

# The *Service Oriented Architecture Manifesto*: The Power of Service-Orientation

## Executive Summary

This paper examines the *Service Oriented Architecture (SOA) Manifesto* and provides a commentary regarding what it means and how it might be used.

By way of background the *SOA Manifesto* was presented at the Service-Oriented Architecture Symposium in Rotterdam in October 2009. It was presented after a three-day closed session with the 15 participants who are the authors.

The authors come from a range of companies, including systems integration, consulting, vendors and analyst companies.

The *Manifesto* does not tackle any relationship between process orientation and service-orientation. In this whitepaper we do tackle this link and explain why they go hand in hand. The only reason this was not done in the context of the *Manifesto* was time and focus.

The centered quotations in the narrative below are taken directly from the *Manifesto*<sup>5</sup>.

## The Manifesto

*“Service orientation is a paradigm that frames what you do. SERVICE ORIENTED ARCHITECTURE (SOA) is a type of architecture that results from applying SERVICE ORIENTATION.”*

There is no doubt that service-oriented architecture, or SOA, has been mis-represented over a long period. This accounts for the hype cycle that immediately followed its introduction in 2000/2001 and the trough of disillusionment from which we are escaping. This initial statement is intended to clear up the hype and focus on what these terms really mean, it is only by understanding the true meaning of these terms that we can focus on the key benefits, the values and principles that this genre of technology can bring.

Service-orientation (SO) is a paradigm, or way of thinking, that constrains how you create a solution. The key to service-orientation is the notion of a service contract (SC). To the layman a service contract, in its abstract form, is the equivalent of any contract to provide a service that we might see in the real world. This could be a washing machine with a guarantee for fixes. The guarantee has specific service levels in terms of time to fix and replace and the washing machine comes with a set of documented functions and a description of what order buttons are pressed and dials turned to achieve some outcome.

In our world of service-orientation, a service contract describes what a service does, and its capabilities. A service contract describes a set of service functions that may be called, a set of service parameters that may be passed to the

service functions, any ordering rules across the service functions, and any non-functional requirements (service policies) such as expected levels of performance (SLAs), security, transactionality and so on. In this way a service contract helps us understand how to use the capabilities that the service provides in terms of functionality, semantics (ordering), performance, and security. A concrete form of a service contract would be a Web Services Description Language (WSDL) XML document that might also include Web Service Policy (WS-Policy) attachments. However, it is very important to understand that this concrete example only provides service functions and static service policies. That means it does not provide any ordering constraints nor any SLAs.

The term service-oriented architecture is really a type of thing and therefore classifies instances that conform to it. Thus we may have an instance of a service-oriented architecture but service-oriented architecture is not an instance in and of itself. The term service-oriented architecture is indicative of some set of components that exhibit clearly defined capabilities accessible through a service contract. That is you cannot have an instance of a service-oriented architecture without some form of a service contract. The service contract provides the de-coupling of capability from implementation. There may be many dissimilar instances of a service-oriented architecture but they will all have in common this notion of service contract and the de-coupling that comes with it. They may use message passing or remote procedure calls to interact with a service through its service contract. They may have different forms of service repositories which store service contracts. But they will all have service contracts and some service repository that stores the service contracts and mediate between a caller or user of a service and the implementation of that service through the service contract.

This is why the *Manifesto* makes it clear that service-orientation is a paradigm and service-oriented architecture is a type in which the instance of that type are created through the application of the principles which underlie service-orientation .

*“We have been applying SERVICE ORIENTATION to help organizations consistently deliver sustainable business value, with increased agility and cost effectiveness, in line with changing business needs.”*

In all cases the *Manifesto’s* authors have been and continue to be involved in delivering business benefits by harnessing information technology that is both agile and cost effective and able to flex to accommodate inevitable changing business requirements. In doing this, the authors have harnessed service-orientation and delivered instances of service-oriented architectures for many years.

### Value Statements

The value statements reflect a desired prioritization of one value over another. It does not mean that the value with a lower priority is not important it just means that the value with higher priority is seen as more important. The value statements help prioritize key decisions. For example having a technical strategy with no business value makes little sense because the technical strategy is functionally dependent on business value. That is business value exists with or without a technical strategy but a technical strategy cannot exist in any meaningful way without business value driving it.

Sometimes the second statement will take precedence over the first one. But in those rare cases, and they should be rare, we need to do so with our eyes wide open.

*“Business value over technical strategy”*

Service-orientation and service-oriented architecture are not about technology. One may use technology in the application of service-orientation and in delivering an instance of a service-oriented architecture. They are about

The term service-oriented architecture is indicative of some set of components that exhibit clearly defined capabilities accessible through a service contract.

One may use technology in the application of service-orientation and in delivering an instance of a service oriented architecture. They are about delivering business value.

**Drift is a common concern and when siloed projects drift away from the strategic goals and focus on short-term issues at the expense of the bigger picture, the result is a more costly solution that has to be re-factored and/or re-built sooner than was anticipated because it failed to meet the strategic goals.**

delivering business value. When we apply service-orientation we focus on business value. When we seek to make things flexible, and when we seek to rationalize capabilities, we do so because of some intrinsic business driver that has some measurable effect so that we can look

back and say “here is the business value.” The business value is the value that the business identifies and the metrics that we use to assess it are jointly agreed upon by the business and IT, jointly because if we do it right IT can help collate and gather the metrics and build them into a solution.

In the world of enterprise architecture and solution architecture, we measure the outcome of deliverables against agreed business expectations.

The technical strategy that is employed is created based upon what we need to focus on technically to deliver business value. Many projects fail because they lose sight of the alignment to business. Ensuring alignment to business value is the key to good governance to application and technology roadmaps and to so many other things. Without alignment we should question the worthiness of a technical strategy.

#### *“Strategic goals over project-specific benefits”*

With effective enterprise architecture, applied to a business landscape, we seek to understand the vision and the mission statement of an enterprise. The vision is aspirational and forward looking, the mission is actionable and focused on the immediacy of the enterprise and business as usual. The vision and mission are used to frame strategic goals for the enterprise. The strategic goals might be implemented by one or more programs of work. These programs may incorporate many projects and some or all of these projects may have a focus on IT. One of the roles of an enterprise architect is to ensure that the programs align with the strategic goals, and that the projects reflect them. Without the transparency and traceability of strategic goals to the vision and mission, projects cannot expect to deliver against them. Drift is a common concern and when siloed projects drift away from the strategic goals and focus on short-term issues at the expense of the bigger picture, the result is a more costly solution that

has to be re-factored and/or re-built sooner than was anticipated because it failed to meet the strategic goals.

#### *“Intrinsic interoperability over custom integration”*

The easier and faster it is to compose new capabilities, the more cost effective IT becomes in supporting the business and its changing needs. If we always have to place a mediator between services to reuse them, then we are falling into the trap of custom integration. What this tells us is that the design of our services does not support intrinsic interoperability. Moving to document-centricity as opposed to a functional-centric approach encourages greater interoperability but compromises the ease-of-use or understanding of a services capabilities. It is often easier for people to see functions than understand documents and types. No where else in designing services is it more important to get the functions and the types of the parameters correct and to be unafraid of re-factoring over time to move towards greater intrinsic interoperability.

#### *“Shared services over specific-purpose implementations”*

The granularity of services in an implementation of an instance of a service-oriented architecture is a common cause for concern. Reuse and the sharing of capabilities through the embodiment of services and their contracts seems to be much harder than anticipated. One of the reasons for this is that sharing is a distinct concept that is different to reuse. Sharing might mean several users use the same service directly. Reuse might mean that the service is used in one or more composite services. The former is easier to manage and the latter, if not carefully considered, may change the original business case for a shared service.

What we all know instinctively is it is never a good idea to re-develop the wheel. Yet we constantly re-develop the car. Cars are created through the composition of many existing components. The new Beetle makes use of the Golf chassis, but it feels like a very different car. Services as components for the construction of new capabilities is a key principle of service-orientation, but doing it is harder than saying it. What is needed are robust component repos-

itories with strong documentation and reviews. Back in the 1980's, Shlaer and Mellor had the same idea describing the need for a book of components for things we use to create new capabilities; this is analogous to those components that electrical/electronic engineers use habitually in creating new electrical capabilities. Sometimes a custom implementation is needed but it is not that often and measurements of custom implementation versus reuse should always be made on a regular basis to ensure that the benefits of shared services are being realized. In fact, such a report is something one should expect to see at an enterprise governance board.

*"Flexibility over optimization"*

There is an old software proverb that rings true. Make it work before making it fast. Making things flexible, choosing where the points of flexibility need to be, is not easy. But by following some simple principles we can move much of what is needed outside of the normal code base, making services configurable. Business rules can be used in place of coded logic; orchestration can be used in place of hardwired configuration. A heuristic approach that might appear in a design principles document for service implementers would be to ensure that business rules and processes stated in requirements are exposed to enable them to be changed.

*"Evolutionary refinement over pursuit of initial perfection"*

The old adage of don't try to boil the ocean comes to mind. We always seek perfection but we always need to temper it with the realization that perfection is a focus not a goal. Effective software projects and programs are phased. Occasionally we will see a big-bang approach, but they rarely succeed. We need to be mindful that we cannot do it all in one go and phase the overall solution. It is common for many service-orientation projects and programs to be led by the user interface inwards. Service enabling as you go is a better approach than trying to service enable and then figure out how users interact with the services. Without the users we cannot be sure of the pain points, as well as the wishes and desires of those users, and therefore cannot address their concerns. It is the users

(internal agents and external customers) that matter and an evolutionary approach is more supportive as it provides evolutionary value out to them, which ensures continual support of their needs.

## Principles

The principles of service-orientation are high-level principles. They do not deal with design or organizational issues but may allude to them. They are part of the *Manifesto* because they act as a guide to approaching service-orientation and how to apply the principles of service-orientation. Following these principles will help ensure that whatever gets delivered provides the necessary short- and long-term business value to justify its existence.

*"Respect the social and power structure of the organization."*

At an enterprise level, service-orientation and service-oriented architecture, provide a more open mechanism for accessing capabilities through their service contracts. It makes it easier to reuse existing services and create new ones in line with business needs. What is very important in looking at this, from an enterprise perspective, is who owns the service, who has the budget and how do we cost account for the use of a service. Respecting the social and power structures of an organization is critical in understanding the economic effect of a service, how it is used and how it can continue to be used over time. Aligning budgets for services and aligning the use of services to some cost-accounting mechanism requires that budgets are aligned to the power structures and that the service is cost-accounted for, based on the social and economic power structures within an organization. Getting this wrong can result in service degradation over time as more people use it and budgets for its subsequent maintenance and operational use fail to reflect its use

**Making things flexible, choosing where the points of flexibility need to be is not easy but by following some simple principles we can move much of what is needed outside of the normal code base, making services configurable.**

**Service enabling as you go is a better approach than trying to service enable and then figure out how users interact with the services.**

and importance to the organization as a whole.

*“Recognize that SERVICE ORIENTED ARCHITECTURE ultimately demands change on many levels.”*

This is a corollary of the previous principle. Adopting service-orientation and implementing and using a service-oriented architecture requires alignment across the organization. Services are used, services change, and services are composed, all of which can happen across the entire organization. Use happens from all parts of the organization; request to change comes from all parts of the organization; and composition of new services may happen anywhere in the organization, and may use

**The capabilities that services expose become un-siloed and that makes them enterprise assets and not just assets for one business unit.**

services from other parts of the organization. This is why service-oriented architecture demands change at many levels. The capabilities that services expose become un-siloed and that makes them enterprise assets and not just assets for one business unit.

This democratization of capability requires change on all levels of an organization as users express future desires driven by changing needs and as the implementation of a service-oriented architecture makes flexibility easier to achieve.

*“The scope of SERVICE ORIENTED ARCHITECTURE adoption can vary. Keep efforts manageable and within meaningful boundaries.”*

This is a principle derived from the evolutionary value statement. It is far better to aspire to the benefits of a service-oriented architecture but realize them incrementally. Typically one might

**Never forget Return on Investment, it pushes and pulls development.**

start with the portalization of some business unit and service enablement along the way. The additional cost for service enablement is mitigated by a standards-

based approach to the integration of the underlying capabilities and the needs of the portalization. We believe this approach is complemented by additional concurrent but offset projects that then seek to further service enable along the way as the portalization efforts move to another business unit. Portalization is not the only way to start. One

might have specific needs to reuse components or applications simply to make available data and process from several applications but to do so in a more flexible way. Service-oriented architecture adoption starts by making it manageable within meaningful boundaries so usage can be effectively measured in terms of the business value that it yields. Never forget Return on Investment, it pushes and pulls development.

*“Products and standards alone will neither give you SERVICE ORIENTED ARCHITECTURE nor apply the SERVICE ORIENTATION paradigm for you.”*

Products and standards do not apply service-orientation to deliver a service-oriented architecture but they may help. It is people who wield them, making sure you have the right skills to get the most out of a service oriented approach and get the most out of a service-oriented architecture is a key aspect of any service oriented program regardless of how it is constructed or the boundaries in which it fits. Cognizant concentrates on matching the right resources to achieve a positive outcome, and the role of the enterprise architect is key to ensuring that all of the moving parts of the Software Development Lifecycle (SDLC) work together.

*“SERVICE ORIENTED ARCHITECTURE can be realized through a variety of technologies and standards.”*

A service-oriented architecture does not have to rely on an Enterprise Service Bus (ESB), WSDL, BPEL or any other standard or technology. The key aspects of service-orientation and service-oriented architecture are the descriptions that we call service contracts and the de-coupling of them from the implementation of their capabilities at both design and runtime. An instance of a service-oriented architecture may use asynchronous messaging and pass documents between services. An instance may use WSDL invocations with some form of remote procedure call. An instance may use BPEL and it may use business rules. In all cases the de-coupling should be self evident. A yard stick for an instance of a service-oriented architecture is to create a service repository, because without one, de-coupling invocation on a service contract from a service implementation becomes impossible to achieve in any uniform way.

*“Establish a uniform set of enterprise standards and policies based on industry, de facto, and community standards.”*

A bit of motherhood and apple pie but something that the can certainly catch you out if you don't do it. Establishing a uniform set of enterprise standards and policies can reduce the risk of mis-delivery of a solution. Where there are industry standards, they often leverage XML as a means of encoding information, and it is better to use them than not. It can cut down integration costs and reduce the time it takes to deliver an enterprise information model as well as the processes that manage that information. The enterprise standards should cover all aspects of governance and should support rapid change. The policies should guide security choices, transactions, and service levels. One key relationship between a service-oriented architecture and this principle is that many industry and de-facto standards are seeking to use the same service oriented approach to describing the processes that underpin standards, this is true in International Standards Organization, the International Swaps and Derivatives Association, Healthcare Level 7 and many others. For those services that are outward facing this may mean that the service contracts are already defined, which can help reduce the cost of external connectivity and provide cross enterprise interoperability.

*“Pursue uniformity on the outside while allowing diversity on the inside.”*

This is a cry for service contracts and process uniformity on the boundaries of the enterprise and diversity as to how you achieve implementation on the inside. Service contracts mandate a degree of uniformity. We can go further by publishing our external boundaries in terms of the service contracts we wish others to see and the ordering rules over those contracts that make up the processes that we wish to make visible. In so doing we ensure a higher degree of interoperability without compromising the diversity or implementation on the inside.

*“Identify services through collaboration with business and technology stake-holders.”*

From a technology platform perspective it is important to identify core technology services.

These will have the highest reuse rating and you can only do this by understanding the technical requirements from the technology stakeholders. Likewise the only way to gain reuse and or sharing at the business level for business services is to do it with the business stakeholders. The widest collaboration, across business units will yield the widest reuse and propensity to share, even if you have a single project, it is worth gathering requirements from a wider set of stakeholders so that you can plan effective reuse, sharing and change management over time. Hence, as the *Manifesto* says, “Recognize that service-oriented architecture ultimately demands change on many levels” and “Identify services through collaboration with business and technology stakeholders” that are complimentary.

*“Maximize SERVICE usage by considering the current and future scope of utilization.”*

Wider collaboration for effective service identification is part of the process of understanding the usage of both current and future services and their utilization. Don't just look at the functional aspects because scalability will be compromised as usage rises. Taking into account scalability issues mitigates the risk of retrenchment and ultimately failure of large-scale enterprise adoption.

*“Verify that SERVICES satisfy business requirements and goals.”*

This principle may well seem obvious, but it has profound consequences. One of the things that service-orientation and service-oriented architecture instances provide is a clean separation of service contracts from service implementation. It matters not if you use WSDL to do this or SOAML or Abstract BPEL, what is important is that before a service is constructed (implemented) we can, at worst, manually check that the collection of service contracts match the requirements that gave rise to them. If these requirements are themselves derived from higher level business

**Where there are industry standards, they often leverage XML as a means of encoding information, and it is better to use them than not.**

**The widest collaboration, across business units will yield the widest reuse and propensity to share, even if you have a single project, it is worth gathering requirements from a wider set of stakeholders so that you can plan effective reuse, sharing and change management over time.**

requirements we can check the derivation. And if the higher level business requirements are derived from business goals we can check their alignment, too.

When we do this manually, and most effective service-oriented architecture practitioners do exactly this; it is called service-oriented architecture governance and it is part of enterprise architecture governance. It is part of the enterprise alignment process that is done manually.

Given that service contracts can be properly described in a formal and structured way (i.e., WSDL, SOAML and Abstract BPEL), and given

**If you get flexibility engineered in from the start then the cost of suggested change will be reduced and the ability of the enterprise to optimize how it does things will increase.**

that we can encode the requirements that give rise to these service contracts, it would make a lot of sense to get computers to check the service requirements against service contracts and automatically show that the "SERVICES satisfy their business requirements". We are not

suggesting at this point that we can do the "satisfiability" test through the different levels of requirements or indeed the business requirements against the business goals – although even this is possible. What we are suggesting is that the immediate requirements (service requirements) that give rise to the service contracts can be automatically checked to show that the services meet their service requirements.

This is what testable architecture<sup>1</sup>, unique to Cognizant, is all about and the wider "satisfiability" is what Savara<sup>4</sup>, a joint Red Hat-Cognizant open source initiative, will deliver over time.

*"Evolve services and their organization in response to real use."*

Accept that we live with imperfection but strive to attain it. Accept that things change. In this way we can focus on evolution to ensure that services meet their business requirements as those same requirements evolve over time. Equally we may choose to re-organize services for the same reasons. We may choose to uncouple a composite or to change it to reflect changes in real use. The best user interfaces are

always tested by users. Apple have been the masters of getting it right and one only needs to look at the take-up of the iPhone and iPod. Apple user tests more extensively and methodically than any other software company. Doing the same for services from the top down, the middle out and the bottom up by evaluating their use in real situations will reveal improvements in design, improvements to requirements and manifest improvements with continual alignment to business requirements and goals.

*"Separate the different aspects of a system that change at different rates."*

Engineering for change is not easy. It requires an understanding of what is needed today against what might be needed tomorrow. A key metric is how executive management and in turn their immediate management team steer the ship. A question to resolve, for example, is to understand the controls and the points of change that they need to do their job. Understanding industry influences and trends on an enterprise -- and understanding the vision -- provide a way of scoping what needs to be made flexible in the future. Understanding the mission and strategic goals provide a scoping of the changes needed in the near term. Having effective change management procedures in place and effective change request processes are imperative to continual improvement.

Another key aspect of rates of change is that processes change infrequently, but when they do change, the changes relate to key decision points. We might think of these as business rules, which change far more frequently. Hence outboarding business rules and using a business rules service is an important architectural pattern which complements outboarding of processes through orchestration and workflow.

Don't think that what you deliver will not change or is optimal. Let the users use and make suggestions. If you get flexibility engineered in from the start then the cost of suggested change will be reduced and the ability of the enterprise to optimize how it does things will increase.

*"Reduce implicit dependencies and publish all external dependencies to increase robustness and reduce the impact of change."*

When we create composite services, just like when we create views in a database, it is important to understand the services from which they are derived. Without documenting and making these dependencies clear we potentially run into huge change management and governance problems. It should be standard best practice to publish the key dependencies both in terms of the services that may be used and the underlying technology dependencies, too. Such dependencies should be documented and published as some form of attachment to services in the service repository.

*“At every level of abstraction, organize each service around a cohesive and manageable unit of functionality.”*

Effective design combines concepts into some form of cohesion. This is true for any piece of software that is constructed. Services are no different in this regard and at each level of abstraction be it a business service, a technical service or a data service, the principle of cohesion applies. Organizing and making things manageable takes effort and discipline but the rewards are often greater use and reuse because the focus of the service becomes clear and the concepts coalesce to deliver appropriate functionality for the level of abstraction.

## Process and Service-Oriented

Process orientation is a paradigm that constrains a solution to focus on the underlying flows between components where a component may be a person or an agent for a person (e.g., a portal) or a software component (which maybe embodied by a service contract).

Business Process Management (BPM) is the management of process that are formulated by applying process orientation. A business process management system is a software platform that supports the construction of process descriptions, their simulation, execution and total lifecycle management.

Orchestration (as embodied by WS-BPEL) is a mechanism for describing, simulating and executing composite services which are “orchestrated” by means of some flow between them to deliver a new service.

Today, BPM and orchestration are common bed-fellows. Many companies use the former to deliver the person-based workflows and the latter for composing complementary services, which may then be consumed by a BPM layer. In the future these two bed-fellows will merge with the introduction of WS-HumanTask and Bpel4People. These two related standards will deliver the necessary descriptions to enable both BPM and orchestration to be merged.

When we think of BPM and service-oriented architecture we need to understand where things are today and where they may be tomorrow and deal with the present and plan for the future. Typically a BPM SOA approach focuses on the encoding of processes at the business level so that legacy applications involved in the process are service enabled along the way. The reason to do the service enablement is that it is a cheaper and faster route to leveraging legacy application within an overall BPM context. This is the case because of the huge technology investment in tools to service enable and in BPM to leveraging services.

Successful BPM and SOA projects (together or separately) are all dominated by a clear view of business value to be delivered. They are often preceded by a Proof of Concept which is constrained and focuses on key integration aspects (for service enablement) and key process aspects (which manifest business value).

Key to business value is the identification of value based on a clear understanding of the vision and mission of an enterprise coupled with strategic goals. This is the case because the strategic goals, which are derived from the vision and mission, provide clear guidance as to where the value lies. Identification of the strategic goals and the Key Performance Indicators (KPIs) that go with them enable a project to be framed so when delivered can be measured to determine its business value. The same KPIs and their

**Organizing and making things manageable takes effort and discipline but the rewards are often greater use and reuse because the focus of the service becomes clear and the concepts coalesce to deliver appropriate functionality for the level of abstraction.**

**Successful BPM and SOA projects (together or separately) are all dominated by a clear view of business value to be delivered.**

derivatives may then be used to monitor and measure the running processes and their attendant services and the delivered flexibility provides the means by which the solution can be tuned and adjusted. This ensures that the runtime governance of the solution always aligns the solution with the business goals through their expression in the KPIs.

## Closer to Home: The Meaning for Cognizant's Customers

The *Service Oriented Architecture Manifesto* and the clear linkage between service and process ORIENTATION embodies much of what Cognizant has done and continues to do. We have developed our own methodologies for service and process identification<sup>1,2</sup> to deal with granularity and business relevance. We have developed testable architecture<sup>1</sup> to ensure services do satisfy their business requirement and meet their goals, in the context of overarching processes. We provide enterprise architecture maturity assessments which look at the organizational alignment and ability to develop, consume and manage services.

SOAD typifies Cognizant's "continual innovation nothing less will do" approach to solving problems for our customers.

There has been much written about service granularity. Many companies have their own variant of a service identification methodology which is in keeping with *"Identify services through collaboration with business and technology stake-holders"*. Cognizant is no different in this regard, having developed Service Oriented Analysis and Design (SOAD<sup>2</sup>). What makes SOAD a little different is the way in which we continue to develop and learn from it and combine it with our enterprise architecture<sup>3</sup> and our testable architecture tools and methods. In many ways SOAD typifies Cognizant's "continual innovation nothing less will do" approach to solving problems for our customers.

Testable architecture is a method and a set of tools that we use to formally *"Verify that services satisfy business requirements and goals"*. This approach of using formalism, is unique to Cognizant as a global services provider. The benefits of using it enable a huge compression in the time it takes to gather requirements and deliver the technical artefacts (i.e., service contracts) that drive delivery of a service oriented solution as well as a reduction

in the overall Systems Development Lifecycle (SDLC) through reduced systems integration testing. The former happens because we can employ faster iterations (agile EA) in the gathering of requirements and the framing of a service oriented solution and prove that the solution satisfies the requirements at the touch of a button. The latter happens through the testing of the architectural model which reduces the number of iterations in systems integration testing. Testable architecture in the future now lives as an open source project called Savara<sup>4</sup>, which is jointly run by Red Hat and Cognizant.

Many in the industry provide service-oriented architecture maturity assessments, which we provide, but do so in the context of an enterprise maturity assessment<sup>3</sup>. This is an important distinction because if one focused only on service-oriented architecture, the organizational alignment that is required is much harder to achieve. We believe that the value statement, *"Business value over technical strategy"*, strongly supports our view of looking at the bigger picture through the eyes of enterprise architecture.

Enterprise architecture, which is a service that we provide in collaboration with customers, is all about the enterprise, hardly surprising given the name. And the enterprise is not just technology, it includes people, roles, organizational structure, processes and IT, holistically driven to strive for a vision, achieve a mission, and deliver against strategic goals. It ensures good governance and timely delivery with sound change management. This enterprise wide view fundamentally *"Respects the social and power structure of the organization"* and ensures that *"Strategic goals are valued over project-specific benefits"*. It directly supports the *"Evolution of services and their organization in response to real use"* and it requires the *"Establishment of a uniform set of enterprise standards and policies based on industry, de facto, and community standards."*

Pictorially we see the service-orientation and architecture in the context of enterprise architecture in the chart on the following page

The red lines show the scope of the tools and methods we use at Cognizant, although in the case of Savara this is aspirational and more akin to testable architecture today.



## About the Authors

*Steve Ross-Talbot is Chief Architect within Cognizant's Advanced Solutions Group in Europe and an author of the SOA Manifesto. He is a pioneer in distributed computing, founding father of WS-CDL and Testable Architecture, and an invited expert on many vertical standards from ISO to HL7. Steve can be reached at [Steve.Ross-Talbot@cognizant.com](mailto:Steve.Ross-Talbot@cognizant.com).*

*Shyam Chivukula is a Senior Architect and leads SOA competency within Cognizant's UK operations. Shyan is an acknowledged contributor to the SOA Manifesto, and has wide-ranging experience in delivering solutions using distributed and integration architectures. He can be reached at [Shyam.Chivukula@cognizant.com](mailto:Shyam.Chivukula@cognizant.com).*

*Dr. Bippin Makoond is a Senior Solution Architect at Cognizant who functions as Global Innovation Lead within the company's Banking & Financial Services practice. He holds three patents within the domain of wireless distributed systems and messaging technologies and is a visiting scholar and expert advisor to the Component and Distributed System Research Group (CODIS) at Kingston University. Bippin can be reached at [Bippin.Makoond@cognizant.com](mailto:Bippin.Makoond@cognizant.com).*

*Sanda Morar is a Principal Architect/Senior Manager within Cognizant's Advanced Solutions Group in Europe and has over 20 years experience working as an Enterprise/Solutions Architect and Consultant. She can be reached at [Sanda.Morar@cognizant.com](mailto:Sanda.Morar@cognizant.com).*

*Ricky Tapper is Director/Chief Architect within Cognizant's Advanced Solutions Group and heads the Architecture Center of Excellence. He has over 30 years of technology architecture and management consulting experience, as well as expertise in enterprise architecture and IT strategy. He can be reached at [Ricky.Tapper@cognizant.com](mailto:Ricky.Tapper@cognizant.com).*

*Nataraj Venkataramanan is a Principal Architect at Cognizant and leads the company's SOA Factory, which focuses on defining and developing SOA service offerings across various business verticals. He can be reached [Nataraj.Venkataramanan@Cognizant.com](mailto:Nataraj.Venkataramanan@Cognizant.com).*

*Paul Mukherjee is a Principal Architect at Cognizant and has worked on several large and complex IT programs across the globe. He can be reached at [Paul.Mukherjee@cognizant.com](mailto:Paul.Mukherjee@cognizant.com).*

## About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services. Cognizant's single-minded passion is to dedicate our global technology and innovation know-how, our industry expertise and worldwide resources to working together with clients to make their businesses stronger. With over 50 global delivery centers and more than 68,000 employees as of September 30, 2009, we combine a unique onsite/offshore delivery model infused by a distinct culture of customer satisfaction. A member of the NASDAQ-100 Index and S&P 500 Index, Cognizant is a Forbes Global 2000 company and a member of the Fortune 1000 and is ranked among the top information technology companies in BusinessWeek's Hot Growth and Top 50 Performers listings.

## Start Today

For more information on how to drive your business results with Cognizant, contact us at [inquiry@cognizant.com](mailto:inquiry@cognizant.com) or visit our website at [www.cognizant.com](http://www.cognizant.com).



**Cognizant**

Passion for building stronger businesses

### World Headquarters

500 Frank W. Burr Blvd.  
Teaneck, NJ 07666 USA  
Phone: +1 201 801 0233  
Fax: +1 201 801 0243  
Toll Free: +1 888 937 3277  
Email: [inquiry@cognizant.com](mailto:inquiry@cognizant.com)

### European Headquarters

Haymarket House  
28-29 Haymarket  
London SW1Y 4SP UK  
Phone: +44 (0) 20 7321 4888  
Fax: +44 (0) 20 7321 4890  
Email: [infouk@cognizant.com](mailto:infouk@cognizant.com)

### India Operations Headquarters

#5/535, Old Mahabalipuram Road  
Okkiyam Pettai, Thoraiipakkam  
Chennai, 600 096 India  
Phone: +91 (0) 44 4209 6000  
Fax: +91 (0) 44 4209 6060  
Email: [inquiryindia@cognizant.com](mailto:inquiryindia@cognizant.com)