no longer be a viable way to grow business. They recognize that they are not positioned to compete with the agile, flexible, entrepreneurial crowd, and that also threatens their long-tem success. So the big dinosaurs, sensing extinction, all develop their own path into Service Oriented Architecture. Some do it more heartily, some less. IBM is in it full force; Microsoft, too. Oracle has no other choice, and SAP finally got on the right track with its ESA initiative.

With all this attention, we feel it is time to look at how this will potentially affect our way of doing business in the near future. We're bringing out our crystal ball for a small glimpse into the future. But never forget what every witch and magician's apprentice will tell you:

"If you want to predict the future, you just need to look into the past."

How Will IT Look in 2015?

How time flies when you're having fun! Well here we are, writing in the year 2015. We, as SAPtips analysts, take pride in being able to give our readers the latest snapshot on the present developments in IT. Wow, how the world of IT has changed in less than ten years. The enterprise computing market is dominated by two giants: "Microsoft SAP" and, once more, IBM. In an unfriendly move but with the support of many end users, IBM acquired all the software components from Oracle (and hence erased a competitor completely from the market). Despite strong fears from economists, both giants did not dominate businesses after all and failed to gain the market share that they were initially expected to achieve. Only 30% of the software sales are now affected by the two giants.

What has been recognized is that there is a dramatic change in the self-understanding of IT, and that there are fewer and fewer centralized ERP dinosaurs. They are regarded as legacy systems and many big enterprises are still struggling to find a way to refurbish their computing landscape with the proper functioning that they depend on so heavily. These companies do not lack the technical excellence, but still the organizations struggle as they did 10, 20, 30, and 50 years ago. We can observe that the inflexibility of their IT landscape has led to severe losses in the total business, caused by extremely high IT costs and the inability to follow the increasing demand for agility in the marketplace.

The more dynamic customers have slackened off of buying the "latest and greatest" from their ERP vendors, but rather concentrate on getting the "right mix" of applications and services that sit on top of their ERP system and messaging backbone (unless they already consume essential services from external providers). The significance of ERP has therefore been lowered, and has been degraded from the diva of enterprise computing to one of many similar services out there.

Generally, more and more companies will begin orchestrating complex[1] and compound business processes using a mixture of both in-house and Internet services, and many companies out there are now running their IT systems almost entirely over the Web. Instead of hosting a fully featured suite of ERP applications on their own server, they purchase the functionality on demand from vendors who offer the services over the Internet. And the very prudent ones have decided to buy services from different competitive sources, just so that they are able to switch between them and not expose themselves to the peril of blackmail.

The H2O Master Company

H2O Chemicals in San Francisco, for instance, is a company that has put its stakes completely in an early adoption of Internet-based service oriented architecture. Instead of installing their own ERP system or enduring a lengthy change management process, the company decided to buy services from the best companies in the world in their respective areas.

Early adoption and going against what others see as the "norm" can most certainly have its benefits. With this insight in mind, H2O followed a "somewhat different approach" to the logistics section of the business and quickly saw great results driven by the flexibility of their chosen business model. Since then H2O has become the leading manufacturing example; similar to what Amazon.com was to the mail order business.

What H2O did in 2005 was steer its attention to formulating a solid SOA that allowed them to buy elementary and compound services from multiple vendors that suited the demand from the business line. All consumed services were loosely and redundantly plugged into their Service Bus architecture, and allowed for quick and easy replacement when a better offer came along. In the beginning, H2O faced the challenge to define quite a lot of services themselves by wrapping many functions of their ERP systems into a service module.

What looked like a bold move back then was practically nothing else than what the company successfully practiced in their physical logistics section. There was a good old time where a company was widely an autonomous biosphere. They did not only cater to the facilities to produce their goods, but they had all the necessary supporting infrastructure in-house; from the laundry, to the cantina, health care, limo service, conference centers, lodging houses, and some even maintained their own retirement resorts. The fat years being over and facing more competition, many companies have learned that sharing a professional laundry service, hiring taxis, and making service-level agreements with providers like food caterers or hotels, are a means to save real money while still attaining higher quality.

In 1992 H2O even outsourced the complex logistics in transporting chemicals to many countries around the world to a number of specialized forwarding companies. It proved to be much more efficient to entrust these shipments to exotic companies whose services included the handling of packaging (like containers) and their returns.

What proved to be an important factor was the decision to set up all logistic services redundantly. Instead of outsourcing shipments to the least expensive forwarder, they opted for a mix of several providers in different price ranges and differing specialized areas (like high or low weighted products, express or collect services, local offices in foreign countries, and many more). In fact they let the providers compete not only for best price but also for quality of service. The decision for which service provider should be used had always been left up to the executive who asked for

---

[1] Please keep in mind that we use *complex* in its stricter mathematical meaning. As such *complex* is not a synonym for complicated or difficult, but stands for non-linear and multi-dimensional.

the service. This extremely successful model of service distribution had been taken as the role model for the service oriented enterprise.

Making the Step to SOA

Each major technology goes through "waves" of evolution with each wave building on the previous one…and ERP is no exception. The first wave of ERP was the introduction of computers to the manufacturing environment. The second wave saw the emergence of more specialized applications in the ERP arena. Web Services and ERP have now joined forces and have constituted the third wave of ERP solutions, and has given ERP a new lease on life!

So what is the big deal with combining Web Services and ERP? Well besides being able to make use of "cheaper services" typically produced in the Indian and Eastern European markets, management will realize this combination of ERP and Web Services will ease integration and will further reduce costs by using the hosted application model.

Today - in 2015 AD - H2O Chemicals makes use of numerous services over the Web, for example:

- Sales order maintenance provided by "End of Universe Ltd."
- Procurement Services are offered by "World Logistics", a steadily growing shipment company that grew on the same principles as Amazon
- MRP is done by "The Holistic Society", a 10-headcount company that holds patents in statistically approved methods for reliable MRP for non-discrete industries
- Financials have been contracted to a local financial controller who will be responsible for a timely and accurate balance and tax reporting
- The company information bank, formerly called business warehouse, is handled by a company that sells online storage and had had its breakthrough by introducing a true object-oriented database
- The company's document management system was been established in 2007 based on a blend of several Wiki style engines from open source Typo3 via Atlassian Confluence and Jira to Microsoft FrontPage

H2O did, however, not stop with consuming outsourced services. They also offer some of their specialties as services to other companies.

- A clever graphical tool for maintaining bills of materials (BOMs)
- A service to produce the declarations for dangerous goods

Via these services, a large portion of the costs for consuming outsourced services are compensated. Good examples are a number of banks that offer electronic financial services, which they initially developed for their own core business, to their major industrial clients as well.

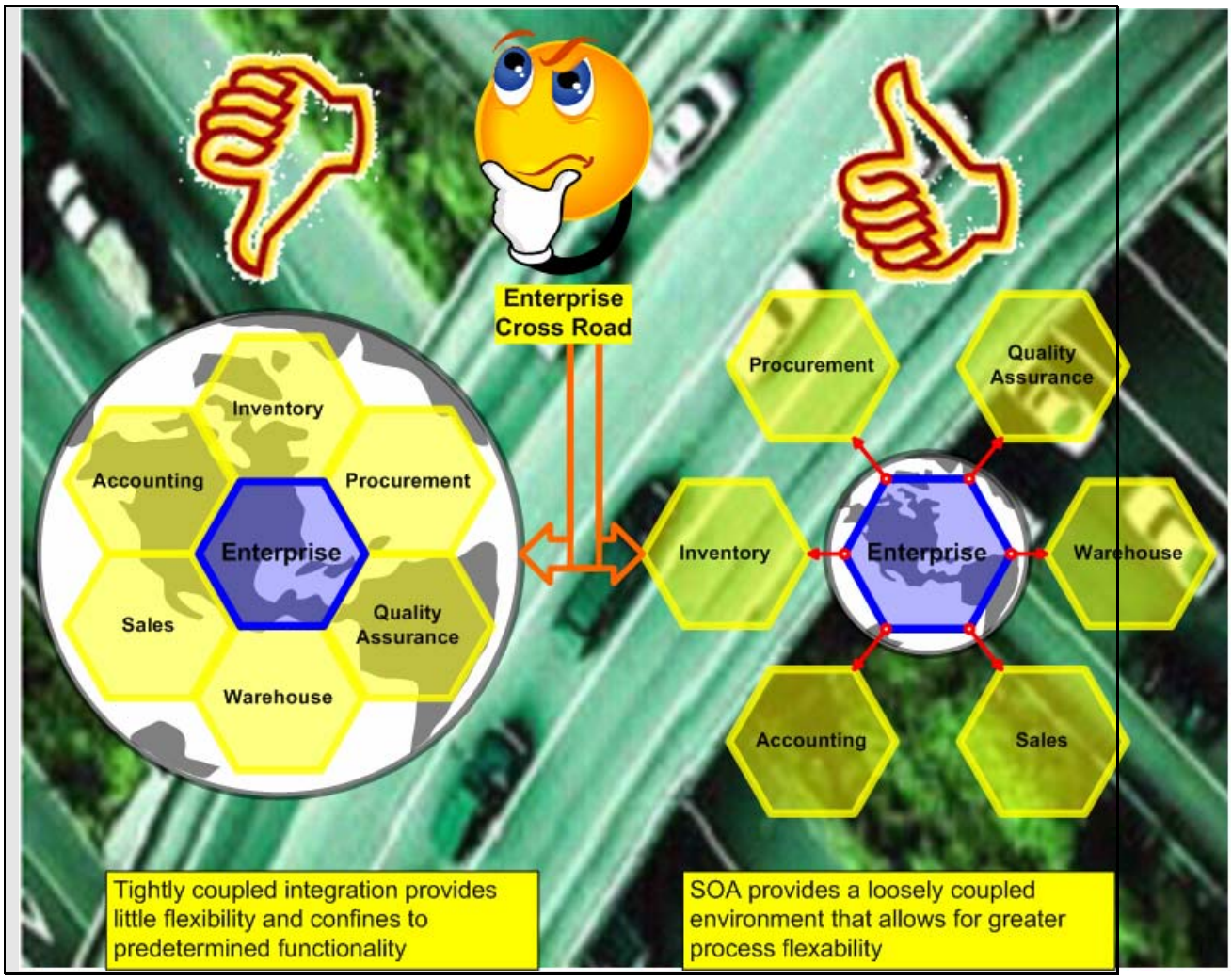

Figure 1: Using Internal and External Services

Back to the Present: Why Will SOA Succeed Now When It Failed Before?

"Even the longest journey starts with the first step!" Common wisdom

SOA comes with the same promises that have been around for many, many years, and it has been implemented here and implemented there, but never with a break-through success. SOA, however, in combination with the self-running success of Web services, is marching through, and nothing at all can stop it.

In the past ten years, integration was widely dominated by Enterprise Application Integration (EAI) efforts. The goal then was to automate the computer-to-computer communication to avoid redundant data entry and data loss or latency between software applications. EAI has always been there but has never really been a true shining star. EAI, to a certain degree, failed to deliver its promise for a number of reasons, namely:

- Overly complex protocol requirements
- Lack of agility and ability to keep pace with business change

- Low visibility and transparency over the process
- EAI is generally data-centric and not process-centric

EAI is most certainly not going away, on the contrary! EAI will merely evolve and will become more flexible and agile as companies increasingly incorporate Web Services into their EAI portfolio.

Simple – Really Simple, That Is the Secret

Why has there been no break-through before? Why have all the former attempts to introduce automated computer-to-computer communication remained as isolated niche applications? Let us look upon some of those early adventures and see why their promises could not be kept. EDI was such a promise, but the great idea was compromised by obscure protocol requirements leading to the deadlocking situation of parties not understanding each other. The essential flaw in all the EDI scenarios was that it did not consider the limited (and often close to zero) budgets and IT skills found in smaller firms. But just these small firms would have been the data providers with the highest potential for savings when automating data exchange.

Complicated Protocols Hindered Efficient Communication

CORBA failed due to an obscure binary protocol and the lack of common agreements on the document object model. DCOM succeeded to a certain degree but only due to the sheer market power of Microsoft.

ODBC and JDBC are of limited success. When trying to write an ODBC adapter, it is obvious that a breakthrough is less likely as developers usually bypass the technology with their own simpler approaches. So what do all those early service attempts have in common? They are not simple, not “really” simple; they are awkward. Something that is too complicated to use, simply won’t be used. We from the agile-development faction teach the students of software development not to document their work. Software especially does not need it. A programming language is a language, so it should tell us the whole story without being seconded by another language. Something that needs documentation is not worth it.

Any early interface protocol was loaded by the nearly obsessive attempt of any interface designer to avoid excessive traffic over the networks. That was playing “deer in the headlights” as all sights were focused on getting “compact, light-weight protocols”, while neglecting the ability to make an interface easy to understand on its own.

The outcome was a shallow, long-lasting learning curve that built up a high barrier to using them. As an irony, one needs to realize that the XML-based communication does not really produce more traffic than binary formats, as nearly all network communication uses multiplexed or compressed data transport on the (below IP) layers of the network. So, all those efforts were for chasing a phantom and were, hence, in vain.

Simple, Simple Services…

Here is the essence; we are now in a different situation. SOA is based on a nominal degree of simplicity. This is achieved, to a large part, by two essential active ingredients:

- The hype stemming from Web services introduced new attractive use cases

The Web service hype is most certainly a driving force, as it simply raises a demand for services that can be delivered by an SOA.

- Communication protocols are achieved through XML

This ubiquitous text-based communication protocol is the break-through of Service Oriented Architecture. While still being formal enough for writing a deterministic parser for it, XML is a compromise between human-readable language and a formal description framework.

SOA Relates Proven Communication Concepts of Other Sciences

When trying to understand the true effect and functioning of SOA, it becomes very helpful to look to sciences outside of IT. Electronics, in particular, may give substantial insights. Modern microcontrollers would not perform as reliably as they do today if they were designed in the same antiquated fashion as today's enterprise computing architectures. The secret of a high performance 2Gigahertz CPU is the intensive and consequent use of asynchronous communication strategies whenever two components are required to exchange information.

Asynchronous communication means – simply speaking – exchanging data via queues and events. Even the simplest active electronic processor implements this communication principle. Traditionally the queue is called a "buffer" and the events are referred to as "interrupts". The component that allows information to flow between endpoints is called the bus, the name that we find again in the well-known "Enterprise Service Bus[2]". The benefit that becomes a necessity in high speed data processing is the characteristic of asynchrony. This means that the consuming partner controls the speed of processing, and therefore cannot be flushed with an affluence of data. In an asynchronous scenario, the sending partner can drop its information to the receiver buffer without being blocked when the receiver is too slow to process the data immediately.

We can often observe such problems in SAP RFC communication when a sender transmits data through a BAPI call faster than the update queue of SAP can handle it. This may lead to a situation that the receiving SAP system seems to be hanging, because other dialog steps are slowed down due to a lack of sufficient processor time.

Going back to electronics, we can see that it goes even further when two communicating components are required to persist independently of each other. They try to take the direct galvanic coupling between two components completely away. This is known as "galvanic decoupling", and guarantees that if one component has a fatal malfunction then proper functioning of any other communication partner will not be affected.

Imagine when a high power audio amplifier connects to a loudspeaker system. When the loudspeakers are directly connected to the amplifier, and a current peek exceeds the physical limits of the speaker's inductor, the latter will ultimately burn through. This is circumvented by removing the direct physical link – the Galvanic Link – between the two components, and introducing a mediation component in the form of an opto-coupler. Because the optical component is completely independent of the communication partners, it acts as a damping element and fuses, in case of severe misbehavior of any linked endpoint.

---

[2] We would like to take this opportunity to honor David Chappell – vice president and chief evangelist of Sonic Software - who can claim the distinction of being the first to identify the benefits of transposing the philosophy of a microprocessor bus to a communication scenario of distributed and loosely coupled software components

Strategies to Getting SOA in Place

We have (hopefully) convincingly demonstrated that implementing an SOA is not really a technical challenge. The hardest part in introducing SOA is in circumventing the concerns and blockades within the organization. But be aware that such reluctance does not come from selfish motivation, but is typically the result of poor education or simple ignorance with respect to modern developments. Far too many project managers still live in the belief that their Outlook calendar that miraculously synchronizes with their Blacberry mobile phone is the latest revolution in IT.

Just as it is true when talking to management, we also need to take away the fears from the responsible IT staff by teaching them the benefits and enabling them to understand SOA in a step-by-step introduction (with positive feedback).

Self-Organization Replaces Control

In IT we find ourselves in a state before enlightenment. We cannot but endorse the insight of an unknown source:

“Creative ability and personal responsibility are strongest when the mind is free from supernatural belief, and operates in an atmosphere of freedom and democracy.”

and the American president and philosopher Thomas Jefferson consents in saying:

“I have no fear that the result of our experiment will be that men may be trusted to govern themselves without a master.”

Getting the initial SOA analysis and technical setup is critical. The options are numerous and generally lack the quality required. Getting the corporate governance solution and project management in place is the delicate issue. It is difficult because it encompasses a shift in mindset and strategy on the project manager’s part. The younger developers who have grown up with object orientation and understand the latest and greatest gadgets in the Internet – blogs, wikis, interactive portals, search engines - already have the feeling for the agile and modular approach to development. But teaching management that they need to surrender their control of the SOA project, and rather let the control manifest itself in a self-organizing and democratic society can be very difficult indeed. It is impossible to plan and command a proper SOA, but SOA can be seeded and then it shall grow right by itself.

Looking at the Far Side

"The only way to make sense out of change is to plunge with it, move with it, and join the dance." - Alan Watts

Implementing SOA is less difficult, risky, and challenging than it may appear. In fact, you could implement SOA over time without anybody even knowing about it. How is this possible? In order to sharpen our senses, we shall have a look at how other professions handle change. Of course we need to choose an example that resembles our challenge very closely. We currently find ourselves in a world of inherently isolated computer applications that have learnt (through the power of networks) to exchange data with each other .This is the same as in the Stone Age where every tribe lived in their own kraal or cavern and often did not know about each other's existence. Over time they discovered and learned to communicate with each other, and slowly but surely built market places. Paths between the villages became set in the wild simply because people found out that following an alley that had been cut by a person walking that way before was easier to follow than trudging through the high grasses.

Computer applications evolved very similarly. Having typed texts on one computer, you needed them on another eventually. So we invented first the tennis shoe network[3] and then Laplink, the latter getting better and better over time with RS232 serial cable transfer[4] and Centronics printer cable (even later).

Indeed, we are now in a situation like farms in the outlands where every farmer builds his own roads and installs his own utilities. As more and more farms appear, they will naturally think of sharing some of the installations. This does not need a central organization. In the beginning the neighbors will simply talk with each other and ask whether one party feels fine with having the other party use their investment. This works fine and the initiative comes from the roots. Third-party intervention would only be necessary if there are more parties involved and one party blocks the others by refusing to collaborate.

After a while, a small town will come into being. Traffic will grow and the initial infrastructure might need some changes. Causes for changes can be from simply organizing traffic, extending the throughput capabilities, and installing new services that are in popular demand. That is exactly the same way that SOA works.

Finally SOA will lead into a new level of networked computing, into “Computing Clouds”. Such clouds form when a great number of computers interact with each other and allow a theoretically infinite capacity for computing power on demand. Services will compete for the best results and when a client application temporarily requires a different flavor of the service, it will just send out a request to the cloud.

[3] Store data on diskette, put on your tennis shoes, and jog over to the disparate computer.

[4] Still anybody out there remembers how awkward it was to mould a serial transfer cable?

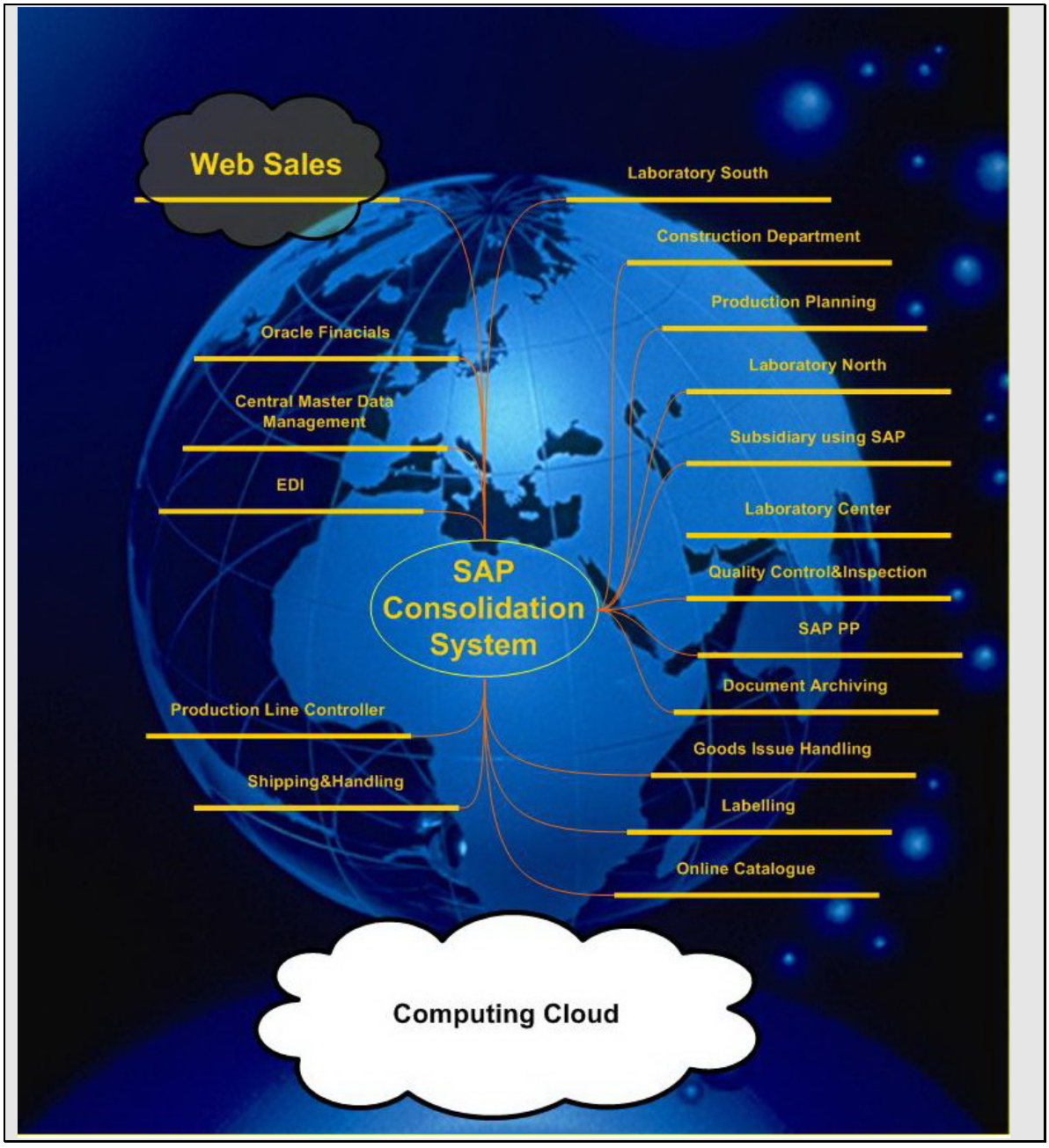

Figure 2: Elastic Computing Cloud with SAP as the Stellar Object

SOA Projects Require a Market-Oriented Governance

We may concede that all the preceding statements are still too high level to allow for a proper proof of the concept in a real project situation. We often do not have a clean academic playground to play on, but rather are often bogged down with a conglomerate of militating departments and numerous applications. These applications often have redundant or

contradicting functionality and incompatible protocols (that are poorly documented and usually highly sensitive with respect to persistent stability in cases of even very minor changes).

The solution to introduce SOA in such a real world IT habitat does not differ from classical project governance. This is well established in many engineering sciences, however, very seldom in IT projects. Introducing an agile, reactive, and self-controlling organization requires principles that are best described with the example of market economics. Instead of setting up an imperative master plan and command actions for every project step, the strategy is based on identifying the key tasks and setting up islands that can provide the required tasks as services. The secret herein is that the services are “well trained” and experienced in delivering results to the target as needed. An imperative scenario does not cope with unexpected situations and normally does not consider that the executing party is not sufficiently experienced in fulfilling a task.

There is another pitfall: the service provider might have the tendency to favor other aspects like reducing department manpower to achieve a better result for the cost center. In the case of many organizations, the fear of making a mistake may override any proactive initiative, resulting in rejecting nearly all service requests.

But this can be easily overcome by putting up a competitive organization. Instead of directing a team or department to cater to a service, every participating player can pick up the request to provide a service and offer a solution it believes will suit the requirement. This is mainly the governance of a market economy that we all in the democratic states are so proud of, and that certainly forms the most stable of all strategies to make a society function in harmony.

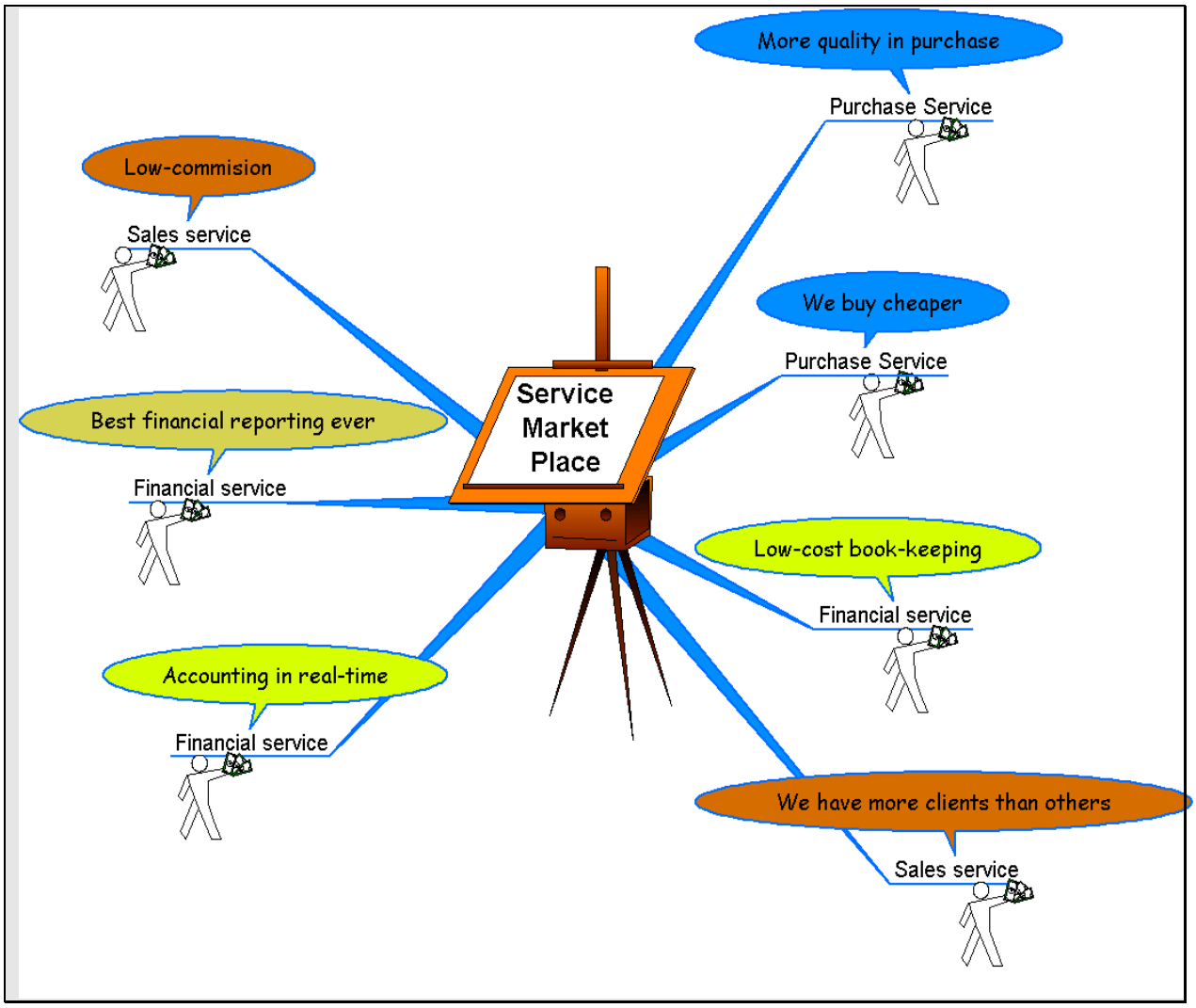

Figure 3: SOA Will Create Competitive Market Places

SOA Implementation

Let us take a look at another example that is closer to engineering. SOA is an architecture, so why not compare it to other architectural approaches? Setting up SOA is like building a new bypass highway as an alternate route for the long-haul traffic. You simply need to build the road and wait. It will tempt the drivers, and over time, more and more will realize the ease of highway driving versus the worn roads through in-town traffic.

SOA does not ask for any strict abidance, but places useful services into the public and encourages (not forces) the society of software applications to make use of them. Although, just like in real societies, one can sponsor the use of the desired services and penalize ignorance.

Certain services will be used and others will not be used. “Networking” between developers and IT personnel will quickly iron out what services are awesome to use and which are not. Once a specific service becomes publicly available in a repository like UDDI (Universal Description, Discovery, and Integration) or similar, it will be subject to public evaluation. Pretty soon society

will “vote” as to whether or not the specific service is doomed. It is up to the service providers themselves to ensure that their portfolios of services are properly marketed and provide useful (if not outstanding functionality) to the applications that use them.

The first step will be to build a common communication platform that encourages replacement of peer-to-peer communication. Installing commonly requested services like monitoring message flow, central business objects, standard reporting formats (Business Activity Monitoring), etc., will make SOA more and more popular. That is analogous to what an operating system offers. No witty developer will care anymore about drawing his or her own graphical windows on a screen but will use the GUI APIs like that of windows GDI or Java Swing. They will utilize printers that use a common printer API to communicate with applications instead of requiring a separate driver for every program. These are simply a few of many possible examples. SOA will put the same success story on another and higher level.

SOA Governance Means “Evolution Instead of Revolution”

When you start an SOA program, you will soon be in the same situation as a chess player; there are few pieces on the board, and each has a high degree of individuality. The game itself is controlled by very few rules, but those are unchangeable during the game. There is no reasonable way to predict how the game will evolve after only a few plays, but with a right and agile strategy, a good player can control the coming moves.

The overriding rule is that we depreciate revolution and promote evolution. We won’t say “All-in-One was yesterday” and “SOA is tomorrow”. Most IT projects have a scent of revolution. That is the result of selling every trivial change in IT as a striking break-through. We need heroes! The hero gets all the budget dollars she or he wants! The prospect of revolution may bring enthusiasm to some, since being patient and passionate in improving things step-by-step is not always sexy. But heroes die young, and what we need is to bring SOA in place by evolution in that we change and improve components without causing turbulences by shattering the whole current IT infrastructure.

We will endorse the fact that the current IT infrastructure is a stable and sophisticated technical environment that does not allow big-bang replacement just for the sake of the benefits of a new technology. Hence there won’t be an “SOA project”; rather, there will be many projects in parallel. These projects won’t be called “SOA projects”. In fact, they might not even know that they are based on it. The only aim of SOA governance is to communicate the spirit of SOA to everybody, to make sure that new projects work toward this ultimate goal. This is not as difficult as it sounds, since with SOA we should be able to demonstrate the benefits (as compared to “old fashioned technology”).

In practice many projects within the SOA program will rise and then disappear again. There will be no way to predict or plan them in the beginning. In fact, the more we try to plan individual projects within the SOA program, the more we shall disconnect from our SOA vision. An SOA program is a challenge that deals with both aleatoric and epistemological uncertainties. Such environments are safely handled by setting very few, very simple rules. The secret in managing distributed programs like SOA lies not in detailed planning but in making sure that everybody in the game knows the rules and the consequences very well. It is known as the “Ten-Commandments-Principle”, stating that only the rules that are unanimously accepted by the whole community are worth being noted while everything else is anarchy that pushes people away from the desired focus.

Confluence Replaces the Waterfall

It will be a hard time for IT project and program managers. They are used to controlling projects through detailed planning and concentrate on one component after the other. For these classical

waterfall strategies, this kind of parallelism is a compromise that is performed solely to allow for quicker completion. They don't like it because the more parallel activities that exist, the more difficult the project's control.

SOA projects will soon establish a high degree of parallelism and develop many uncontrolled activities. The more parallel actions that are pursued, the higher the success and ROI (of an SOA) will be. All SOA projects will abide by two major rules:

- Every function should be made accessible for others through a well defined interface.
- Every application should make use of existing services, while enhancing them instead of rebuilding them.

All solutions created in the mind of SOA will eventually flow into a common pool of services, and project results will flow together after time to form a completely new organism. This form of program management is called *Confluence*. The analogy to water should help to help you visualize the process; the waterfall in the river is seen as being uncontrollable and wild, but the confluence is where the river is tamed and navigable.

Patterns Will Replace Roadmaps

So, what is now the roadmap to get a perfect Service Oriented Architecture? The active ingredients and the action steps are not very different from other complex projects.

In the very beginning, we need to publish the idea of SOA as the desired target within the enterprise; the "SOA Program" or the "SOA Charter".

We are tempted to say "In the next step", but that is exactly the thinking we should avoid in the future. There will not be a linear roadmap. SOA projects won't be linear; they will have a high degree of parallelism with iteration and recursion in place. Roadmap is a bad expression for an SOA program. What we actually will do is implement abstract patterns and flesh them out with variations of actual use, to become reusable components.

In order to have a common marketplace for communication and message exchange, we need to establish an Enterprise Service Bus in a very early state. The solution we decide on in the beginning needs not necessarily be the final one. There can be many ESBs in parallel. One of the ESBs is the arbitration hub, so it is not necessary to fall into a lengthy evaluation process for an ESB. Take what appears to be the easiest path to learn for now, or what you already have in place. Changing the ESB later is like changing a car—as long as the defined business processes (aka workflows) adhere to some standards like WSDL and BPEL, all should be OK. Choosing some open source ESB will bring you to the safe side with respect to standards compliance, as they don't have the tendency to violate standards for their own commercial interests as do the giants like SAP or Oracle[5].

What is even more important is having good tools in place. Most nowadays SOA offerings consist merely of a translation and workflow processing engine, with poor tools to build, test, and simulate the processes. Having a look at what the open source community delivers in this respect is really worth the effort. Some of the big ones understood the importance of front-end and development aids. IBM for instance, made a big step forward by adding a fast Business Process

[5] Just as a side note to avoid misunderstandings: Falling in the category of the giants, IBM won't fit that pattern as they committed themselves to support the internationally accepted W3C standards and proactively support the open source community.

Designer to their ESB offering, that not only sports many design features, but also allows process simulation based on stochastic models, as shown in Figure 4. Microsoft BIZTALK has similar visual tools ready, while SAP's Exchange Infrastructure XI will have to invest some overtime to keep pace with the leaders.

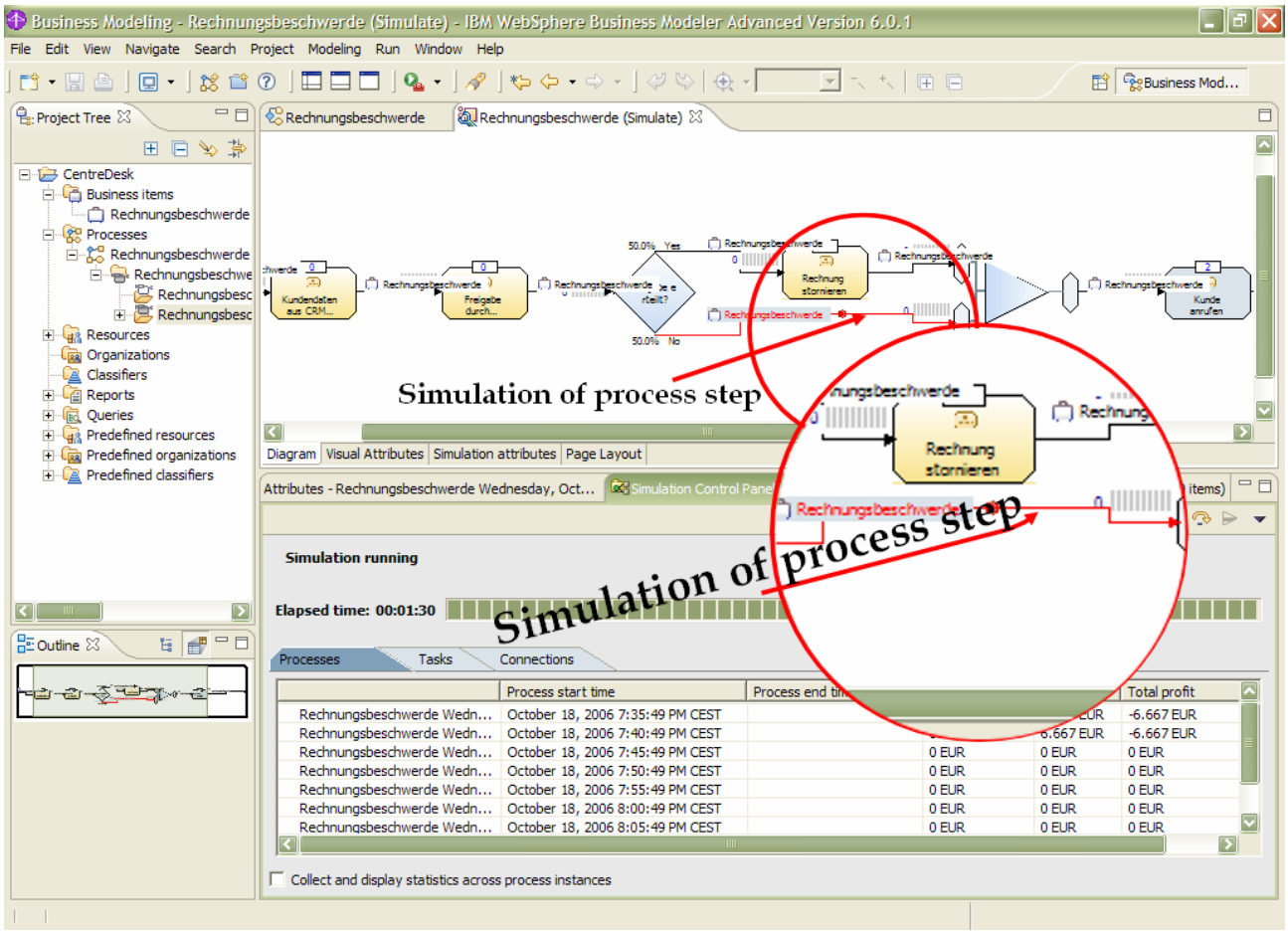

Figure 4: IBM's New BPM Designer Allows Simulation of Workflow Designs

Here is a small synopsis of the tasks required to get SOA in place (with examples for SAP and civil engineering) just to sharpen the imagination

Task	Examples for SAP	Examples in Civil Engineering
Teach the basics of SOA and dream the new world	SAP will remain the core enterprise solution but PP will be replaced by a dedicated planning tool and packaging by a specialized service enabled software	Draw a plan for new city highway and new towns
Introduce an Enterprise Service Bus	We install one XI as a gateway proxy for every SAP and an open source ESB for the time being as a central exchange hub	We build some provisionary roads to drive on during construction periods
Identify components that can be exposed as services	All SAP BAPIs are already service ready and can be consumed as such; in addition all workflow enabled SAP transactions will receive a simple event broadcaster to permanently connect them to an ESB	Buy one tractor and one truck to be used for many construction sites
Make components expose their services	All workflow enabled SAP transactions will receive a simple event broadcaster to permanently connect them to an ESB	Create a central rental location for the building machines
Create competitive services that improve the existing ones	A good place to look for are the SAP user-exits; whether they can be coded as a central service for everybody	Build a bypass for a local road and see how it is accepted over time
Replace individual software components through service usage	Address verification, packaging and handling, product configuration, and central master data management are good examples for it	When your old tractor breaks, don't buy a new one but rent it from the on-demand pool

Figure 5: Creating Inroads with SOA in SAP VS Civil Engineering Road Construction

Conclusions

The Service Oriented Architecture will be the driving technology topic for the coming decade. It will provide a sound and reliable distributed operating system to allow effortless computer-to-computer interaction within enterprises as well as across enterprise limits. Enterprise software will abandon an all-in-one and centrally hosted paradigm as it finds replaceable and reusable components that follow simple and practical interfacing protocol standards. On such grounds, computing will find itself in a market place of competing service. No longer will the central

business application specify requirements for components, but, rather, will select the most suitable existing service component out of rich and abundant public offerings.

Building the technology platform for Service Oriented Architecture is an easy task and consists of the installation of an arbitrary WSDL and BPEL compliant Enterprise Service Bus solution (offered by hundreds of vendors). From there on, you need to gradually attach the existing IT solutions to the ESB. Simultaneously, critical applications should be reviewed to determine if they can take advantage of central service components instead of their "island" solutions.

The tough part in SOA is to change the mindset and remove the blocking factors in the brains of traditional IT. The secret in SOA governance lies in defining an SOA program that favors evolutionary changes instead of big-bang projects.

While classical projects are linear with granular tasks set out from the onset, SOA favors many parallel activities with very few central planning elements. For program management, it means that confluence and concert will replace waterfall methodology. Project steering will be abolished in favor of project arbitration. In order to allow control over the shape of the future, an SOA program needs to be based on a very small, but rigid and well-balanced set of commandments. It is the spirit of a civil society that will drive an SOA vision and will disband old illusions of central control over IT. Softly, gently, SOA will unfurl its splendor, once this spirit is free.

What Next?

In the sequels to this series on SOA Governance, we will describe the secret of the "SOA concerto" and demonstrate the core technologies from SAP to Open Source. We will present some examples of how you can make your R/3 SOA ready. We will also see how our fictional "H2O Company" has implemented their service and avoided common pitfalls and mistakes.

About the Authors

Axel Angeli, logosworld.com. *Axel is a senior SAP and EAI advisor and principal of logosworld.com, a German-based enterprise specializing in coaching SAP and EAI project teams and advising IT management on implementation issues. Axel has been in the IT business since 1984, and throughout his career, he has always worked with cutting edge technologies. Axel's SAP experience stems from the good old R/2 days, and he is an expert on SAP's NetWeaver technology and any kind of ABAP development. A speaker of several languages, Axel specializes in coaching and leading large multi-national teams on complex projects with heterogeneous platforms and international rollouts. Known for his intensive and successful trouble-shooting experience, Axel has been nicknamed by his colleagues as the "Red Adair" of SAP projects. He is the author of the best-selling tutorial "The SAP R/3 Guide to EDI, IDocs, ALE and Interfaces."*

Lynton Grice, NetWeaverGuru.com *is an application developer contracted to one of the big automotive companies in South Africa. He has a passion for programming and new technologies and takes care of SAP Portal development as well as other SAP EAI requirements. Coming from a typical Java school he is now just as familiar with ABAP.*

Back to the Future with SOA: The Importance of Governance