



# SERVICE ORIENTED ARCHITECTURES & OSS

CONTENTS

1. INTRODUCTION	1
2. WHAT IS A SERVICE ORIENTED ARCHITECTURE?	2
3. SERVICE ORIENTED ARCHITECTURE APPLICABILITY	5
4. SERVICE ORIENTED ARCHITECTURE AND OSS TRANSFORMATION	8
5. CONCLUSION	9

# 1. INTRODUCTION

Every CSP today operates in an environment where new low-cost competitors threaten their very existence. Traditional CSPs therefore need to minimize customer attrition and deliver competitive products at the same time as they re-architect both their networks and their supporting systems to enable a substantially lower cost base for delivering these new services more rapidly and cost-effectively in the future.

Certainly the cost-effective and rapid creation of new services in the retail market is key to maintaining a competitive position in a rapidly expanding marketplace. At the same time, traditional CSPs have access to new kinds of business opportunities, where the need to deliver wholesale products to pure play 'service providers' requires rapid delivery of inter-organization integration and communication mechanisms.

To successfully meet these challenges, Communications Service Providers (CSPs) must balance three architectural imperatives:

- > Greater interoperability across the industry, to enable new kinds of business opportunities
- > Greater interoperability and integration within the organization driven by cost pressures
- > The need to deliver transformation to a network and support system infrastructure that supports multiple services and multiple technologies

It is the balance that is critical; the requirement to support both existing and 'next-generation' services mean that a CSP needs to consider not only how to manage business-as-usual service provision, but also how to build a new, agile and flexible OSS architecture that meets the demands of an industry that is undergoing a seismic shift at its core.

More and more CSPs have looked to Service Oriented Architectures (SOAs) – largely an architectural pattern for integration – to deliver these multi-faceted requirements. But early adopters of SOA have, in a number of cases, struggled to realize the expected promise of better and quicker integration. There are a number of reasons for these perceived 'failures' – some of the likely causes and possible solutions are explored in this paper.

## 2. WHAT IS A SERVICE ORIENTED ARCHITECTURE?

Before we explore some of the issues around Service Oriented Architectures and what these mean for OSS, it is worth clarifying what the term Service Oriented Architecture actually means. A Service Oriented Architecture (SOA) is a design approach to developing software that involves some mechanism for defining and utilizing business services (as exhibited by software) in a distributed environment.

SOA can broadly be defined as any architecture that exhibits the following characteristics:

- > Published services
- > Standardized consumption mechanism

In this context, a service is a defined set of business logic that provides specific functionality. For example, the creation, management and completion of an order in an external system might consist of a number of interactions over a variety of distinct interfaces.

Defining and utilizing services in a SOA environment can be achieved by a vast array of different mechanisms defined by a number of different standards – some independent, some not. What is clear is that SOA provides a *pattern* of developing interoperability, and as such, it provides broad guidelines, not a prescriptive formula, for integration.

## 2. WHAT IS A SERVICE ORIENTED ARCHITECTURE? (CONT.)

### 2.1 SOA – THE PROMISE

In any software environment, a range of disparate systems evolve over time. As business requirements change, a single system will no longer meet the business requirement. This has been particularly apparent in the telecommunications industry, and especially where merger and acquisition activity has been intense. In these circumstances, service providers are faced with the choice of either building whole new systems, or taking existing software, and integrating and customizing to fit the new business need. In the past, the cost and effort of integration has been high – too high, and so while functionality from existing systems could have been reused, the effort involved in making existing systems work together has simply been greater than the effort involved in building an entirely new system. As a result, service providers currently have hundreds, even thousands of systems in their Operational Support System estate.

Service Oriented Architectures were borne out of the growth in understanding of distributed computing, and how the associated concepts should be articulated. Simply put, the idea is that software should be developed as components called ‘services’. Developing new functionality should therefore simply be a matter of reconfiguring existing services, rather than starting from scratch.

SOA as a term has gained widespread market recognition over the past five years from work that IBM and Microsoft (and others), have done in defining their strategies around Web Services. As an architectural concept however, SOA has been around for decades.

The promise of this architecture is the ability to rapidly craft interfaces that can be used by other parties relatively quickly – a kind of ‘Morse code’ of integration that is understandable and usable by anybody. The potential is that where the integration issue can be solved, organizations can construct their software to both maximize use and reuse, and enable rapid change through configuration of existing components rather than re-development. In this way, organizations gain the ability to operate at the speed their business demands, rather than being held hostage to the speed at which software development can take place.

In reality though, there are still a number of issues that need to be bridged before true value can be achieved.

The understanding of requirements around Enterprise Application Integration has matured, leading to the generally accepted conclusions that integration must be:

- > Upgradeable
- > Customizable/configurable
- > Scalable/performant
- > Comprehensible
- > Easily achievable

These requirements have provided compelling reasons to develop standards and technology that will meet these needs. The resultant concept of componentized integration points is defined by SOA.

## 2. WHAT IS A SERVICE ORIENTED ARCHITECTURE? (CONT.)

### 2.2 SOA – A QUESTION OF IMPLEMENTATION

While the concept of SOA is relatively straightforward, in practice when it comes to interoperability, different organizations, groups and vendors have chosen to implement SOA in different ways. Current ‘SOA’ implementations revolve around Web Services or Enterprise Service Bus, among others. These are underpinned by a number of implementation standards such as web services, all of which have their proponents and their detractors.

What this means is that even within one service provider, where multiple instantiations of SOA exist, interoperability is still problematic; that is, a service defined under .net cannot necessarily interact with a service defined under Java without further translation. A common approach to design is helpful, but ultimately, a slew of standards means that the quest for one single service oriented architecture has been unfulfilled.

Another key issue around SOA implementation that has been problematic is that implementations tend to persist with their existing, old-style, point-to-point interfaces using SOA principles. The real promise of SOA is only delivered however, when a number of additional factors are carefully weighed before embarking on implementation. These are discussed in some detail in the chapters that follow.

### 2.3 SOA – MAKING IT REAL

As SOA standards have been implemented, and interoperability issues became apparent, so a new standards framework has been developed to address these specific issues. The Open Service Oriented Architecture (see <http://www.osoa.org>) collaboration involves a number of vendors in the SOA space who have encapsulated some interoperability standards in a framework call the Service Component Architecture (SCA). This provides a set of standards that further define not only communication but also information standards that are cross-platform and can hence be implemented in any environment.

SCA allows implementations to focus on the business or system integration in question, rather than the mechanics of making it happen. Consider Figure 1 a Java language implementation.

In this figure we can see that the use of SCA provides a level of insulation between the infrastructure and implementation that allows the user to concentrate on getting the implementation correct without having to be concerned about the specifics of communication, assembly or binding into the infrastructure. As far as the interface is concerned, the interface can be exposed via a Web Service, Enterprise Service Bus, or any other SOA implementation pattern.

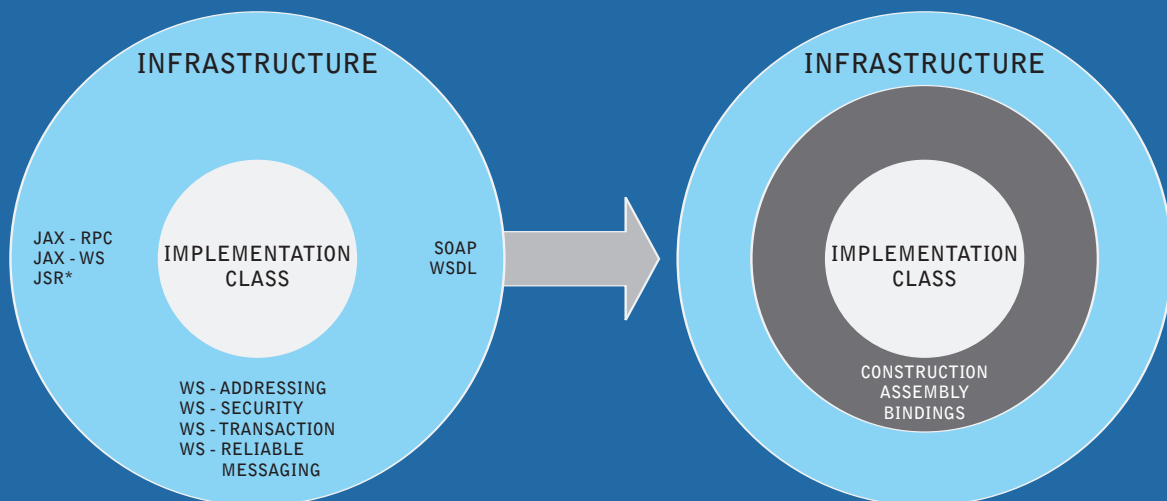


FIGURE 1: SOA DEPLOYMENT – TRADITIONAL VS SCA-BASED

### 3. SERVICE ORIENTED ARCHITECTURE APPLICABILITY

When implementing SOA there are some specific, high-level (architecturally speaking) decisions that need to be made in order to realize the promise of SOA. These fall into the following three categories:

- > Business Applicability
- > Functional Applicability
- > Technical Applicability

#### 3.1 BUSINESS APPLICABILITY

There are three different business issues that need to be considered when developing interoperability using SOA: standards, cost and agility.

When interfaces are extended beyond the corporate boundaries most industries look to their respective standards bodies to provide the direction and specifications that should be employed. Increasingly this is becoming true in the Telecommunications market as the various standards such as defined by the TeleManagement Forum have matured. Fortunately, and perhaps not unsurprisingly, most of the specifications of interfaces have been expressed in terms of SOA-compliant technologies, with the MTOSI and OSS/J standards both defined in terms of XML messages that can be delivered over a variety of SOA patterns. Implementing standards-based interfaces, especially over SOA, has both short- and long-term benefits. Short-term benefits such as shortened timescales through clearly defined information and binding models and reduced risk in terms of non-compliance with regulatory demands, mean that a standards-based approach utilizing SOA, where possible, should be considered before all others. Long-term benefits revolve around softer, business agility related factors, but also include tangible lower cost of ownership improvements including reduction to maintenance, management and upgrade costs.

The second business applicability issue is cost. Too often the issue of higher initial cost of SOA, compared to more traditional 'needs-based' approaches, is seen as a stumbling block for the implementation of SOA patterned interfaces. It is our view that a longer-term view should be taken. When due consideration is paid to the ongoing effort associated with maintenance, troubleshooting and management of turn-key interfaces —the Total Cost of Ownership (TCO)— there is a compelling argument to use SOA. By utilizing the maturing SOA standards and leveraging the infrastructure that accompanies those, CSPs are assured of a lower cost of ownership of the interfaces.

The last aspect of business applicability for SOA is agility. The ability to bring new products and services to market rapidly is a key competitive differentiator. CSPs are looking for their application infrastructure to drive, not hinder, speed to market. One of the key constraints to this ability has always been that new products typically required new operational support systems and all the associated integration to the existing environment, and this resulted in protracted IT projects. By leveraging the SOA infrastructure, a CSP can be assured that the cost and speed of implementation in the operational environment will be less of a constraint on new product launches.

Vendors are responding to the business requirement for interoperability. Just as Cramer6 OSS Suite has for a number of releases now presented SOA pattern services as standard, so a number of other OSS domain vendors and even equipment manufacturers (witness Cisco's ASA) have developed standard, published SOA patterned services. By leveraging these standard interfaces, a CSP can realize the benefits of lower risk, lower TCO implementations that aid faster time to market.

## 3. SERVICE ORIENTED ARCHITECTURE APPLICABILITY (CONT.)

### 3.2 FUNCTIONAL APPLICABILITY

Functional applicability of interfaces, specifically in relation to SOA, revolves around two main arguments: granularity of interfaces and reusability of interfaces.

When the decision to use a SOA pattern has been made, the issues of granularity and reusability seem to be in direct competition for deciding the semantics and specific implementation of the interface. More fine-grained interfaces allow for greater uncertainty in future reuse situations; so choosing these may seem to be the better option. But this must be weighed against the ease of use of the interface; more coarse-grained interfaces are easier to understand and are likely to be closer to well-understood business processes.

A good example to illustrate the difference in fine and coarse-grained interfaces is the difference between OSS/J and MTOSI. OSS/J take the view that interfaces should be implemented as highly granular interfaces, with multiple calls to the interface required to carry out any specific function. MTOSI on the other hand, have a view that a single call to a more abstract interface is most desirable.

Cramer, Amdocs OSS Division, believes that abstract (in terms of functionality of course – not specificity) interfaces are best suited for integration with external systems regardless of whether those interfaces are northbound or southbound. The reason for this is that we believe that abstract interfaces are more easily aligned with the ability to rapidly roll out new services, including the integration components. But the choice of the wrong granularity of interface can easily prevent any possibility of reuse of interfaces, and at the worst can result in them having to be developed afresh for each new product offering. We believe that balancing high-level business logic interfaces with experience of the business functions most often required is the best way to select the appropriate granularity of interface. In this way, the CSP can be assured that multiple services can use the same interface, thereby minimizing the impact of interface requirements on launch of new products and product bundles.

As an example – the Cramer6 OSS Suite provides an abstract Order Fulfillment interface that is completely reusable across any product. The ability to define inside the OSS Suite, rather than in the interface, the data requirements for any specific product, means that the enabling of an order interface for a new product is more about defining the product than specifying and developing additional interfaces. Contrast this with the interfaces specified by the OSS/J standard where multiple service calls are required to complete each order, and each new product that might have different options requires a different orchestration of the order interface. This highly granular interfacing is exacerbated by the performance issues that are associated with many vs. few interface calls.

## 3. SERVICE ORIENTED ARCHITECTURE APPLICABILITY (CONT.)

### 3.3 TECHNICAL APPLICABILITY

The final area of applicability that requires some consideration is that of technical applicability. The main areas for consideration here are:

- > Performance
- > Security
- > Reusability

When deciding what SOA mechanisms to use in implementing specific interfaces, a CSP should be aware that there are practical as well as architectural limits in performance that are associated with every SOA transport and binding mechanism. Due diligence in ensuring that performance of the specific mechanisms employed are in line with the business requirements is one aspect, ensuring that the correct choice of asynchronous or synchronous mechanisms is just as important. While asynchronous mechanisms may have a negative impact on performance, they often provide levels of resilience that synchronous mechanisms do not.

Security considerations should also play a significant role when making technical applicability decisions. Deployment beyond the enterprise borders can be a daunting exercise in purely technical terms and it is important that every CSP undertaking these integrations be aware of security aspects of the implementations.

The nirvana promised by standards such as Web Services have also been found wanting – especially at implementation. Organizations that have tried to implement services that are intended to be consumed by a variety of mechanisms and across multiple platforms have found, to their dismay, that reimplementation (or at the very least redeployment) is necessary from one mechanism to another. The ability to reuse business logic over a number of different interfaces is simply been too difficult.

This specific problem was recognized some time ago by a number of vendors resulting in the creation of the Open Service Oriented Architecture Group. The culmination of their efforts to address these interoperability issues is a new standard – recently handed over to OASIS – called Service Component Architecture. SCA is a platform neutral specification for the creation, assembly and binding of SOA interfaces that already has support in a number of environments, including Apache Tuscany, IBM Websphere, BEA WebLogic, and Tibco ActiveMatrix. Service providers should look to their infrastructure suppliers, including OSS vendors, to support these initiatives when it comes to technical applicability decisions.

## 4. SERVICE ORIENTED ARCHITECTURES AND OSS TRANSFORMATION

One of the key areas where Services Oriented Architectures can make a real difference is in the area of OSS Transformation. A typical scenario faced by service providers wanting to undertake OSS Transformation is the challenge of merging new and old software architectures and technologies while carrying on 'Business as Usual'. The architectural and platform differences can present a serious obstacle to transformation, especially when the integrity and security of existing services need to be protected. When faced with a disparity of existing systems, often on a range of platforms that have a brittle integration infrastructure underpinning their interoperation, service providers often take the path of least resistance and leave existing OSS as is – while augmenting with a new NGN-focused one.

A more pragmatic, and cost-effective, solution would be to SOA-enable the existing infrastructure – especially those elements that are to be retained in the future model. By publishing their capabilities, or enabling them to make use of external capabilities through a SOA component approach, service providers can be assured of lowering both the costs and risks associated with a transformation program. This is the one instance where the granularity of interfaces over SOA patterns is less important, and their suitability to their specific purpose is of greater importance.

Regardless of the OSS Transformation Model (see OSS Transformation: A fundamentally different approach) employed, service providers need to be cognizant of the integration cost and effort associated with change. Underestimating the challenges of integration across platforms and architectures, especially when legacy applications were not designed for interoperability, can have serious consequences for the operational effectiveness, not to mention the profitability of the organization.

Throughout a transformation program service providers should expect to build out increasing numbers of SOA-enabled interfaces to legacy systems, and then phase those interfaces out as components in the OSS are replaced or retired. The relatively low cost of delivery of SOA interfaces ensures that the cost associated with transition can be minimized, without adversely affecting the risks associated with change.

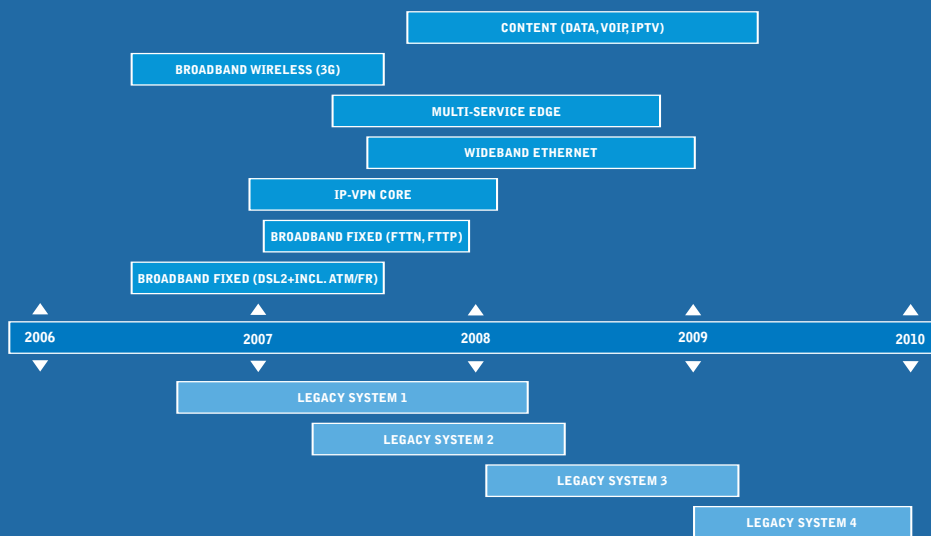


FIGURE 2: EXAMPLE TRANSFORMATION PROGRAM

## 5. CONCLUSION

The ability to support Services Oriented Architectures has already become a defined requirement for many organizations. When considering Operations Support Systems requirements, service providers should consider the SOA capabilities that are provided by individual vendors, especially when OSS Transformation is envisaged, and also the specific SOA patterns that deliver the highest return on investment without sacrificing organizational agility.

The ability to maintain business as usual, while undertaking an OSS renewal or transformation program to address Next-Generation Networks requires more than just careful planning. The need to rapidly implement a standardized integration environment should be a key consideration in evaluating vendor and in-house solutions to both existing and future OSS.

## ABOUT AMDOCS

Amdocs is the market leader in customer experience systems innovation, enabling world-leading service providers to deliver an integrated, innovative and *intentional customer experience*<sup>™</sup> – at every point of service. Amdocs provides solutions that deliver customer experience excellence, combining the software, service and expertise to help our customers execute their strategies and achieve service, operational and financial excellence. A global company with revenue of \$2.48 billion in fiscal 2006, Amdocs has over 16,000 employees and serves customers in more than 50 countries around the world. For more information, visit Amdocs at [www.amdocs.com](http://www.amdocs.com).

## ABOUT CRAMER, AMDOCS OSS DIVISION

Cramer, Amdocs OSS Division, was formed following the acquisition of Cramer, a leading provider of operations support systems (OSS). The combined Amdocs-Cramer solution is unique in its combination of OSS and BSS, delivering complete visibility of the customer, the network and the service. This will help service providers transition from legacy to next-generation networks and systems, and rapidly launch new converged services that quickly turn network investment into service revenue.

Amdocs has offices, development and support centers worldwide, including sites in:

THE AMERICAS:	ASIA PACIFIC:	EUROPE, MIDDLE EAST & AFRICA:			
BRAZIL	AUSTRALIA	CYPRUS	HUNGARY	THE NETHERLANDS	SPAIN
CANADA	CHINA	CZECH REPUBLIC	IRELAND	POLAND	SWEDEN
MEXICO	INDIA	FRANCE	ISRAEL	RUSSIA	TURKEY
UNITED STATES	JAPAN	GERMANY	ITALY	SOUTH AFRICA	UNITED KINGDOM
	THAILAND				

For the most up-to-date contact information for all Amdocs offices worldwide, please visit our website at [www.amdocs.com/corporate.asp](http://www.amdocs.com/corporate.asp)



© Amdocs 2007. All Rights Reserved. Reproduction or distribution other than for intended purposes is prohibited, without the prior written consent of Amdocs. Amdocs reserves the right to revise this document and to make changes in the content from time to time without notice. Amdocs may make improvements and/or changes to the product(s) and/or programs described in this document any time. The trademarks and service marks of Amdocs, including the Amdocs mark and logo, Ensemble, Enabler, Clarify, Return on Relationship, DDP/SQL, DDP/F, Intelecab, STMS, Collabrent and Intentional Customer Experience are the exclusive property of Amdocs, and may not be used without permission. All other marks are the property of their respective owners.

© Cramer Systems Limited, 2007  
Effective Date: May 2007

Cramer and the Cramer logo, whether or not appearing with the registered symbol, are registered trademarks of Cramer Systems Limited. Any third-party products or company names referred to may be trademarks of their respective owners. This document is valid from the "Effective Date" shown and supersedes any previous issue. It contains summarized information that will be subject to periodic change and customers are advised to check with Cramer Systems that they are using the current version. Cramer Systems has made all reasonable endeavors to ensure that the statements contained within this document are accurate at the time of publication but cannot guarantee that the document is free from errors.