WHITE PAPER

SOLVING THE INTEGRATION PROBLEM –

KNOWLEDGEONE (K1) VERSUS SERVICE ORIENTATED ARCHITECTURE (SOA) DEVELOPMENTS

Solving the Integration Problem

Knowledgeone (K1) Versus Service Oriented Architecture (SOA) Developments

I think it was Gartner (www.gartner.com) that first used the term "the Integration Problem" in 1996 when describing how corporations struggle to manage and integrate twenty to thirty disparate application systems in order to run their businesses.

We first looked at this problem in 1993 and my first paper on an "It Does Everything Application" (IDEA) was published in Image & Data Magazine in March 1994, (on our website at: http://www.knowledgeonecorp.com/news/pdfs/It%20Does%20Everything%20Application.pdf). I also presented a paper to Edith Cowan University in 1994 on "Building a Unified System".

So, the problem isn't new and as it has been a major issue for as far back as I can remember (and I began working on early mainframes in 1964).

In essence, every organization would like to have a single application system that solves all of their application needs. This would be from a single vendor, with a uniform user interface and have single instances of all major tables including people, entities, etc thus totally avoiding any dangerous duplication of data. It would enforce the use of common code classes for consistency and data integrity. There would be just a single application for the IT gurus to install and maintain and the CIO's life would be a piece of cake instead of an ongoing nightmare.

There would only be one type of technology to manage, one database to backup and protect and one user interface for everyone to learn no matter what application they were running. Training and retraining costs would be minimal and productivity levels would rise across the enterprise.

The reality is that most major corporations and government agencies still struggle with 20 or 30 or more disparate applications systems from different vendors, using different technologies, with completely different user interfaces and different copies of core data.

In order to minimize the dangerous duplication of data, IT specialists try their best to 'integrate' these 30 or so disparate applications and try to make them share and exchange data. They are however constantly frustrated because of different technologies, operating and database system dependencies, unstable APIs and a plethora of new releases each of which has the potential to push them back to step one.

This is what the industry means when it talks about the "Integration Problem". By all accounts, it costs the industry billions of dollars every year.

It was also, to my knowledge, Gartner that first popularized the SOA approach in 1996, see: http://www.gartner.com/DisplayDocument?doc_cd=114358

At this stage we should probably pause and expose a few industry definitions of SOA.

What is SOA?

Wikipedia

Service-oriented architecture (SOA [pronounced "sō-uh" or "es-ō-ā"]) expresses a perspective of software architecture that defines the use of loosely coupled software services to support the requirements of the business processes and software users. Resources on a network[1] in an SOA environment are made available as independent services that can be accessed without knowledge of their underlying platform implementation.[1]

Microsoft

SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners.

The goal for a SOA is a world wide mesh of collaborating services, which are published and available for invocation on the Service Bus. Adopting SOA is essential to deliver the business agility and IT flexibility promised by Web Services. These benefits are delivered not by just viewing service architecture from a technology perspective and the adoption of Web Service protocols, but require the creation of a Service Oriented Environment that is based on the following key principals:

Service is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form.

SOA is not just architecture of services seen from a technology perspective, but the policies, practices, and frameworks by which we ensure the right services are provided and consumed.

With SOA it is critical to implement processes that ensure that there are at least two different and separate processes—for provider and consumer.

Rather than leaving developers to discover individual services and put them into context, the Business Service Bus is instead their starting point that guides them to a coherent set that has been assembled for their domain.

IBM

Service Oriented Architecture (SOA) is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services. SOA helps users build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes.

Kodali

Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature; i.e., the service interface is independent of the implementation. Application developers or system integrators can build applications by composing one or more services without knowing the services' underlying implementations. For example, a service can be implemented either in .Net or J2EE, and the application consuming the service can be on a different platform or language.

Gartner

Essentially, SOA is a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls. SOA would be better-named "interface-oriented architecture." SOA is a relationship of services and service consumers, both software modules large enough to represent a complete business function. Services are software modules that are accessed by name via an interface, typically in a request-reply mode. Service consumers are software that embeds a service interface proxy (the client representation of the interface).

So, after reading the above and any other number of examples courtesy of Google we become to realize that SOA is a loosely coupled architecture that SHOULD BE based on business requirements NOT any specific technology. SOA is NOT about Web Services or Java or .NET or any specific technology. SOA is a systems approach to solving the integration problem by building application systems that utilize a common set of business 'services' each of which has been designed and built independent of its various implementations. That is, each 'application' calls on a set of common services.

Whereas a SOA implementation does not mandate Web Services, Web Services is most often proposed as the 'glue' to link all of these services and applications together.

One more definition, just in case any of you are unfamiliar with Web Services or SOAP (Simple Object Access Protocol).

What is Web Services/SOAP?

Wikipedia

SOAP (originally **Simple Object Access Protocol**) is a protocol for exchanging XML-based messages over computer network, normally using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework that more abstract layers can build on. The original acronym was dropped with Version 1.2 of the standard, which became a W3C Recommendation on June 24, 2003, as it was considered to be misleading.

There are several different types of messaging patterns in SOAP, but by far the most common is the Remote Procedure Call (RPC) pattern, in which one network node (the client) sends a request message to another node (the server), and the server immediately sends a response message to the client. SOAP is the successor of XML-RPC, though it borrows its transport and interaction neutrality and the envelope/header/body from elsewhere, probably from WDDX.

In layman's terms, SOAP/Web Services is a way of sending transactions from a client to a server using HTTP or HTTPS (Secure Socket Layer with encryption), the same protocol the Internet uses.

The big advantage of SOAP is that you don't need a LAN or WAN in place to have a remote user communicate with your application; you can utilize either the Internet or (if you have one) your Intranet. Oh, and one final point on SOAP, it uses XML as its 'language'; so, we may say that SOAP is based on XML.

So, let's summarize by saying that SOA is an approach to designing and building business application systems that envision using a collection of 'common' routines or 'services' to service a number of different business applications. Web Services or SOAP comes into the equation simply as the most appropriate technology to tie all these services and applications together. If you will, SOAP is the carrier of all transactions.

What is SOA Not?

SOA is not an application system. SOA is not a single application able to solve multiple business problems. SOA is not a product. SOA is not "Off-The-Shelf", ready to solve your integration problem.

SOA is a methodology, a proposed way to solve the integration problem by utilizing a particular solution architecture based on the concept of loosely coupled services.

SOA requires a high level of design expertise. SOA requires a high level of programming expertise. SOA requires an extensive testing regime and tightly managed roll-out. A SOA solution will take many, many man years to design, program, test and implement. A SOA solution will require an enormous degree of cooperation and agreement between the various departments of a corporation. Any SOA implementation is a major expense and involves a major risk of failure because despite all the hype (particularly from the major players) SOAP probably requires the kinds of expertise you do not have and the size of budget you were hoping you didn't have to spend and an implementation timeframe you were hoping you didn't have to suffer.

Is SOA Real?

It is if you are smart enough to make it work but do not underestimate the difficulty factors involved. Implementing an enterprise-wide SOA system is a hugely complex, time consuming and expensive undertaking.

Is SOA a Possible Solution to the Integration Problem?

Yes, if you are big enough and rich enough and smart enough and tough enough to see it through. And yes if your vendor or systems integrator or solutions partner can actually walk-the-walk as well as talk-the-talk. All the major players are jumping on the SOA bandwagon so just make sure there is plenty of expertise to back up the marketing rhetoric. Keep reminding yourself that SOA is not an 'Off-The-Shelf" solution, it is a methodology.

Now before we compare SOA to Knowledgeone ^{K1} let's first tell you what Knowledgeone ^{K1} is.

Knowledgeone ^{K1} (K1)

K1 is a generic application solution (we coined the acronym GAS) based on the Microsoft .NET model. It is a single, multilingual application that is able to run any number of disparate applications concurrently. It uses a single, common user interface (UI) for every application. It uses a single database (with no duplication of data) for all applications. All user functions are 'thin-client', meaning that all user/client functions run in the browser (IE6 or IE7) and communicate with the K1 server(s) via HTTP/HTTPS.

K1 comes standard with all the tools you will need to modify any of its 'out-of-the-box-applications' (e.g., asset management, help desk, complaints management, HR management, CRM, records management, document management, workflow, imaging) or create brand new applications. It also comes with all of the tools you will need to import or export data or integrate to any other application.

K1 was designed from the outset to solve the integration problem by providing a single application that can happily run any number of different applications concurrently.
And, K1 also provides optional plug-ins for knowledge management, fully automated (no end-user involvement required) email management and archiving and fully automated electronic document management.

Unlike SOA, K1 is a working multiple application solution that you can install and start using the next day.

Is Knowledgeone [K1] a Complete Solution to Every Organization's Integration Problem?

No, of course it isn't. There will always be business needs that are better handled by a specialist application that is designed, written and updated by industry specialists. There will always be existing applications that work fine and don't present any problems and where the old adage of, "If it ain't broke, don't fix it!" should always apply, (for wise men anyway).

We see K1 as an ideal and extremely cost effective (and cost saving) solution for old mainframe applications. We see K1 as an ideal solution for any classic 'information management' application like help desk, complaints management, HR management, asset management, CRM, library management, email management, etc. We also see it as an ideal solution for any records management, document management, imaging or workflow requirement. All of the above applications are basically 'out-of-the-box' and will be an 85% plus fit for most organizations.

But, unless you are prepared to make substantial changes to K1 (and we do provide all of the tools for you to change any aspect of K1 including the data model and any process) then you shouldn't consider K1 as a replacement for your materials handling system or payroll system or accounting system or inventory management system, especially if they are working fine.

So, we are proposing K1 as a partial solution to the integration problem but one that is low cost, easy to install, easy to modify, easy to configure and easy and fast to roll out. It is also, most importantly, a low risk solution because we have done all of the hard work building and testing a robust, .NET, high-throughput, extensible, thin-client transaction system that you or your systems integrator would have to do if you were to attempt the SOA approach. We have done all the hard yards so you don't have to. We have spent many, many man years designing, programming, prototyping, testing and debugging so you don't have to. We also allow you to enjoy the benefits of a new system now, not in 3 or more years when the SOA development is finally debugged and signed off.

Is Knowledgeone [K1] Built on SOA Architecture?

The answer is yes and no. The first specification for K1 pre-dates SOA architecture but K1 architecture is remarkably similar to SOA architecture with independent 'services' (we call them common classes) being used by multiple applications (we call them personalities) and the use of Web Services to link all of our .NET 'smart clients' to the K1 relational database and common classes. I guess we could say "great minds think alike". Basically, it is just common sense and good design, the SOA approach (under different names) has in fact been around since the mainframe days. In effect, we are all just re-inventing a tried and proven methodology using the latest technologies and tools.

Is Knowledgeone [K1] a Competitor to the SOA Approach?

Again, the answer is both yes and no depending upon your needs and preferences. It depends upon the problems you are trying to solve and your current hot buttons or most pressing needs. For many applications K1 is by far the better solution because it is lower cost, lower risk and can provide benefits now, not in 3 or 4 years time.

If I was a CIO in a major corporation (as I used to be before starting K1Corp) I would be sensibly looking at both K1 and SOA. Think "horses for courses", not ideology or dogma. Don't let any particular technology drive your business, be smart enough to select the technologies that are most appropriate for your business needs.

K1 can realistically replace 5, 10 or more of your current core business applications with very little effort, very little cost and very little risk. It would be insane not to consider a generic application

solution like K1 as part of a collection of solutions to the integration problem. If nothing else, we give you a proven way to reduce the size of your integration problem by at least fifty-percent.

Best Regards,

Frank McKenna, CEO the Knowledgeone Corporation

For more information on K1 please look up the following links:

http://www.knowledgeonecorp.com/products/knowledgeone.htm
http://www.knowledgeonecorp.com/products/gem.htm
http://www.knowledgeonecorp.com/products/capture.htm
http://www.knowledgeonecorp.com/products/Tacit.htm
http://www.knowledgeonecorp.com/products/MiniAPI.htm
http://www.knowledgeonecorp.com/products/button.htm
http://www.knowledgeonecorp.com/products/scan.htm
http://www.knowledgeonecorp.com/products/xchange.htm
http://www.knowledgeonecorp.com/products/drm.htm