Service Orientation

Winning Strategies and Best Practices

Companies face major challenges as they seek to flourish in competitive global markets, fueled by developments in technology, from the Internet to grid computing and Web services. In this environment, service orientation – aligning business processes to the changing demands of customers – is emerging as a highly effective approach to increasing efficiency. In this book, Paul Allen provides an accessible guide to service orientation, showing how it works and highlighting the benefits it can deliver. He provides roadmaps, definitions, templates, techniques, process patterns and checklists to help you realize service orientation. These resources are reinforced with detailed case studies Queensland Transport and Credit Suisse.

Packed with valuable insights, the book will be essential reading for CIOs, IT architects, and senior developers. IT facing business executives will also benefit from understanding how software services can enable their business strategies.

For the last 40 years, business has viewed IT as its principal tool to increase productivity, and rightly so. Industries as diverse as retailing and financial services have been transformed by technology, especially when used in a networked way. Yet very often, IT has constrained the evolution of those business processes by imposing rigid technology implementations on them. And even worse, as IT has grown and become more important, it has become more complex and difficult to manage, especially in larger enterprises. In some cases the sheer complexity of the IT environment has started to limit the ability of companies to innovate. However, as Paul Allen shows, recent advances in IT processes and technologies, especially service-oriented architectures are starting to free companies up from these constraints, and allow them to once again align their IT processes in support of their business processes. *Service Orientation: Winning Strategies and Best Practices* charts the course for handling what promises to be as profound a change to IT as any that has occurred in the last few decades.

John A Swainson
President and CEO
CA International

Paul Allen is the principal business–IT strategist at CA (www.ca.com), and is widely recognized for his innovative work in component-based development (CBD), business–IT alignment, and service-oriented architecture. Paul's detailed knowledge combines with a uniquely practical understanding of the problems companies face as they apply new technologies in search of business value. His pragmatism stems from over thirty years' experience in the management and development of large-scale business systems. His current focus of interest is in helping organizations' transition to service orienation.

Paul is a widely published author and has held the position of editor of Cutter Consortium's *Component Development Strategies*. He is also a popular speaker at industry conferences worldwide. His co-authored book *CBD for Enterprise Systems* was one of the first and most practical descriptions of what was involved in building component-based applications.

Sam Higgins is now with Forrester Research Inc. (www.forrester.com) as a senior analyst. Formerly he was a director of Encode Services, an Australian technology consulting firm, and managed the Innovation and Planning Unit of Queensland Transport's Information Services Branch. He has extensive knowledge of achieving winning strategies in service orienation.

Paul McRae is the application architect within the Innovation and Planning Unit of Queensland Transport's Information Services Branch. He has a wealth of experience in developing best practices in service oriented architecture.

Hermann Schlamann is a senior architect in the architecture group of Credit Suisse. Since starting his career in 1971, he has built a deep and extensive understanding in the practicalities of using methodologies to produce successful software.

Service Orientation

Winning Strategies and Best Practices

Paul Allen

with Sam Higgins, Paul McRae, and Hermann Schlamann

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9780521843362

First published 2006
Reprinted 2008

A catalogue record for this publication is available from the British Library

ISBN 978-0-521-84336-2 Hardback

Contents

PART 3 SERVICE-ORIENTED ARCHITECTURE

PART 4 SERVICE-ORIENTED MANAGEMENT

PART 5 CASE STUDIES

Foreword

In the broader scheme of things our use of computer systems is still in its infancy. In just fifty years we have made amazing progress, to the point that computer systems are essential to the smooth running of home and business. Yet any systematic or casual measurement of user satisfaction with enterprise systems will inevitably return unsatisfactory ratings. The problems are legion but, with monotonous regularity, surveys report that users believe that total costs of ownership are too high and systems are insufficiently responsive to constantly changing circumstances.

The root problem is that insufficient attention has been paid to the fundamental structure of software systems. In other engineering disciplines such as automotive, construction, or electronics, there is unambiguous understanding between the user requirements and the delivered product, such that the various stakeholders can engage with the architect in a highly meaningful manner. For example, the architect and designer of the automobile, bridge or silicon chip has well-understood responsibilities to optimize the design and economics by using standard patterns and components which deliver a product that has a well-understood life cycle from production engineering through to retirement.

In contrast, the typical software system has no coherent structural and economic model that governs its life cycle. Rather, the approach to acquiring software systems is usually a mixture of blind faith and optimism. Consequently the Chief Information Officer (CIO) role has become a high-risk career and the typically brief time in the position encourages short-term thinking, in contrast with the real requirements of enterprise systems which are always medium to long term. The result is that most enterprises buy their software systems from enterprise solution providers because it moves the responsibility to an external party. However until very recently the enterprise software providers have been unable to offer any better solutions than an enterprise might develop themselves, and the resulting chaos from ill-fitting systems being shoehorned into unsuitable situations has led to widespread dissatisfaction. This deep discomfort is often expressed as an urgent need for more adaptable systems.

Since the mid-1990s, many companies have worked to solve these problems by introducing *middleware* – technical software systems that allow disparate systems that were never intended to work together to cooperate at a fairly basic level. However the

costs of integrating systems in this way has proved to be prohibitive, so that many information technology (IT) organizations report that 80% of their entire budget is consumed by this inherently unproductive activity.

Software engineering is in many ways very similar to automotive, construction and electronic engineering. While the resulting software product at an aggregate level is much less well defined and generally less stable, at what might be termed the "sub-assembly level" there is almost always a high level of stability. This suggests that the formal concepts of componentization, widely used in other engineering disciplines, is entirely appropriate.

Consider a personal computer (PC) system. The typical PC comprises many thousands of individual components that are clustered together into a manageable number of sub-assemblies. Each of these assemblies is linked to one of several *bus structures* that provide communications between the major functions such as the processor and disks. Each of the components and assemblies offer services – for example, a disk access service. These services are offered under an agreed service contract between the components. The underlying capability is offered purely on the basis of the service interface, and by design the consumer of the service should not need to know how the service is implemented. This encapsulation, together with stability of contract, allows many different parties to collaborate in the overall product, enabling the computer seller to create a highly optimized supply chain. Even more important, the seller can constantly evolve the overall specification of the end product, substituting parts and assemblies to respond to changes in the supply chain and to deliver improved capabilities to market.

In the long term, the existing monolithic structure and economic model underlying enterprise systems delivery was unsustainable. The software industry recognized that the inherent structural problems needed to be addressed through *industry standards* which would, over time, facilitate a fundamental reconfiguration of software products. The twin ideas of middleware and software components and services have been the basis of an unprecedented initiative by industry leaders to create standards for service interoperability. This technical backplane, referred to as Web services specifications, is today rapidly maturing and while it will take some years to become fully mature, is now in widespread use throughout industry.

The service backplane for enterprise systems exactly parallels the service bus in the personal computer. Business relevant services such as order taking and credit checking can be constructed as independent components and collaborate with other services such as master data, using formal contracts that govern their respective responsibilities. For the first time, the internal technical representation of systems functionality is exactly the same as the business perspective, providing enormous opportunity to establish common understanding between all stakeholders. Perhaps even more important, the services are implemented as an invokable resource that offer real-time, or near-real-time services, which enable software systems to mirror real-world activity. Like the personal computer the business components and services encapsulate the underlying

implementation complexity and can be substituted, modified, and upgraded with a well-understood impact on other collaborating components and services, creating a constantly evolving information system that can more closely reflect business reality.

In the beginning, we talked about this approach as a service-oriented architecture or SOA. Not surprisingly, because the genesis of the concept is about fundamental restructuring of software systems, to break up monolithic designs into collections of moving parts that can be managed in a similar way to an automobile or PC. As our thinking matured we realized that the architecture was merely one discipline of a much broader approach that requires significant change right across the software supply chain and enterprise organizations impacting organization structures, business processes, and project management techniques as well as software architectures. We call this *service orientation*.

This book is important because it addresses the subject in depth. It provides a comprehensive and detailed treatment, probably for the first time, of the wider business and technology impacts of service orientation.

David Sprott
Founder and CEO, CBDI Forum

Preface

Why this book?

We live in a world of change. During my thirty-plus years in the software industry, I have witnessed some remarkable changes. One constant, however, seems to me to be that successful companies cope with change by nurturing their people, learning fast, building on what they have learned, and discarding things that no longer work. Surely, that's not rocket science! If that's the case then, "Why," I ask myself, "write a book about it?" It seems appropriate to begin by offering the reader some kind of answer!

Even moderately large companies can no longer afford to "just get by" as they seek to cope with the challenges of a world of unforgiving change. Business is increasingly moving toward a marketplace model. This model is in sharp contrast to the traditional view of an organization as a production line. In this world, organizations collaborate together, consuming and offering services to maximize efficiency, better serve customers, and achieve long-term advantage.

This is the world of service orientation, in which a hasty decision to outsource an activity may well be a decision that the firm repents at leisure. Equally, focusing the company's own energies on the wrong kinds of thing could be a disaster. The power of technology, from the Internet to grid computing and Web services, is the irresistible force that is fueling this change. Business is increasingly less supported by software and more enabled by it. This means that simply automating business activities in software is no longer enough. The software has to be agile enough to cope with change and foster innovation.

The really successful companies are companies that seem somehow to be acutely aware of these things. This is in some contrast to those that struggle! These organizations have, to a large degree, lost control over their software and the business processes that depend on that software. In domestic terms, it's like a mother that spends all her time cleaning her own windows while taking no interest in her children!

So what can we do about this?

A threefold response

- *First*, *improve business processes* to reflect the reality of service orientation, with an appropriate gearshift in approach to business architecture and process redesign.
- *Second*, *nurture software services* as the common threads between business and IT. Service-oriented architecture is the discipline that aims at reducing the risks posed by change by structuring an agile portfolio of software services.
- *Third*, introduce *effective control of the execution of software services* that extends their reach beyond the firewall to more and more external users and partners. "Service-oriented management" is the emerging discipline designed to monitor and control the live, executing software that transcends the old traditional boundaries.

The focus of the book

These three areas are often treated independently, resulting in a fragmented approach. This compartmentalizing of concerns is deeply ingrained in the IT psyche. One of my main aims in writing this book has been to tackle this issue by providing an holistic approach in which all three disciplines are integrated.

Most of the technologies that are enabling service orientation are developing out of previous well-established generations of technology. The long-term trend is toward greater and greater standardization of the interfaces through which software systems talk to each other.

The great paradox that we are faced with today is that developments in technology are rendering specific technology issues less and less important, but are hugely magnifying the potential scale and complexity of our business. At the same time, these developments promise greatly improved return on investment from utilization of technology in the form of on-demand computing – buying just enough software services as and when you need them – "by the drink" instead of "by the case," if you will.

The standards that are enabling these trends (specifically Web services) are both complex and constantly evolving. Rather than attempting the impossible task of reviewing and documenting all of these technologies and standards I have aimed at providing a badly needed technology agnostic approach to best practices, designed for maximum resilience to the pace of change.

The audience of the book

While this book is not a panacea, my hope is that it will at least provide you with some equipment (roadmaps, definitions, templates, techniques, process patterns, and checklists) to help you on your journey to service orientation. If you have already

embarked on that journey, or are faced with making it, you are likely to be a CIO, IT architect or senior developer working in a large or medium-sized business. Equally, the book should be relevant to a second group: business executives with an interest in, and responsibility for, IT and managers who need to understand how to plan and develop cost-effective strategies to cater for service orientation. Thirdly, by clarifying concepts that have often been a source of past confusion, the book should be of value to teachers and students of both IT and business studies.

The structure of the book

The book is organized in five parts.

Part 1 sets the context for the remainder of the book by explaining the principles of service orientation, as a business phenomenon enabled by developments in technology.

Part 2 provides an approach to business architecture that is based on evolution of best practices and in tune with the need to provide useful business process improvements quickly.

Part 3 lays out the two major aspects of the service-oriented architecture: policy and structure. We explain how to achieve an integrated approach that is driven "top-down" by the business architecture, and equally that is supported "bottom-up" by service-oriented management.

Part 4 introduces a new discipline, service-oriented management (SOM), to tackle the run-time challenges of software services in tune with both business architecture and service-oriented architecture (SOA).

Part 5 provides two real-life case studies in service orientation that illustrate the *execution* of an adoption strategy.

Winning strategies and best practices

Although plans are essential, an organization's willingness and ability to execute its plans are the hallmarks of a *winning* strategy. The book is therefore slanted toward practical guidance above theory and recognizes that the approach that your organization chooses to take will depend on a range of factors, from its target business model to the state of its legacy systems. Pragmatic models that capture valuable organizational knowledge must be nurtured as a long-term investment, so that this knowledge does not walk out the door every time that the CIO or any of her software architects leaves the company for another job!

The case studies are particularly important for understanding how to execute the winning strategies and best practices contained in the book.

In chapter 14, Sam Higgins and Paul McRae describe how Queensland Transport developed a service-oriented approach that is used to allow third-party software developers to integrate regulatory processes seamlessly within commercial motor vehicle dealer business processes. In chapter 15, Hermann Schlamann describes how Credit Suisse uses service orientation to enable money transfers for customers over different channels, and to provide various foreign exchange and accounting services to the smaller banks.

Both case studies are the result of ongoing improvement programs that have successfully applied the core techniques explored within this book. They both show how architectures and models can be nurtured, with a focus on getting the most out of your existing software assets, so that service orientation becomes integral to the organization's business strategy.

"You can't measure what you don't specify"

In my visits to many organizations and in my discussions with hundreds of colleagues and professionals a constant theme emerges: there is a resounding need for clear guidance on service-oriented concepts, unfettered by detailed technical jargon and unwieldy procedures. There seem to be no end of processes and procedural guidance available, but relatively little in the way of plain common sense approaches to specification, particularly specification of services. Over twenty years ago Tom DeMarco (1982) told software project managers with his usual elegance that: "You can't control what you don't measure."

As a software project manager at the time, those words have made a lasting impression on me. So much so that one of the main messages that you will find repeated again and again in this book is that: "You can't measure what you don't specify."

If there is one thing above all else that I would like the you to take away from this book it is that simple, yet critical, message.

Acknowledgments

Many of the ideas in this book have been shaped by previous work done with many client organizations and individuals too numerous to name individually. While I extend my thanks to them all there are some individuals that I would also like to thank personally.

Much of the thinking behind the book emerged through the ebb and flow of discussions at CA with John Dodd (now an independent methods architect). These were sometimes passionate, never dull! I am particularly thankful to John for being at once my strongest critic yet greatest ally, for doing his best to keep my feet on the ground, and for his insightful reviews. Bruno Lefever and Danny Saro also deserve thanks for the part that they played in these discussions.

I also thank Bruno Lefever, Michael Stephenson and Janique Vandekerkhove of Computer Associates for their review comments. Champion reviewer honors however go to Danny Saro for his extremely thorough and often downright annoying comments. It is perhaps ironic that an Englishman is thanking a Belgian for keeping him true to his own language!

I extend my thanks to the case study contributors: Sam Higgins and Paul McRae of Queensland Transport, and Hermann Schlamann of Credit Suisse. Their influence on me stretches much further than the case studies, in that all have worked closely with me, during the progress of this book, in both discussion and through reviews.

Thanks are also extended to Art Sedighi, solutions architect, of Platform Computing for helping to guide my thinking and for his review comments.

Credit is also due to the following individuals who have stimulated or educated my thinking in various ways that have played a part in shaping this book: Jason Bloomberg, John Daniels, Paul Harmon, Richard Solely, David Sprott, Borys Stokalski, Richard Veryard, Tom Welsh, and Lawrence Wilkes.

I also thank the management team at CA for their support in enabling me to write this book. In particular, I would like to thank Colin Bannister for his help and encouragement.

Thanks are also extended to my editor at Cambridge University Press, Emily Yossarian.

Last but not least, I thank my wife and children for their patience with my distractions while writing this book, and above all for helping to keep me sane.

Acronyms and abbreviations

AHS	American Hospital Supply
APS	Analytic Systems Automated Purchasing
API	application program interface
APPo	Application Portfolio
APQC	American Productivity and Quality Centre
ASAP	aligning services and priorities
ASP	application service provider
AST	agreed service time
ATM	automatic teller machine
B2B	business-to-business
BA	business architecture
BAM	business activity manager
BCM	business capacity management
BIAT	business–IT alignment table
BLA	business-level agreement
BPEL4WS	Business Process Execution Language for Web Services
BPM	business process management
BPMI	Business Process Management Initiative
BPMN	Business Process Modeling Notation
BPR	business process re-engineering
BRM	business rules manager
BSB	business service bus
CBD	component-based development
CEO	chief executive officer
CEP	complex event processing
CIO	chief information officer
CMDB	configuration management database
CMMI	Capability Maturity Model® Integration
CRC	class responsibility collaborator
COBOL	Common Business Oriented Language
COM	Component Object Model

COM+	an extension of COM
CORBA	Common Object Request Broker Architecture
CRC	class responsibility collaborator
CRM	customer relationship management
CRODL	create, read, update, delete, and list
CS	Credit Suisse
CIS 3.1	Component Standard 3.1
DAIS	Dealer Agency Interface System
DIS	Dealer Interface System
DCOM	Distributed Component Object Model
DSDM	Dynamic Systems Development Methodology
DMTF	Distributed Management Task Force, Inc.
EAI	enterprise application integration
EDI	electronic data interchange
EFQM	European Foundation for Quality Management
EJB	Enterprise JavaBeans
ERP	enterprise resource planning
ESB	enterprise service bus
eTOM	enhanced Telecom Operations Map
EU	European Community
FIOM	Financial Instruments Object Model
FTP	File Transfer Protocol
GGF	Global Grid Forum
HR	human resources
HTTP	Hypertext Transfer Protocol
IDL	Interface Definition Language
ISB	infrastructure service bus
ISO 9000	International Organization for Standardization 9000
IT	information technology
ITIL	Information Technology Infrastructure Library
it SMF	IT Service Management Forum
J2EE	Java 2 Enterprise Edition
JAP	Java Application Platform
JIT	just-in-time
JMS	Java Message Service
KPI	key performance indicator
M & A	mergers and acquisition
MDA	Model Driven Architecture
MOM	message-oriented middleware
MOWS	management of Web services
MSMQ	Microsoft Server for Message Queuing

MTBF	mean time between failures
MTTR	mean time to repair
MUWS	management using Web services
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Office of Government Commerce
OGSA	Open Grid Services Architecture
OLA	operational-level agreement
OMG™	Object Management Group
OO	object-oriented
PC	personal computer
QC	quality check
QoS	quality of service
QT	Queensland Transport
R & D	research and development
RAEW	responsibility, authority, expertise, work
RCM	resource capacity management
RMI	Remote Method Invocation
ROI	return on investment
SAML	Security Assertion Markup Language
SBS	Services Booking System
SCOR	Supply Chain Operations Reference
SCM	service capacity management
SDB	Service Database
SDLC	systems development lifecycle
SEI	Software Engineering Institute
SDM	software development methodology
SEM	service execution management
SFTP	Secured File Transfer Protocol
SLA	service-level agreement
SLM	service-level management
SOA	service-oriented architecture
SoS	specification of service
SOT	security outage time
SOV7	seven service-oriented viewpoints
SOAP	Simple Object Access Protocol
SOM	service-oriented management
SSL	Secure Sockets Layer
SSO	single sign-on
TAC	Technical Advisory Committee
TC	technical committee
TCO	total cost of ownership

TIA	technical infrastructure architecture
TRAILS	Transport Registration and Integrated Licensing System
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modeling Language
W3C	Worldwide Web Consortium
WS-CDL	Web Services Choreography Description Language
WS-I	Web Services Interoperability Association
WAP	Wireless Application Protocol
WS80	Wertschriften System Asset Management
WSBPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
WSDM	Web Services Distribution Management
XML	extended mark-up language
xP	eXtreme programming