Service Oriented Architecture: The Future of Information Technology

Ezekwe Chinwe Genevra[1], Okwu Patrick[2], Onuodu Friday Eleonu[3]

[1-2] Electronics Development Institute, Awka National Agency for Science and Engineering Infrastructure (NASENI), Federal Ministry of Science and Technology,
[3] Department of Computer Science, University of Port Harcourt, NIGERIA.

[1] norakingchi@yahoo.com,[2] okwupi@gmail.com, [3] gonuodu@gmail.com

ABSTRACT

Service-Oriented Architecture (SOA) is emerging as an effective solution to deal with rapid changes in the business environment and to handle fast-paced quality of its products prior to implementation. However, literature and industry has yet to explore the techniques for evaluating design quality of SOA artifacts. To address this need, this paper presents a Microsoft hierarchical quality assessment model for early assessment of SOA system quality. Microsoft advocates a "middle out" approach which combines both top-down and bottom-up methodologies. The SOA architectural models are fractal. This means that a service can be used to Expose IT assets (such as a Line of Business system), be Composed into workflows or Business Processes (each of which may also be exposed as a service) and be Consumed by end users, systems or other services. The key result is independent services with defined interfaces that can be called to perform their tasks in a standard way, without a service having foreknowledge of the calling application.

Keywords: Service Oriented Architecture (SOA), Web Services Description Language (WSDL), SOA protocol

INTRODUCTION

Service-oriented architecture (SOA) is a software design and software architecture design pattern based on discrete pieces of software providing application functionality as services to other applications zdnet.com (2009). This is known as Service-orientation. It is independent of any vendor, product or technology.

A service is a self-contained unit of functionality, such as retrieving an online bank statement. Services can be combined by other software applications to provide the complete functionality of a large software application. SOA makes it easy for computers connected over a network to cooperate. Every computer can run an arbitrary number of services, and each service is built in a way that ensures that the service can exchange information with any other service in the network without human interaction and without the need to make changes to the underlying program itself.

LITERATURE REVIEW

Service Orientation (SO) is the natural evolution of current development models. The, 80s saw object-oriented models; then came the component-based development model in the 90s; and now we have service orientation (SO). Service orientation retains the benefits of component-based development (self-description, encapsulation, dynamic discovery and loading), but there is a shift in paradigm from remotely invoking methods on objects, to one of passing messages between services. Schemas describe not only the structure of messages, but also behavioral contracts to define acceptable message exchange patterns and policies to define service semantics. This promotes interoperability, and thus provides adaptability

ISSN: 2223-9944, e ISSN: 2223-9553

benefits, as messages can be sent from one service to another without consideration of how the service handling those messages has been implemented zdnet.com (2009).

Services are unassociated, loosely coupled units of functionality that are self-contained. Each service implements one action, such as submitting an online application for an account, retrieving an online bank statement or modifying an online booking or airline ticket order. Within a SOA, services use defined protocols that describe how services pass and parse messages using description metadata, which in sufficient details describes not only the characteristics of these services, but also the data that drives them. Programmers have made extensive use of XML in SOA to structure data that they wrap in a nearly exhaustive description-container. Analogously, the Web Services Description Language (WSDL) typically describes the services themselves, while the SOAP protocol describes the communications protocols. SOA depends on data and services that are described by metadata that should meet the following two criteria:

1) The metadata should be provided in a form that software systems can use to configure dynamically by discovery and incorporation of defined services, and also to maintain coherence and integrity. For example, metadata could be used by other applications, like a catalogue, to perform auto discovery of services without modifying the functional contract of a service.
2) The metadata should be provided in a form that system designers can understand and manage with a reasonable expenditure of cost and effort.

Benefits

The main benefit of SOA is to allow simultaneous use and easy mutual data exchange between programs of different vendors without additional programming or making changes to the services. Waldo,(2002). These services are also reusable, resulting in lower development and maintenance costs and providing more value once the service is developed and tested. Having reusable services readily available also results in quicker time to market.

Principles

There are no industry standards relating to the exact composition of a service-oriented architecture, although many industry sources have published their own principles. Some of the principles published include the following Waldo, (2002). :

a. ***Standardized service contract*:** Services adhere to a communications agreement, as defined collectively by one or more service-description documents.
b. ***Service loose coupling*:** Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
c. ***Service abstraction*:** Beyond descriptions in the service contract, services hide logic from the outside world.
d. ***Service reusability*:** Logic is divided into services with the intention of promoting reuse.
e. ***Service autonomy*:** Services have control over the logic they encapsulate.
f. ***Service statelessness*:** Services minimize resource consumption by deferring the management of state information when necessary
g. ***Service discoverability*:** Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted.

ISSN: 2223-9944, e ISSN: 2223-9553

h. ***Service composability*:** Services are effective composition participants, regardless of the size and complexity of the composition.

i. ***Service granularity*:** A design consideration to provide optimal scope and right granular level of the business functionality in a service operation.

j. ***Service normalization*:** Services are decomposed and/or consolidated to a level of normal form to minimize redundancy. In some cases, services are denormalized for specific purposes, such as performance optimization, access, and aggregation.[13]

k. ***Service optimization*:** All else equal, high-quality services are generally preferable to low-quality ones.

l. ***Service relevance*:** Functionality is presented at a granularity recognized by the user as a meaningful service.

m. ***Service encapsulation***: Many services are consolidated for use under the SOA. Often such services were not planned to be under SOA.

n. ***Service location transparency***: This refers to the ability of a service consumer to invoke a service regardless of its actual location in the network. This also recognizes the discoverability property (one of the core principle of SOA) and the right of a consumer to access the service. Often, the idea of service virtualization also relates to location transparency. This is where the consumer simply calls a logical service while a suitable SOA-enabling runtime infrastructure component, commonly a service bus, maps this logical service call to a physical service.

Service Composition Architecture

One of the core characteristics of services developed using service-orientation design paradigm is that they are composition-centric. Schroth et al (2007) Services with this characteristic can potentially address novel requirements by recomposing the same services in different configurations. Service composition architecture is itself a composition of the individual architectures of the participating services. In the light of the Service Abstraction principle, this type of architecture only documents the service contract and any published service-level agreement (SLA); internal details of each service are not included.

If a service composition is a part of another (parent) composition, the parent composition can also be referenced in the child service composition. The design of service composition also includes any alternate paths, such as error conditions, which may introduce new services into the current service composition.

Service inventory architecture

A service inventory is composed of services that automate business processes. It is important to account for the combined processing requirements of all services within the service inventory. Documenting the requirements of services, independently from the business processes that they automate, helps identify processing bottlenecks. The service inventory architecture is documented from the service inventory blueprint, so that service candidates can be redesigned before their implementation.

Service-oriented enterprise architecture

This umbrella architecture incorporates service, composition, and inventory architectures, plus any enterprise-wide technological resources accessed by these architectures e.g. an (entity relation process)ERP system. This can be further supplemented by including

ISSN: 2223-9944, e ISSN: 2223-9553

enterprise-wide standards that apply to the aforementioned architecture types. Any segments of the enterprise that are not service-oriented can also be documented in order to consider transformation requirements if a service needs to communicate with the business processes automated by such segments. SOA's main goal is to deliver agility to business

Web services approach

Web services can implement a service-oriented architecture. They make functional building-blocks accessible over standard Internet protocols independent of platforms and programming languages. These services can represent either new applications or just wrappers around existing legacy systems to make them network-enabled.

Each SOA building block can play one or both of two roles:

1) *Service provider*: The service provider creates a web service and possibly publishes its interface and access information to the service registry. Each provider must decide which services to expose, how to make trade-offs between security and easy availability, how to price the services, or (if no charges apply) how/whether to exploit them for other value. The provider also has to decide what category the service should be listed in for a given broker service and what sort of trading partner agreements are required to use the service.
2) *Service consumer*: The service consumer or web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its web services. Whichever service the service-consumers need, they have to take it into the brokers, bind it with respective service and then use it. They can access multiple services if the service provides multiple services.

SIGNIFICANT OF SOA

Service Orientated Architecture is important to several stakeholders Schroth et al (2007):

a. To developers and solution architects, service orientation is a means for creating dynamic, collaborative applications. By supporting run-time selection of capability providers, service orientation allows applications to be sensitive to the content and context of a specific business process, and to gracefully incorporate new capability providers over time.

b. To the IT manager, service orientation is a means for effectively integrating the diverse systems typical of modern enterprise data centers. By providing a model for aggregating the information and business logic of multiple systems into a single interface, service orientation allows diverse and redundant systems to be addressed through a common, coherent set of interfaces.

c. To the CIO, service orientation is a means for protecting existing IT investments without inhibiting the deployment of new capabilities. By encapsulating a business application behind capability-based interfaces, the service model allows controlled access to mission-critical applications, and creates the opportunity for continuous improvement of the implementation behind that interface. Service orientation protects investments from the swirl of change.

d. To the business analyst, service orientation is a means of bringing information technology investments more in line with business strategy. By mapping employees, external capability providers and automation systems into a single model, the analyst can better understand the cost tradeoffs associated with investments in people, systems and sourcing.

ISSN: 2223-9944, e ISSN: 2223-9553

To Microsoft, service orientation is a crucial prerequisite to creating applications that leverage the network to link the actors and systems that drive business processes. This application model transcends any single device, crossing boundaries respectfully, and rejecting the restrictions of synchronicity. SOA-enabled solutions pull together a constellation of services and devices to more effectively meet your business challenges than the disconnected applications of the past.

THE TECHNOLOGY

While a well planned and executed SOA undertaking can help organizations realize greater responsiveness in a changing marketplace and technology, not all service-oriented efforts have been successful. SOA projects may experience limited success when they are driven from the bottom up by developers unfamiliar with the strategic needs of the organization Hoyer et al(2008). Building SOA for the sake of SOA without reference to the business context is a project without organizing principles and guidance. The result is a chaotic implementation that has no business relevance. On the other hand, taking a top-down mega-approach to SOA requires such enormous time investments that by the time the project is complete, the solution no longer maps to business needs. (This of course is one of the problems SOA is supposed to solve!)

THE METHODOLOGY

By contrast, Microsoft advocates a "middle out" approach which combines both top-down and bottom-up methodologies. In this approach, SOA efforts are driven by strategic vision and business need, and are met through incremental, iterative SOA projects that are designed to deliver on business goals one business need at a time. Microsoft has been using this technique to assist customers with their SOA initiatives since the .NET framework was first released in 1999.

The concept of SOA can be viewed from several possible perspectives. While no single perspective or set of perspectives represents a definitive view of a SOA, from a holistic view these perspectives assist in understanding the underlying architectural requirements. Microsoft believes that there are three abstract capability layers exposed within a SOA. They are: a) Expose, b) Compose, c) Consume.

An illustration of these categories and their relationships to one another appears below:

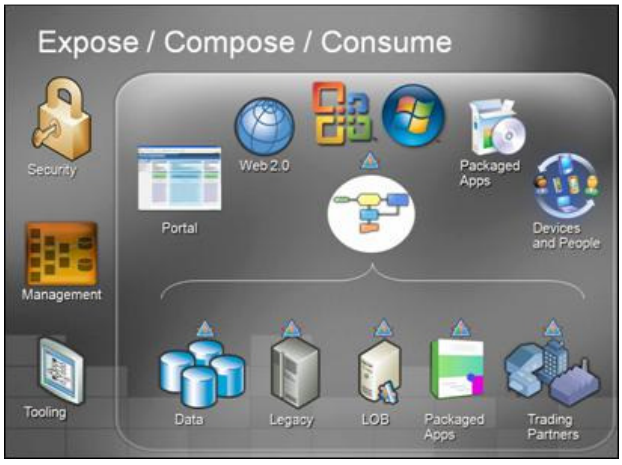

Figure 1. An Abstract Reference Model for SOA Microsoft White Paper, (2006).

ISSN: 2223-9944, e ISSN: 2223-9553

Expose

Expose focuses on how existing IT investments are exposed as a set of broad, standards-based services, enabling these investments to be available to a broader set of consumers. Existing investments are likely to be based upon a set of heterogeneous platforms and vendors. If these applications are unable to natively support Web services a set of application or protocol-specific set of adapters may be required. Service creation can be fine grained (a single service that maps on to a single business process), or coarse grained (multiple services come together to perform a related set of business functions). Expose is also concerned with how the services are implemented. The functionality of underlying IT resources can be made available natively if they already speak Web services, or can be made available as Web services though use of an adapter.

Compose

Once services are created, they can be combined into more complex services, applications or cross-functional business processes. Because services are existing independently of one another they can be combined (or "composed") and reused with maximum flexibility. As business processes evolve, business rules and practices can be adjusted without constraint from the limitations of the underlying applications. Services compositions enable new cross-functional processes to emerge, allowing the enterprise to adopt new business processes, tune processes for greater efficiency, or improve service levels for customers and partners. A *Service Integration Architecture* describes a set of capabilities for composing services and other components into larger constructs such as business processes. Composing services requires some sort of workflow or orchestration mechanism. Microsoft provides these capabilities via BizTalk Server 2006 (BTS) or Windows Workflow Foundation (WF). While BTS and WF may appear to serve similar needs, they are actually quite different. WF and BTS are *complementary* technologies designed to serve two very different needs: BTS is a licensed product designed to implement workflow ("orchestrations") across disparate applications and platforms.

WF is a developer framework designed to expose workflow capabilities within your application. There are no fees or licensing restrictions associated with using or deploying WF.

Consume

When a new application or business process has been created that functionality must be made available for access (consumption) by IT systems, other services or by end-users. Consumption focuses on delivering new applications that enable increased productivity and enhanced insight into business performance. Users may consume "composed" services through a broad number of outlets including web portals, rich clients, Office business applications (OBA), and mobile devices. "Composed" services can be used to rapidly roll out applications that result in new business capabilities or improved productivity. These application roll-outs can be used to measure the return on investment (ROI) in an SOA. A *Service Oriented Application Architecture* in section VI describes how "composed services" are made available for consumption through as business processes, new services or new end-user applications. This concept is sometimes referred to as *Composite Applications* since it implies service consumption by end-user applications. Microsoft's Office Business Applications (OBAs) support the Composite Application notion of transactional systems while expanding the scope of user interaction via the familiar Office environment.

ISSN: 2223-9944, e ISSN: 2223-9553

While the architectures described within Expose / Compose / Consume may be interdependent, they are designed to be loosely coupled. This enables services to be managed, versioned and configured independently of how they are exposed,

IMPLEMENTATION

Implementations can use one or more of these protocols and models, for example, might use a file-system mechanism to communicate data conforming to a defined interface specification between processes conforming to the SOA concept Christopher (2005). The key is independent services with defined interfaces that can be called to perform their tasks in a standard way, without a service having foreknowledge of the calling application, and without the application having or needing knowledge of how the service actually performs its tasks.

Service-oriented modeling is a SOA framework that identifies the various disciplines that guide SOA practitioners to conceptualize, analyze, design, and architect their service-oriented assets. The Service-oriented modeling framework (SOMF) in fig 3 offers a modeling language and a work structure or "map" depicting the various components that contribute to a successful service-oriented modeling approach. It illustrates the major elements that identify the "what to do" aspects of a service development scheme. The model enables practitioners to craft a project plan and to identify the milestones of a service-oriented initiative. SOMF also provides a common modeling notation to address alignment between business and IT organizations.

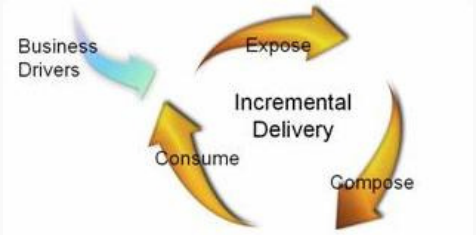

Fig. 2. An incremental, business-driven approach to SOA Hadi rt al(2009)

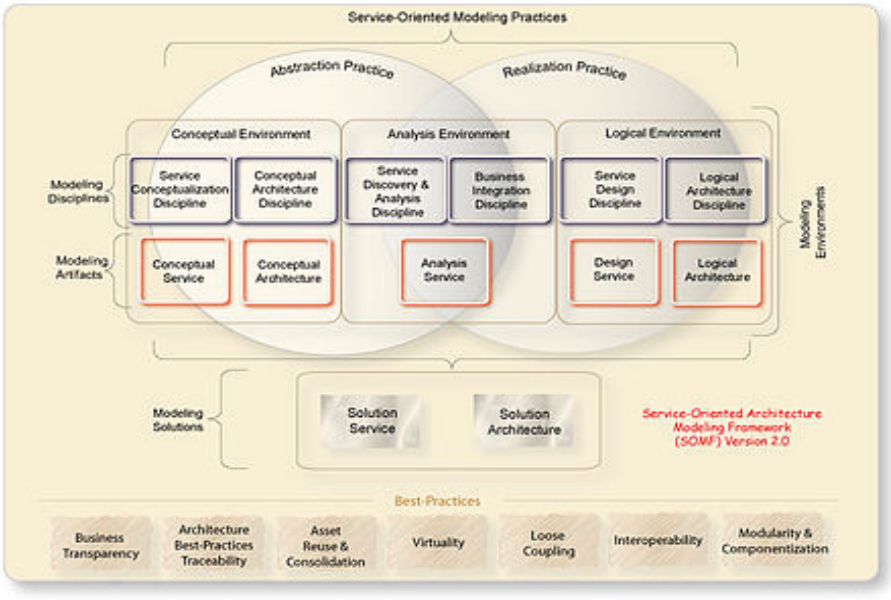

Fig 3. Service Oriented Modeling Framework (SOMF) Version 2.0 Christopher (2005)

ISSN: 2223-9944, e ISSN: 2223-9553

Services are commonly used to expose IT investments such as legacy platforms and Line of Business applications. Services can be assembled (or "composed") into business processes, and be made available for consumption by users, systems or other services. The process is an iterative one of creating ("exposing") new services, aggregating ("composing") these services into larger composite applications, and making the outputs available for consumption by the business user Hadi et al, (2009).

Fundamental to the service model is the separation between the interface and the implementation. The invoker of a service need only (and should only) understand the interface.

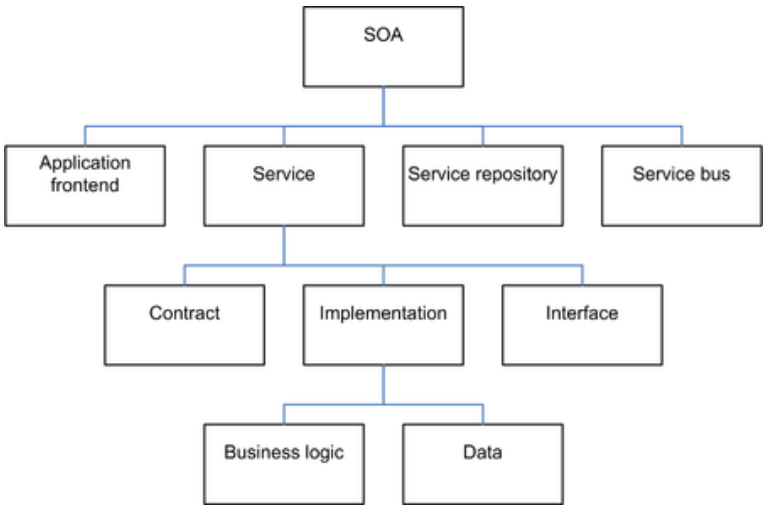

Fig 4. Elements of SOA (Hierarchical Model) Cardoso et al (2006).

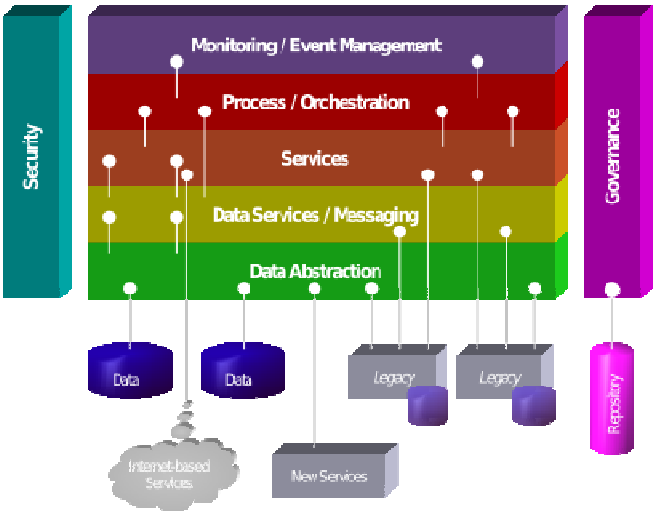

Fig 5. SOA meta-model, The Linthicum Group, 2007

The implementation can evolve over time without disturbing the clients of the service. Several key benefits of service orientation derive from this abstraction of the capability from how the capability is delivered. This means that, the same interface can be offered by many implementations, or conversely, that implementations can change without affecting the aggregate application. At its most abstract, service orientation views everything — from the

ISSN: 2223-9944, e ISSN: 2223-9553

mainframe application to the printer to the shipping dock clerk to the overnight delivery company — as a service provider. The service model is "fractal:" the newly formed process is a service itself, exposing new aggregated capabilities.

Recurring Architectural Capabilities

As we saw earlier, the SOA architectural models are fractal. This means that a service can be used to *Expose* IT assets (such as a Line of Business system), be *Composed* into workflows or Business Processes (each of which may also be exposed as a service) and be *Consumed* by end users, systems or other services Microsoft White Paper, (2006).. SOA is a fractal, not layered model. While the Abstract SOA Reference Model provides a holistic view of several important SOA concepts, the Expose / Compose / Consume portions of the model should not be interpreted as layers (despite their apparent appearance in the model). Designing a SOA as a set of well-defined tiers (or layers) will constrain the value and flexibility of your services, resulting in dependencies across unrelated components. This is why the Expose / Compose / Consume portions of the model can be thought of as independent architectural initiatives referred to as a *Service Implementation Architecture* (Expose), a *Service Integration Architecture* (Compose) and an *Application Architecture*(Consume). While each of these architectures are designed to be independent of one another, they share a set of five common capabilities.

Fig. 6. Recurring Architectural Capabilities Microsoft White Paper, (2006).

Messaging and Services

Messaging and Services focuses on how messaging is accomplished between senders and receivers. There are a broad array of options and patterns available – from pub/sub and asynchronous to message and service interaction patterns. Services provide an evolutionary approach to building distributed software that facilitates loosely coupled integration and resilience to change. The advent of the Web Services architecture has made service-oriented software development feasible by virtue of mainstream development tools support and broad industry interoperability. While most frequently implemented using industry standard Web services, service orientation is independent of technology and architectural patterns and can be used to connect with legacy systems as well. Messaging and Services are not a new approach to software design – many of the notions behind these concepts have been around for years. A service is generally implemented as a coarse-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled (often asynchronous), message-based communication model. Messages tend to be based upon an agreed-upon set of semantics (such as an industry-standard Purchase Order) and serialized using an interoperable, extensible syntax (usually XML, although alternate approaches like JSON, RNC and ASN1 are sometimes used).

Workflow and Process

Workflow and Process are pervasive across multiple layers of an integration architecture – from formal orchestration of processes to flexible ad hoc systems and collaborative workflow across teams. Since business processes are dynamic and evolve with the organization, the workflow that models these processes must be equally adaptable. In addition, effective workflow involves not only process modeling, but also monitoring and analytics in order to respond to exceptions and optimize the workflow system over time.

ISSN: 2223-9944, e ISSN: 2223-9553

Data

The lynchpin to success in many integration architectures is the ability to provide ***Data*** management. The need to deliver a shared view across disparate, often duplicate sources of data is more important than ever, as businesses strive to achieve a 360-degree view of organizational information. Entity aggregation, master data management, and the ability to make data useful via analytics and mining are crucial elements of integration architecture.

User Experience

Successful integration architectures depend upon both service delivery and the ability to consume services in a rich and meaningful way. Service consumption needs to be contextual, mapping to the natural workflow of employees, customers, and partners. To that end, an integrated ***User Experience*** spanning smart clients, rich clients, lightweight Web applications, and mobile devices enables service consumption by the broadest possible audience.

Identity and Access

To support integrated user experiences, customers require the ability to manage the identity lifecycle – providing integrated Single Sign-On (SSO), access management, directory services, and federated trust across heterogeneous systems. Today, many solutions are built using fragmented technologies for authentication and authorization. In the new application model, access decisions and entitlements need to be made at multiple layers and tiers, in which a federated ***Identity and Access*** across trust boundaries becomes a key requirement.

RESULT

During the lifetime of a service, the service most probably changes in different perspectives as listed below. As a result, one service will probably have to be available in several versions.

1. Difference in interface (e.g. extended interface, but same business object)
2. Difference in semantics with same interface (business objects changed)
3. Difference in Question of Services, e.g. slower but cheaper or high-available but more expensive

The overall Service oriented management must encompass many capabilities, some of which are listed below:

4. A comprehensive solution for change and configuration management, enabling organizations to provide relevant software and service updates to users quickly and cost-effectively.
5. Reducing the complexity associated with managing the IT infrastructure environment with a focus on lowering the cost of operations.
6. Centralized backup services capturing changed files to disk. Centralized backup should enable rapid, reliable recovery from disk while providing end-user recovery without IT intervention.
7. Pre-deployment capacity planning coupled with best-practice guidance and hardware-specific knowledge to help information technology professional make low-risk architectural decisions.
8. Data warehouse and reporting to help IT better support corporate decision making, improve the quality of service provided, and better administer resources through

ISSN: 2223-9944, e ISSN: 2223-9553

improved reporting capabilities and management data integration from a broad variety of resources.

Supporting the Common Architectural Capabilities

The five architectural capabilities discussed above are supported by the Microsoft SOA Platform in fig 7.

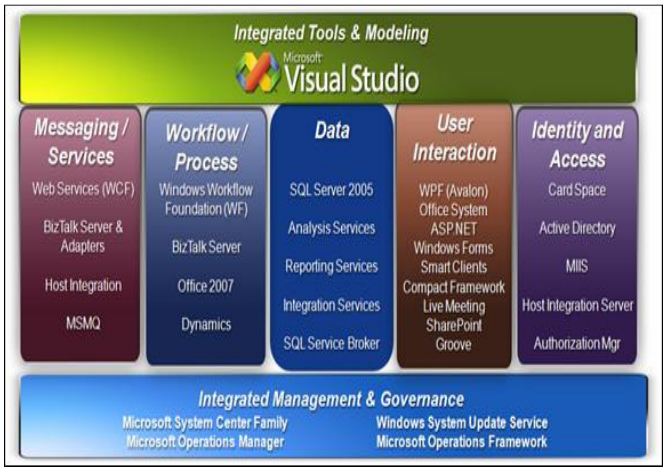

Fig.7. SOA Capabilities on the Microsoft Platform Microsoft White Paper, (2006).

We can also think of these five common architectural capabilities as set of perspectives for viewing and understanding the Abstract SOA Model. The five architectural capabilities serves as a set of lenses to help us view and better understand the challenges associated with *Exposing* existing IT investments as services, *Composing* the services into business processes and *Consuming* these processes across the organization.

THE FUTURE

Network World - The enterprise application as we know it is dead. Zombie-like, it still lumbers along, lifelike enough to fool many IT professionals. But, make no mistake - it is dead.

In today's climate of ever-accelerating change, big, one-size-fits-all applications do not help organizations do business more effectively. The monolithic applications that prevent companies from adapting to new business challenges are giving way to a matrix of services called the service-oriented architecture (SOA) Elizabeth (2005).

The complexity and rigidity of traditional systems results in the too-familiar misalignment between IT and the business. The IT side is bogged down with the burden of maintaining 20th-century systems and processes that have become too bloated and inefficient to deal with the demands imposed by the 21st-century business environment. In many organizations, that maintenance burden approaches 80% of the total IT budget.

This is not news to IT professionals, who for years have put up with these headaches, because they had no alternative to automate critical business processes. Inflexible automation was better than no automation at all, until now. Enterprises can adopt an SOA and create applications composed of modular software components that are interconnected through well-defined, open Web service standards.

Just as companies moved from mainframe to client-server, so must IT move from monolithic applications to a matrix of loosely coupled Web services that enable the composition and recomposition of business processes.

ISSN: 2223-9944, e ISSN: 2223-9553

For example, checking an account balance is a common function in banking applications. In a traditional architecture, each new banking application would include code to check balances. In an SOA, "check account balance" would live as a Web service. Any application requiring that functionality can simply link to the appropriate Web service.

The biggest benefit of SOA is flexibility. An SOA allows the enterprise to quickly change its infrastructure in response to changes in the business environment, because most of the basic processes are already available as Web services. New functionality can be developed as a Web service and linked back to other necessary business processes. The business can adapt and thrive.

While the value and inevitability of building SOAs are generally recognized, many organizations have failed to revamp their approach to building software. Almost everyone understands the importance of a service architecture's ability to deliver cost savings and agility. But we are in the frothy part of the adoption curve, where everyone is adopting SOA - even those who aren't entirely sure how to do it.

But don't be fooled: SOA is not just another overhyped IT trend. Organizations that have deployed well-managed SOAs are seeing enormous gains in cost-savings and infrastructure flexibility. Unfortunately, too many IT departments are building Web services as though they were constructing the cumbersome, stand-alone applications whose demise they are to oversee.

CHALLENGES IN SOA

One obvious and common challenge faced involves managing ***services metadata*** Dion (2005). SOA-based environments can include many services that exchange messages to perform tasks. Depending on the design, a single application may generate millions of messages. Managing and providing information on how services interact can become complex. This becomes even more complicated when these services are delivered by different organizations within the company or even different companies (partners, suppliers, etc.). This creates huge trust issues across teams; hence SOA Governance comes into the picture.

Another challenge involves the ***lack of testing*** in SOA space. There are no sophisticated tools that provide testability of all headless services (including message and database services along with web services) in a typical architecture. Lack of horizontal trust requires that both producers and consumers test services on a continuous basis. SOA's main goal is to deliver *agility to businesses*. Therefore it is important to invest in a testing framework (build it or buy it) that would provide the visibility required to find the culprit in the architecture. Business agility requires SOA services to be controlled by the business goals and directives as defined in the business Motivation Model (BMM).

Another challenge relates to providing ***appropriate levels of security.*** Security models built into an application may no longer suffice when an application exposes its capabilities as services that can be used by other applications. That is, application-managed security is not the right model for securing services.

CONCLUSION

As traditional application architectures evolved from the mainframe application through client server to multi-tier web applications, the applications have remained to a large extent tightly coupled. In other words, each of the subsystems that comprise the greater application is not only semantically aware of its surrounding subsystems, but is physically bound to those

ISSN: 2223-9944, e ISSN: 2223-9553

subsystems at compile time and run time. The ability to replace key pieces of functionality in reaction to a change in a business model, or of distributing an application as individual business capabilities, is simply not possible.

With a Service Oriented Architecture the application's functionality is exposed through a collection of services. These services are independent and encapsulate both the business logic and its associated data. The services are interconnected via messages with a schema defining their format; a contract defining their interchanges and a policy defining how they should be exchanged.

REFERENCES

[1] Cardoso, J., Sheth, A. P. (2006). Semantic Web Services, Processes and Applications. Semantic Web and Beyond: Computing for Human Experience. Foreword by Frank Leymann. Springer. xxi.ISBN 978-0-387-30239-3.

[2] Christopher, K. (2005). A New Blueprint For The Enterprise. *CIO Magazine*.

[3] Dion, H. (2005). "Is Web 2.0 The Global SOA?" *SOA Web Services Journal*.

[4] Elizabeth M. (2005). Building a Better Process. *Computer User* 20pp.

[5] Hadi V., Bavar A.Z., Kh. N.M., Negin D. (2009). A Brief Survey of Software Architecture Concepts and Service Oriented Architecture, in Proceedings of 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT'09, pp 34-39, China.

[6] Hoyer, V. , Stanoesvka, S., K., Janner, T., Schroth, C. (2008). *Enterprise Mashups: Design Principles towards the Long Tail of User Need.* Proceedings of the IEEE International Conference on Services Computing (SCC 2008). Retrieved 2014-01-29.

[7] Microsoft White Paper, (2006). *Enabling "Real World" SOA through the Microsoft Platform.* http://www.microsoft.com/biztalk/solutions/soa/whitepaper.mspx

[8] Schroth, C. ; Janner, T. (2007). *Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services*. IT Professional 9", No.3, pp. 36-41, IEEE Computer Society.

[9] Waldo, J. (2002). "The End of Protocols". *The Source*. Sum Microsystems. Retrieved 2013-12-11.

[10] zdnet.com (2009). "SOA services still too constrained by applications they represent". Retrieved 2014-01-27. www.zdnet.com

ISSN: 2223-9944, e ISSN: 2223-9553