SYSPRO

on SOA

A GUIDE TO SERVICE ORIENTED ARCHITECTURE

Sean Wheller

SYSPRO™

Simplifying your Success

SYSPRO on SOA

A Guide to Service Oriented Architecture

Sean Wheller
SYSPRO (PTY) Ltd.

SYSPRO on SOA: A Guide to Service Oriented Architecture

by Sean Wheller and SYSPRO (PTY) Ltd.

Published 2006

Table of Contents

Preface xiii
Who should read this book xiii
About this Book xiv
Acknowledgments xv
SYSPRO Enterprise Software xvi
Feedback xvi
I. Overview 1
1. Introduction 3
2. Background 7
1960s-Pre-Computer Era 7
1970s/1980s - Advent of Computers in Manufacturing 8
MRP - The Initial Impact 8
CRP - The Next Development 9
MRPII - Connecting Manufacturing and Finance 10
3. Computing Today 13
1990s - Enterprise Resource Planning 13
ERP Defined 14
Impact of the Personal Computer 14
Evolution of ERP to ERPII 15
II. Extended Enterprise 17
4. Business Drivers 19
Globalization 19
Flexibility and Agility 21
Real Time Information 23
Pro-active Decision Making 25
Conclusion 25
5. Connecting all the Pieces 27
Business Processes 27
Transactions 29
Intelligence 29
Collaboration 30
Business Information 31
Business User 32
Making the Connections 33
6. Service Orientation 35
Introduction to SOA 35
SOA Defined 36

Quick History of SOA 37
Key Concepts in SOA 39
Services 39
Loose Coupling 42
Interoperability 44
Dynamic Discovery 45
Front Ends 46
Message Based Technology 47
Distributed Network Architecture 48
Web Services 50
From Key Concepts to Supports 51
7. SOA Supporting Concepts and Technologies 53
Poles, Pegs and Ropes 53
XML 54
XML Schemas 56
Enterprise Software Bus 58
WSDL 60
SOAP 61
Grid Computing 62
Ethernet 66
Broadband 66
GSM 67
RFID 68
Client Front Ends 70
A Whole Tent 71
8. Benefits of SOA 73
Introduction 73
The Business Value of SOA 73
Agility 74
Better Return on Investment 76
Service Assembly 77
Lower Development Costs 77
Future Proofing 78
Better Scalability 79
Improved Business Alignment 79
Improved Customer Satisfaction 80
Summary 80
The IT Benefits of SOA 81
Code Mobility 81
More Code Reuse 82
Focused Developer Roles 82
Better Parallelism in Development 83

Better Testing/Fewer Defects 83
Support for Multiple Client Types 84
Better Maintainability 84
More Security 85
Summary 85
Assembly Without Programming - The Composite Application Platform 86
Dynamic Meta Data Repository 88
Refining Services 88
Defining Interoperability 89
Conclusion 89
9. Challenges of SOA 91
Introduction 91
Technical Challenges 91
Data Rationalization 92
Business Service Enablement 93
Abstraction Incompatibility 94
Service Accessibility 94
Constrained Innovation 95
Standards Interoperability 95
Business Challenges 96
Transfer Pricing Model 96
Ensuring Customers Service Levels 97
Consequences of Change 97
Ownership Issues 98
Roadblocks Ahead 98
The Build vs. Buy Argument 99
Background 100
Buy (Packaged Solution) 100
Build (Custom Application) 102
Somewhere In Between 103
Conclusion 105
10. Security in SOA 107
Security Concerns 107
Trust 108
Message Security 108
Distributed Policies 109
Identity Management 110
Interoperability 110
Current Web Based Security Standards 111
SOA Security Summary 112
11. Summary of SOA 113

III. SOA Models 115
12. Demand Driven Supply Networks 117
Introduction to DDSN 117
Collaborative Supply-Chain 118
Synchronized Swimming 119
Demand Management 119
Supply Excellence 120
Continuous Innovation 120
DDSN Capability Model 120
13. Collaborative Planning, Forecasting and Replenishment 123
Introduction to CPFR 123
The CPFR Model 124
Strategy and Planning 125
Demand and Supply Management 125
Execution 126
Analysis 126
IV. Implementation 129
14. Introduction 131
15. Implementation Methodologies 133
Technologies and Tools 133
Methods 135
16. Business Models 139
17. Planning 141
Examining the Impact of Change 141
Logical Migration 142
Technical Migration 143
Web Services Management Platform Requirements 144
Service Continuity - Improving and Keeping Everything Going 146
Align 147
Comply 147
Observe 148
Respond 148
Optimize 149
Achieving SOA Command and Control 149
Achieving Integration 150
User Interaction Integration 151
Business Process Integration 152
Application Integration 153
Data Integration 154
Choosing the Right Integration Solution 154
Data Transformation and Rationalization 154
Architecting an Effective Solution for your Enterprise 159

18. SOA Adapting to Change 163
Increased Enterprise Responsiveness to Change 163
Increased Responsiveness to Market Change 165
Enterprise SOA - Large, Dynamic and Rife with Interdependencies 170
The Impact of Change within a SOA 170
Unexpected Change to a Networked Service 171
Example of the Impact of Change 173
Planned Change to a Networked Service 180
Planned Simultaneous Change to Many Networked Services 182
The Ultimate Impact of Change = Skyrocketing Costs 184
Web Services Management Platform Requirements 188
Architecting a Solution 191
Service Deployment Action 194
The Solution in Action 199
SOA Implemented 202
V. Applications 205
19. Introduction 207
20. SYSPRO e.net solutions and the Extended Enterprise 209
Product Description 209
Extending the Enterprise 210
Extending Functionality 210
Extending Boundaries 211
Custom Application Development 211
Enterprise Application Integration 211
Electronic Commerce 212
Collaborative Commerce 212
Collaborative Private Exchanges 212
21. Intra-Enterprise 213
22. Inter-Enterprise 215
VI. SYSPRO - In Action 217
23. Introduction 219
24. Dewhurst plc. 221
The Situation 221
Planning Stages 222
What a Difference! 223
SOA Summary 226
25. Lakeshirts, Inc 227
The Situation 227
Solution 228
Action Results 228
The New System 229

26. Union Carriage and Wagon 231
UCWs' Need 231
Existing Conditions 233
Planning 234
Execution 236
Results 239
27. Bendalls Engineers 241
The Company 241
The System 242
The Results 243
28. Cedarlane Laboratories Limited 245
The Company 245
SYSPRO Scalability 245
SYSPRO SOA Solution 246
SOA Results 247
Bibliography 249

List of Figures

1.1. Evolutionary Stages toward SOA 4
5.1. Connecting People, Processes and Information 28
6.1. CD Player as a Service 36
6.2. Charge Credit Card Service (Verify Customer Credit History) 40
6.3. Artificial dependency - different power plugs and sockets 42
6.4. Loose coupling advance in Christmas Lights 43
6.5. Message based technology 47
6.6. Message package 47
6.7. Connections within a Distributed Network Architecture 49
7.1. An ESB translating messages 60
7.2. Different parts of a process within a Grid 64
8.1. Combining individual services into a composite application 87
13.1. Framework of the CPFR Model 124
13.2. Strategy and Planning Tasks 125
13.3. Demand and Supply Management 126
13.4. Execution 126
13.5. Analysis 127
16.1. The Enterprise Framework 140
17.1. The 5 View Model 160
17.2. Practice of SOA 161
18.1. From monolithic to loosely-coupled, simpler, modular software components 164
18.2. Initial project built with Web Services: Clearing excess inventory 166
18.3. Customers are quickly provided direct ordering access through e-commerce 167
18.4. Web Services leveraged across the enterprise to power real-time business activity monitoring 168
18.5. New services and applications continue to be built, enhanced and connected 169
18.6. First Impact 173
18.7. Second Impact 174
18.8. Third Impact 175
18.9. Fourth Impact 176
18.10. Fifth Impact 177
18.11. Sixth Impact 178
18.12. Failures appear to be random and unrelated 179
18.13. Sources of change in an enterprise service network 185
18.14. Running and evolving an unmanaged enterprise service network 186

18.15. Skyrocketing costs and complexity of an unmanaged service network 187
18.16. Service brokers, Web Service providers and a centrally managed policy 191
18.17. Alternate deployment option placing Active Agent on Web Service provider 192
18.18. Service brokers and active agents deployed in support of initial project 195
18.19. New application components link to existing services 196
18.20. Successive project deployments 197
18.21. The effective Web Services Management Platform 198
18.22. Service broker intermediates provider and consumer interactions 199
18.23. Non-disruptive application or service replacement 200
18.24. Web Services management platform automates reactions 201
18.25. Web Services Management Platform lowers the cost curve 202
26.1. UCW Project Timeline 237

Preface

Welcome to *SYSPRO on SOA*. This book is a guide to what is probably one of the most profound and exciting changes in the history of information technology; Service Oriented Architecture (SOA).

The subject of SOA is as endless as there are technologies to implement. For every expert on the subject you will find a slightly different perspective thereon and twenty others who disagree in whole or in part. Perhaps the reason for this is the abstract nature of Service Oriented Architecture. Perhaps the reason is that there can be as many permutations of what a Service Oriented Architecture should be, could be, would be, as there are companies whose IT systems must evolve to expose SOA capabilities.

Despite expert differences, software as a service is beginning to take hold.

Who should read this book

While industry experts will surely find much educational value in this book, the primary audience in mind while writing this book is non-expert.

SYSPRO on SOA aims to explain SOA in the simplest terms possible. Rather than explain SOA from an IT perspective, which most books on the subject do, the SYSPRO on SOA approach is from a business perspective. It is this focus on business process that makes SYSPRO on SOA unique amongst other books on the subject.

If you are a business manager or business process engineer, SYSPRO on SOA will help you understand the concepts of SOA and the benefits thereof from the business perspective.

While SYSPRO on SOA is a book written by SYSPRO, the platform and technology neutral, heterogeneous nature of SOA means that most of the knowledge gained from this book will remain applicable in most business computing environments.

As one of less than a handful of books on the subject, SYSPRO on SOA is a valuable addition to the library of any person having to evolve IT-capabilities and align them with the future.

About this Book

This book is divided into six parts, corresponding to particular topic groups. The information in each part is intended to be read in order, each part and chapter building on the one before it. It is also possible to use this book in a non sequential order, as a reference.

At the start of each part an abstract is provided to give you an idea of the information contained within the part. If you find that you are already familiar with a topic, you can skip ahead to the next heading. When skipping sections, keep in mind that knowledge of specific concepts relayed in previous sections may be assumed.

- Part I, “Overview” [1] - traces the relationship between business and IT so you will better understand the development of business software from materials requirement planning (MRP) to full blown extended enterprise resource planning (ERPII) systems and the move toward Service Oriented Architectures within the enterprise software framework.
- Part II, “Extended Enterprise” [17] - is devoted to a discussion of the business drivers that are currently affecting business needs and IT solutions that are encouraging companies and enterprises to examine the concept of SOA to explore what will benefit them by adopting it.
- Part III, “SOA Models” [115] - presents two model applications of the SOA concept: Demand Driven Supply Networks and Collaborative Planning Forecasting and Replenishment. These are discussed with definitions, examples and detailed business implications.
- Part IV, “Implementation” [129] - discusses the process of evaluating and creating Service Oriented Architectures and provides steps and models that can be followed along the way.
- Part V, “Applications” [205] - explains some systems of implementation and management we now look briefly at SYSPRO applications in relation to SOA. You will find a product description, as well as discussion on the extended enterprise, intra-enterprise relationships and inter-enterprise relationships.
- Part VI, “SYSPRO - In Action” [217] - contains five real life installations of SYSPRO are presented to demonstrate SYSPRO's system as the core of a Service Oriented Architecture in these different enterprises.

Acknowledgments

Writing a book is always a collaborative exercise in crafting words and molding collective ideas into pages. This said, InWords [http://www.inwords.co.za] would like to acknowledge the people who have helped in development and production of SYSPRO on SOA.

Much thanks goes to Graham Goode, member of the InWords team, who has done a incredible job of assisting me in producing this manuscript. Graham has been instrumental, constantly chipping away and sorting through a plethora of information, rewriting what was verbose and not clear and constantly remaining open to new ideas and the addition of perspective. Thanks Graham, working with you has been great, really a fire and forget experience.

This is the third book I have written with SYSPRO and for the third time, I must extend my gratitude to Stanley Goodrich from SYSPRO U.S.A [http://www.syspro.com/us/]. Each time we write a book Stan steps forward to lend his most excellent and rare editing skills to the project. Editing a manuscript this size, in short time, is not easy. Thanks Stan, I hope we can look forward to support from yourself and SYSPRO U.S.A on future projects.

Thanks goes to our reviewers Jeremy Hart and Howard Joseph from McGuffie Brunton Ltd. [http://www.mcguffie.co.uk/], Dale Kehler from Syspro Business Solutions Inc. [http://www.syspro-solutions.com/] and Hellen Hollick from SYSPRO South Africa [http://www.syspro.com/za/].

We would also like to thank the following SYSPRO customers for their permission in publishing case studies on their implementations.

- Dewhurst plc.
- Lakeshirts
- Union Carriage and Wagon
- Bendalls Engineering, a division of CARR'S Engineering
- CedarLane

SYSPRO Enterprise Software

SYSPRO is an internationally recognized, leading provider of enterprise business solutions. Formed in 1978, SYSPRO was one of the first software vendors to develop an enterprise resource planning solution. Today, SYSPRO is a global business solutions vendor with offices on six continents and over 1500 channel and support partners. Over 12,000 licensed companies across a broad spectrum of industries in more than 60 countries trust SYSPRO as the platform on which to manage their business processes.

By focusing on people and building lasting relationships with customers and partners, SYSPRO consistently excels at guiding customers through all aspects of their implementation. Tried, tested and reliable, SYSPRO has stood the test of time as a company and as a software solution.

Drawing on its heritage, SYSPRO's vision is focused on delivering customer needs today and in the future. The company's mission is to continually develop remarkable software that simplifies operational effectiveness and keeps customers in control of their businesses.

As a customer-centric company, SYSPRO aims to deliver world-class software that gives customers the control, insight and agility they need for competitive advantage in a global economy. Customer focus is a core component of the corporate culture and continues to be one of the key reasons why SYSPRO maintains a strong leadership position in the enterprise application market.

For more information about SYSPRO visit the company web site [http://www.syspro.com].

Feedback

As the reader of this book you are our most important critic and commentator. We value your opinion and want to know what we are doing right and wrong. We want to hear your ideas on how we may improve.

You can email your feedback to <publications@za.syspro.com.>

Part I. Overview

By the end of this part of the book, you will have seen the development of the relationship between computing and businesses from the early days of computers through to the current interconnected relationship. In tracing this relationship you will better understand the development of business software from materials requirement planning (MRP) to full blown extended enterprise resource planning (ERPII) systems and the move toward Service Oriented Architectures within the enterprise software framework.

Chapter 1. Introduction

> "Faced with the choice between changing one's mind and proving that there is no need to do so, almost everybody gets busy on the proof."
>
> —John Kenneth Galbraith, OC, Ph.D., LL.D, twentieth-century economist.

Hindsight is foresight with revisions. What would we give for the chance to accept or reject decisions made? What would we give for the chance to accept or reject decisions *not* made? Given the ability to review our world as a set of revisions, would we accept or reject the changes we have or have not made? Would we use this hindsight as foresight? Considering the amount of resistance we put up in response to daily changes around us, many would argue that we would do what we do naturally, resist the change.

When resisting change, we knowingly or unknowingly behave in ways that attempt to keep things 'the way they were.' Between the worlds where business and IT meet, we time and again, find that we need to help ourselves move from resistance to acceptance of change. We frequently resist change for one or all of the following reasons:

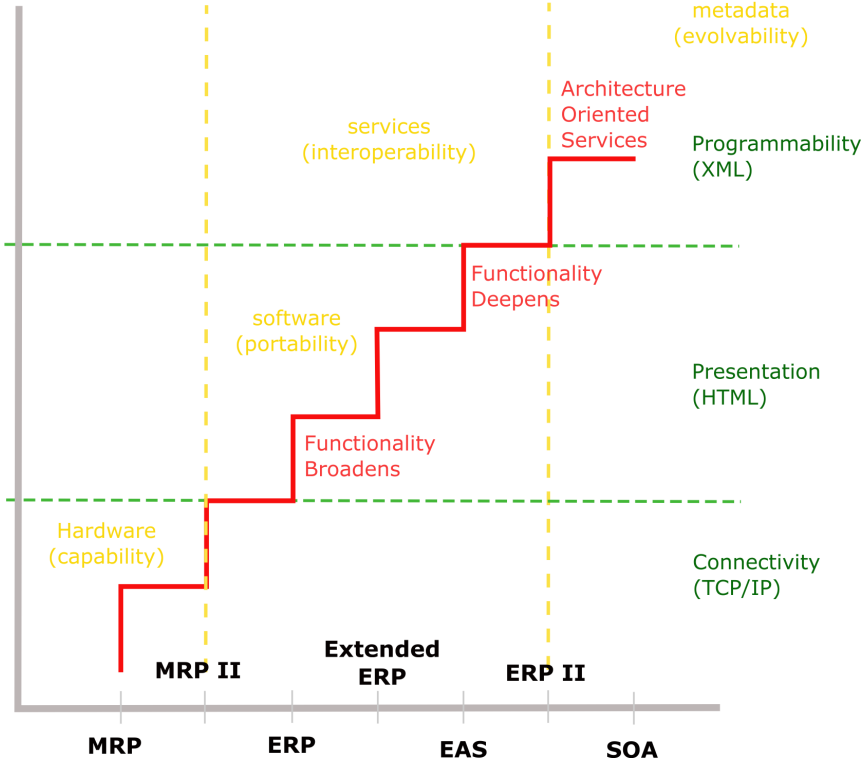
- We do not understand and therefore fear.
- We will lose something and don't want to let go.
- We know it works and see no reason to fix it.

What does all this have to do with SOA?

Well, the fundamental philosophy behind SOA is all about change and the ability of business to evolve IT-capabilities with greater flexibility and agility. Before learning anything about SOA we must be comfortable in our own ability to accept change. The design philosophies of SOA embrace change as inevitable. SOA recognizes that resisting change is like holding your breath - If you succeed you die. Embracing change is therefore the only acceptable course of action and architecture based on SOA are designed for change. This belief, coupled with an understanding of how enterprise applications have evolved, why your systems are they way they are, form important background information that will greatly assist any person planning for SOA. For this reason, we have provided some historical information on the evolutionary stages enterprise systems have been through as they moved close to

SOA. The evolutionary stages are depicted in Figure 1.1, “Evolutionary Stages toward SOA” [4].

Figure 1.1. Evolutionary Stages toward SOA



From this picture it is plain to see that the important part about evolution is its intention. Each milestone in the evolutionary process happens because somebody or something, perhaps our collective consciousness, had a vision and the intention is usually positively focused. In the bigger picture evolutionary changes do not take place in a vacuum. Often, before a technology can move into mainstream use it requires the evolution and maturity of other technologies and even our collective understanding.

Take for example the bigger picture of the enterprise application evolution. During the period of MRP and early ERP, the main ingredient that needed to advance or

evolve was the hardware used to run such systems. Cumbersome and expensive, hardware restricted the evolution of enterprise applications from wide adoption. Fortunately driven by developments in telecommunications and TCP/IP connectivity, hardware was forced to evolve and eventually matured. Who would have thought that hardware would become so cheap as to become a commodity?

The same evolutionary trend is visible in later periods. During the period spanning late ERP to EAS the main limiting factor was no longer hardware or connectivity, but portability of software. Greater focus on presentation and the advent of HTML for web-based delivery unleashed new possibilities and the problem of application portability was soon addressed.

We have seen terrific evolutions in technology. The pace of these evolutions have happened at such a rate that they typically outstripped the willingness of business and ability of IT to keep pace. The result is that in todays business environment many of yesterdays business applications and technologies are still gainfully in service. Chugging along in the bellies of computing devices that most IT professionals consider relic are yesterdays business applications. The result is that the IT-systems of today are a far cry from being homogeneous, instead the IT landscape is a mixed heterogeneous tapestry of systems that are usually monolithic in nature.

When examining the origins of life, biologists look to the process of evolution to categorize and classify the past. In a similar manner, technology evolves and is subject to the same "natural selection" and "survival of the fittest" laws found in the natural world. Things that do not adapt to changes in their environments do not survive. Enterprise Resource Planning (ERP) systems and business enterprises are not exempt from the influences of these evolutionary principles.

P.J. Jakovljevic, in his article *Enterprise Applications - The Genesis and Future, Revisited*, traces the historical process of Enterprise Resource Planning systems. As we examine the past out of which SYSPRO developed, we will use the general structure that Jakovljevic uses in his article, since it presents a helpful framework within which to see the process of development.

According to Jakovljevic, integrated enterprise resource planning (ERP) software solutions became synonymous with competitive advantage, particularly throughout the 1990s. The idea behind ERP systems was to replace "islands of information" with a single, packaged software solution that integrated all traditional enterprise management functions such as financials; accounting; payroll; human resource (HR) management; and manufacturing and distribution. The objective was to ensure enterprise-wide transaction system coherency. Knowing the history and evolution of

ERP within the broader enterprise applications concept is essential to understanding its current use and its future developments.

Like fossils in the rock strata, different eras within the computing landscape provide us with evolving systems and improvements and adaptations.

Chapter 2. Background

1960s-Pre-Computer Era

SYSPRO was not formed as a company until 1978, but it is important to know this background information in order to understand the process of software development within the manufacturing and business sectors.

The focus of manufacturing systems in the 1960s was on inventory control. In this computing era when a computer would occupy an entire wing of a building at a local university or a government building, most manufacturing companies could not afford to own one. However, manufacturing companies had to be able to afford to keep enough inventory on hand to satisfy customer demand. It was the age of the re-order point system (ROP) where the assumption was that the customer would continue to order what they had before and the future would consequently look very much like the past. In most industries this was a valid assumption, since product life cycles were measured in years for competitive purposes at the time.

Inventory itself was regarded as an asset, not only on the balance sheet, but also in the mind of the average manager. Therefore, production planners created schedules and managed materials manually, whereby, in the production control office, the manual explosion of **bills of materials** (BOMs) often resulted in errors, since pesky index card files were used to record material allocations, receipts and issues. When the unallocated inventory balance on the card seemed low for a certain part, a planner would give a card to a buyer, who would then place a new purchase order. Those card files provided a real help to a planner as long as each index card was updated in a timely manner and then put back in the right place, which was far from guaranteed. The number of errors and faults within this manual system could not be controlled.

Further, the order entry/sales department usually created the schedule (for the items manufactured in-house.) As a result, people who had little or no access to material availability information loaded forecasted sales and actual customer orders into the schedule. This lack of visibility, combined with the cumbersome inventory record-keeping process, caused frequent schedule changes and missed or delayed customer deliveries. Often the shop would start an order only to learn that required materials were not available. As a result, the ensuing excessive work in progress/process (WIP) and raw materials tied up unnecessary capital funds and shop floor space, which ultimately led to a number of other missed opportunities.

1970s/1980s - Advent of Computers in Manufacturing

When computers finally became small and affordable enough to be deployed by an average manufacturing company, their use for resolution of materials mismanagement initially gained the highest priority status. Silently, the need to order only what was really needed crept into the mindset of manufacturers. No longer could a company afford to order 'some of everything,' since orders had to be based only on what was being sold, while what was already in inventory or committed to arrive on a purchase order or through internal manufacturing order would offset this requirement. As a result, **materials requirements planning** (MRP) computer systems were developed to provide for "having the right materials arrive at the right time," while the **master production schedule** (MPS) was built for the end items and finished goods. The MPS provided information to the MRP, which contained all the time-phased net requirements for the planning and procurement of the sub-assemblies, components, raw materials and ingredients.

SYSPRO was founded during this era, in 1978, initially developing applications surrounding the Stars II accounting system. In 1983 the company purchased the source code for Stars II and began converting it to COBOL. Three years later SYSPRO launched IMPACT, one of the first UNIX based accounting products on the market. During this time period, SYSPRO was also establishing itself in the UK, Canada, the US and in the Asia/Pacific markets. During this time period, the focus of the company was limited to the enterprise accounting arena. However, as the business environment changed and more was expected from software systems, so did SYSPRO software change, eventually evolving onto a full ERP system.

MRP - The Initial Impact

The impact that the computer had on material planning and enterprise management was immense. From the manual planning and huge inventory posting card decks, this new computer system promised to automatically plan, build and purchase requirements based on the finished products to be shipped, the current inventory on hand, the allocated inventory for other orders and the expected arrivals. The posting originally done on the manual input/output cards was replaced by transactions directly made in the computer and documented on pick lists. The amount on inventory was supposedly visible to anyone with access to a computer without having to go to the card deck to access the information.

MRP, or "little MRP," represented a huge step forward in the planning process. For the first time, based on a schedule of what was going to be produced supported by a list of parts needed for that finished item, the computer could calculate the total need and compare it to what was already on hand or committed to arrive. This comparison could then suggest an activity to place an order, cancel orders that were already placed or simply move the timing (i.e., expedite or delay) of existing orders. The real significance of MRP was that, for the first time, the planner was able to answer the questions "what, when and how much?." In other words, rather than being reactive and waiting until the shortage occurred, the planner could be proactive and time phase orders, including releasing orders with multiple deliveries. Indeed, the enterprise systems currently in use by most large corporations worldwide are an evolution of the MRP systems, one of the first being devised by IBM and the US tractor maker J I Chase. The early MRP systems were indeed quantum leaps, given they had managed to regiment to a degree former chaotic manual systems.

Nevertheless, some simplifying assumptions were needed to allow the computers of the day to make the required calculations. One was that the orders should be started at the latest possible date in order to keep a minimal inventories while still meeting the customers' delivery needs. This method is referred to as "backward scheduling." All orders were scheduled backwards from the desired completion date to calculate the required start date. There was no inherent slack time in the schedule and the downside of this assumption was that any hiccups in the execution of the plan would most likely result in late deliveries to the customer. Further, if only one part needed for the finished product was going to be late, there was no automatic way to know the impact on the other needed parts. Therefore, slack was built into the schedule through conservative, often unjustifiably pessimistic lead times. Despite this drawback, the benefits far outweighed the costs and more companies began to embrace the tools and techniques of MRP.

CRP - The Next Development

As more people learned how to utilize this material planning methodology, they quickly realized something else very important was missing. Namely, it did not suffice to have all the parts to get the job done, since sufficient plant capacity was needed as well. Thus, the idea of closing the loop with a capacity plan was introduced and "closed loop MRP," "big MRP," or **capacity requirements planning** (CRP) was born.

At the same time, computers were increasing in power and decreasing in price - a trend that thankfully continues today. Thus, the computing capacity to do the extra

mathematical computations was affordable and available. Now, not only could the materials be calculated, but also a capacity plan based on material plan priorities could be created. In addition to the BOMs needed for each of the finished parts, defined paths for the production process were necessary. Defined paths for the production process, commonly called routings, specified the machines or group of machines (work centers, production lines and so on) to be used to build the parts and the operations to be performed, so that capacity and workload could be planned and scheduled.

Yet, another critical assumption needed to complete the computations of the computers of the day was that infinite capacity existed at each of these work centers to satisfy this calculated demand when it was required. Unfortunately, infinite capacity is not an accurate reflection of reality and this drawback in the use of traditional MRP/CRP remains present today. However, for the first time, reports were available that identified the overload conditions and pro-actively resolved for each machine or work center. At least, this enabled some preparation of plans and options to address the overload situation before the problem occurred. Typically, lead times were long enough to allow work centers to "smooth out" unbalanced workloads in the short term and still support the overall required completion of the work order. Still, after the BOM explosion and time phasing of materials and capacity had been accomplished through MRP/CRP, other problems on the shop floor became evident. Namely, while planners created a feasible schedule with all the right material on its way or in stock, one would discover that maintenance on a critical piece of equipment had been overlooked or that skilled production workers were unavailable. Therefore, planning of all manufacturing resources, other than materials and nominal capacity, became the first priority at this stage.

MRPII - Connecting Manufacturing and Finance

The implications of the rapid use of computers in manufacturing reached beyond the factory floor. Once again the technology improved simultaneously with the realization that as every piece of inventory moved, financial transactions occurred or moved as well. For example, if a part was received at the factory or warehouse, not only should the inventory on hand quantity go up but also there should be a corresponding increase in the raw material inventory asset on the financial books. This is balanced by an increase in the liability level in the **accounts payable** (AP) account. As a group of parts moves to the shop floor to build the finished product, the raw material asset should go down and the **work in progress** (WIP) asset should

go up. The labor and overhead charges from the shop floor personnel also are added to the WIP asset account with an offset to the AP account. When the finished part completes its route through the shop, the WIP asset account goes down. Finally, as the finished product is sold, the finished goods asset account goes down and the **accounts receivable** (AR) asset account goes up. Consequently, at every step of the way, as the inventory moves, financial accounting moves with it - in duplicate - with balanced credits and debits. (These the principles of checks and balances and of double entry bookkeeping were established by old Venetians.)

The early system that has now evolved into SYSPRO was an enterprise level accounting system, the first fully capable of running on UNIX mainframes and servers. As such, it was a pioneer within the MRPII arena and a part of the movement that connected manufacturing and finance.

Available information technology (IT) now had the power and was affordable enough to track inventory movement and financial activity. As a result, the basic programs for manufacturing were integrated into one package using a common database that could be accessed by all users. These were the first manufacturing resource planning (MRPII) packages, used predominantly by discrete manufacturers. Since MRP assumes infinite capacity and strict adherence to schedule dates, process and flow manufacturers have found little use for it. Instead, during this, they instead focused their efforts on other aspects of the supply chain, particularly forecasting, purchasing and distribution.

By incorporating more resources and continuous monitoring of planned versus actual results, MRPII was a significant evolution of MRP. MRPII closed the loop with the financial accounting and financial management systems. The American Production and Inventory Control Society (APICS) defines MRPII in its dictionary, 11th edition, as follows:

“Manufacturing Resource lanning (MRP II) - A method for the effective planning of all resources of a manufacturing company. Ideally, it addresses operational planning in units, financial planning in dollars and has a simulation capability to answer what-if questions. It is made up of a variety of processes, each linked together: business planning, production planning (sales and operations planning), master production scheduling, material requirements planning, capacity requirements planning and the execution support systems for capacity and material. Output from these systems is integrated with financial reports such as the business plan, purchase commitment report, shipping budget and inventory projections in dollars. Manufacturing resource planning is a direct outgrowth and extension of closedloop MRP. ”

In other words, for the first time, a company could have an integrated business

system that: provided visibility to the requirements of material and capacity driven from a desired operations plan; allowed input of detailed activities; translated the activities to a financial statement; and suggested actions to address those items that were not in balance with the desired plan. Good information leads to good decisions, and, therefore, these integrated, closed-loop information systems provided a competitive advantage.

Meanwhile, other functional areas of companies had also been requesting help from data processing departments. Today, these are known as **management information systems** (MIS), information systems (IS), or IT departments. Systems were developed for support of each major functional area. As an example, accounting and finance departments had a set of programs that helped manage the *general ledger* (GL), *accounts payable and receivable*, *cash flow management*, as well as *capital assets* and *financial reporting*. These accounting programs were combined to form an integrated system for accounting, much like MRPII already integrated the manufacturing programs. Sales, engineering, purchasing, logistics, plant maintenance, project control, customer service and human resources departments followed suit and each developed their own sets of integrated computer systems. Unfortunately, these disparate systems were unable to interact and exchange information. Information exchanges between these systems, often time consuming and error prone, were enabled by **application programming interface** (API) programs.

Chapter 3. Computing Today

1990s - Enterprise Resource Planning

By the time each functional area of a company had developed its individual software program, the need for tightly integrating them became obvious. The next major shift during the late 1980s and early 1990s was that "time to market" was becoming increasingly short. The shift from 'Fordist' assembly line mass production to the modern mass-customization principles and mindset (You get to choose parts, colors, accessories, etc.) irreversibly changed the society and economy standards. Lead times expected by the market continued to shorten and customers were no longer satisfied with the service level that was considered world class only a few years earlier. Also, by the 1980s, competition from Japanese manufacturers and their philosophy has caused US and West European enterprises to look for new efficiencies using information technology.

Customers were now demanding to have their products delivered when, where and how they wanted them. Companies were therefore compelled to develop and embrace the philosophies of *just in time* (JIT) and closer supplier partnerships as a way to remain competitive. During the same time frame, the *cost of goods sold* (COGS) was shifting drastically from labor to purchased materials. Consequently, planners needed to know materials allocations or finished goods *available-to-promise* (ATP) values immediately after customer order entry. On the other hand, buyers needed to know the sales plan several months in advance in order to negotiate prices for individual materials. Empowerment of employees was needed to provide the agility that was required to compete in the market.

Hence, the highest priority for IT professional was to develop a system with tightly integrated programs that would utilize data stored on one common database for use enterprise-wide (actions in one department's program driving actions elsewhere.) No longer was it tolerable to submit a request to the IT department and wait several "man-months" of programming time to obtain this critical information. This common-database, company-wide integrated system was named Enterprise Resource Planning (ERP), as companies realized the need to see the entire picture.

ERP Defined

The American Production and Inventory Control Society defines ERP as follows:

"An accounting-oriented information system for identifying and planning the enterprise-wide resources needed to take, make, ship and account for customer orders. An ERP system differs from the typical MRPII system in technical requirements such as graphical user interface (GUI), relational database management system (RDBMS), use of fourth-generation language (4GL), and computer-aided software engineering (CASE) tools in development, client/server architecture and open-system portability; 2) More generally, a method for the effective planning and control of all resources needed to take, make, ship and account for customer orders in a manufacturing, distribution, or service company."

Given many new very recent functional and technological developments many may rightly consider certain parts of the above definition as somewhat outdated or not all encompassing. In general, the second part of the definition holds true, given the fact that traditional ERP involves software packages that by and large automate and support the processes of the administrative, production, inventory, and product development aspects of an enterprise. We must also remember that ERP was and still is evolving.

Impact of the Personal Computer

The cost of technology continued to decrease as the advent of the personal computer (PC) again revolutionized the face of business management systems. At a fast pace, the large inflexible mainframes were replaced by new client/server technology. The power of these small PCs exceeded the power of the large mainframes that had been routine only a few years earlier. It became possible to run a fully integrated MRPII system on a small PC. Still, these systems have trickled down slowly from large to smaller enterprises. Many manufacturing enterprises were not computerized in the 1980s. In fact this is largely true even nowadays in small workshops that still get by with little or no computerization.

IT, however, gained momentum in the 1990s, when PCs became even cheaper, software more sophisticated and companies became more amenable to using the technology. The changing pace of technology had once again encouraged the development of planning and control systems to fulfill real business needs. In addition, unlike previous evolutions, the ERP software vendors also offered these

critical business applications to non-manufacturing companies, selling "know-how" rather than physical products. ERP is far more than just MRPII which runs on a client/server architecture. It encompasses, material planning, capacity planning, quality control, forecasting, budgeting, purchasing, distribution, reporting tools and communication systems, to name but a few. These critical business issues affect not only manufacturing companies but also all companies that desire to achieve competitiveness by best utilizing their assets, including information. In other words, ERP systems should help companies become leaner by integrating the basic transaction programs for all departments, allowing quick access to timely information. However, ERP inherited MRPIIs basic drawbacks, which are the assumption of infinite capacity and the inflexibility of scheduling dates, preventing companies from taking full advantage of speedy information flow.

A typical ERP system indeed now offers broad functional coverage nearing the best-of-breed capabilities; vertical industry extensions; a strong technical architecture; training, documentation, implementation and process design tools; product enhancements; global support and an extensive list of software, services and technology partners. While it is not yet a system-in-a-box, the gap between desired and actual features is becoming smaller every day.

Evolution of ERP to ERPII

The current changes in the evolution of ERP started around the year 2001 and have since been gradually gaining momentum. These changes have now reached the critical mass required for user-businesses to raise their heads and take note. Depending on whom you are talking to, this step in the evolution of ERP will be given different names. The most common of all has been ERPII, a term coined by Gartner. At SYSPRO the term ERPII has been used interchangeably with the term "Extended Enterprise." The Extended Enterprise is the iteration that adapts ERP to the Internet based world of today and tomorrow, through changes in functionality, technology and architecture.

Within the extended enterprise, the functionality of applications becomes deeper and more specific to industry requirements, while the technologies employed are focused on leveraging the power of networking (internal and external networks and the Internet) for inter-enterprise connectivity and a unification of the end-user experience. The application of these technologies is not limited to inter-enterprise connectivity, but are also used to bridge gaps between existing and disparate systems within the enterprise.

To accommodate this, the architecture of traditional ERP has evolved in order to

provide easier integration and interoperability.

The extended enterprise is continuing to evolve and add value to business systems. The world in which we live is changing everyday. As the business environment changes, ERP and extended enterprise applications and platforms must adapt to those changes.

Part II. Extended Enterprise

Having seen the development of ERP and some of it's current evolution to ERPII and beyond, we are now able to delve more deeply into the influences and workings of the extended enterprise, specifically within the concept of a Service Oriented Architecture. This next section will take you through a discussion of business drivers that are currently affecting business needs and IT solutions, encouraging companies and enterprises to examine the concept of SOA and find what will benefit them by adopting it. After discussing these business drivers, we will examine some basic definitions of SOA and take a brief tour of the history of SOA. You will then be prepared for the indepth discussion of the concepts and technologies that support a SOA.

By the end of this part of the book, you will better understand the influence of the discussed business drivers as they affect the relationship between IT and business as well as the importance of these influences in relation to Servie Oriented Architectures. You will know the basic terminology of SOAs and the process through which the concept of SOA came into being.

As we delve deeper into SOAs, you will learn more about the benefits and challenges that SOA brings to the enterprise environment. We examine the benefits of business agility, better return on investment, future proofing, scalability, the alignment of IT with business goals, increased customer satisfaction, code re-use, focused development in multiple layers, better testing and debugging of code development, multiple client types and better maintainability. You will also understand the concept of the composite application platform in relation to the benefits of SOA.

As we discuss the current challenges of SOA, you will better understand the technical difficulties that must be considered. You will be familiarized with data rationalization, service enablement, abstraction compatibilities, accessibility of services and differing interoperability standards. You will also understand the business challenges of pricing services, defining ownership and the consequneces of change in the enterprise arena. As a final challange regarding SOA, we will discuss the 'build vs. buy' challenge and you will be able to make up your own mind on a possible solution between the two models.

Finally, you will have a better understanding of the role of security within a SOA environment as we examine the concepts of trust, message security, policies, identity management, interoperability of security systems and the current Web based security standards available.

Now let's get ready to broaden our understanding about the business world that we work in and learn a little about what is driving the evolving extended enterprise.

Chapter 4. Business Drivers

There are many demands that will influence enterprises and software systems. To list them all would take more time and space than five books the size of this one. We are therefore going to briefly examine the business drivers that are steering the evolution of ERPII systems like SYSPRO.

SYSPRO's enterprise software has evolved to meet the demand of today's extended enterprise market and will continue to improve in functionality and usability. As it evolves, there are specific forces that have influenced and will continue to influence and drive that evolutionary change within the functionality and usability of the system. The main business drivers that we will discuss are:

- Globalization
- Real-time information exchange
- The increased need for flexibility and agility
- The benefits of proactive decision making

These business forces are driving ERPII systems toward Service Oriented Architectures and whatever lies beyond. Let's take a bit of time to discuss each of these four business drivers.

Globalization

The term globalization is frequently used to identify the trend toward increased flow of goods, services, money and ideas across national borders and the subsequent integration and collaboration of businesses across the global economy. It is the generalized expansion of international economic activity which includes increased international trade, growth of international investment (foreign investment) and international migration and increased creation of technology among countries. Globalization is the increasing world-wide integration of markets for goods, services, labor and capital.

"Globalization" is a relatively new word that has also been used to describe the ongoing, multidimensional process of worldwide change. It describes the idea that

the world is becoming a single 'global' market. It includes the idea that time and space have been shrunk as a result of modern telecommunications technologies which allow almost instantaneous communication between people almost anywhere on the planet. It proposes that cultures are blending and mixing and that cultural icons and values from dominant northern cultures are being adopted in the south, while at the same time unique ethnic differences are being strengthened and local identities are being exerted. It describes the idea that the planet as a whole, rather than individual continents or landscapes, is considered as 'our home' and that some human activities can have a negative effect on people and environments far from their source or have an negative effect on the planet as a whole.

This trend is influencing how businesses identify themselves, their products and their markets. It is driving businesses to include the possibility of outsourcing parts of their processes to other parts of the world where there are greater economic advantages. It is also leading to greater collaboration among businesses at various levels, including supply chain and management.

Today, the US economy produces $12 trillion of goods and services annually with a population of 280 million. Including Scandinavia, 'new' Europe does $13 trillion with 450 million people, for a combined 'West' of some $25 trillion GDP by just over 700 million people. In contrast, Japan does $5 trillion. The South East Asian tigers (Korea, Taiwan, Singapore) do $1.5 trillion. The Asian periphery (Indonesia, Malaysia, Thailand, Vietnam, Cambodia, Bangladesh, Philippines) does $0.8 trillion. China and Hong Kong ($2 trillion) and India ($0.7 trillion) represent 70% of Asia's population, but still only produce less than 30% of Asia's GDP. Asia's combined GDP of $10 trillion still falls short of Europe's $13 trillion and America's $12 trillion. But Asia has about 3 billion people, even today outranking the West 4:1 (Europe 7:1 and America 11:1.)

Over the next three decades these relative population weights will continue to change, though not overly dramatically. The combined 'West' will probably stagnate (Europe may shrink slightly, while America will expand,) while Asia's population may still expand substantially (to over 4 billion, with bits shrinking, some stagnant, others still growing and nobody really knowing what pandemics such as HIV/Aids and potential epidemics such as bird flu will still do to the various regions.) In contrast, the relative economic positions should continue to change radically if Asia continues to industrialize rapidly, even as the old West expands at a much more leisurely pace. The biggest changes are expected in China and (with a lag) in India. On a 30-year view, China is expected to keep growing rapidly, possibly still averaging 9% annually and incurring up to 20% currency appreciation, increasing its GDP to $30 trillion by 2035. India, assuming 7% growth and currency appreciation,

would reach $7 trillion in today's dollars. Japan growing at just over 2% may reach $10 trillion. Asia's tigers growing at 5% would reach $7 trillion. The Asian periphery may reach $4 trillion. America, achieving just over 3% average growth, would grow its GDP to $30 trillion. Europe growing at just over 2% would reach about $25 trillion. So, by 2035 the global economic balance would look radically different from today. The US and China would be equally matched in size at $30 trillion each, though with US per capita income probably still four times higher than China's. Both countries would each outrank the other 'wannabees,' Europe only just so, Japan by 3:1 and the South East Asian tigers by 4:1.

According to Beth Gold-Bernstein [http://www.ebizq.net] (ebizq.net's strategic products and services manager and co author of *"Enterprise Integration: The Essential Guide to Integration Solutions"*) there is no longer such a thing as 'business hours.' She tells us that in a global economy, "business hours are 24/7." In the global economy, competition can arise from anywhere and at any time. With every possible tradable good or service increasingly turned into a global mass product and with sophisticated production and servicing networks on standby to offer competitive battle with high-volume and low-margin efforts, most of the world's productive capacity will increasingly face the requirement to modernize, innovate and trim costs. Implied in such efforts is increasing labor and business specialization. This is the continued drive of globalization.

Globalization is the biggest driver within the extended enterprise arena. It is the force behind many of the other drivers. The consequences of globalization for businesses are the challenges that Service Oriented Architectures will address.

Continued globalization forces change within each sector of the market. Due to this change, businesses everywhere are increasingly being challenged to change within the markets in which they are involved. Flexibility and agility are two terms that deal with change within an environment, business or biological. Let's examine "Flexibility" and "Agility" from the business perspective.

Flexibility and Agility

As we have seen, these two terms deal with the idea of change within an environment. They also have very similar meanings. In the business sense, "flexibility" refers to the ability of the enterprise' system to quickly incorporate other systems and "agility" refers to the ability of the enterprise to respond quickly to a change in the market. Both terms are a measurement of the ability to quickly change.

Agility is the ability to change direction quickly. This requires an adaptive infrastructure so business processes and rules can be quickly changed. It also requires an adaptive organization. People must also be agile and willing to change to optimize business processes and increase competitiveness. The infrastructure needs to support rapid change.

According to Beth Gold-Bernstein, of ebizq, "Business agility is one of the key business buzzwords today and is showing up on business plans as a primary strategic goal." With the other trends of globalization, increased speed of communication, real-time information management, etc., only the companies that have increased flexibility and agility will have a competitive edge.

When thinking of flexibility, we must note that in the past manual business processes were very inflexible. If a company incorporated another company through a purchase, a completely new set of books (and sometimes a completely new set of operating procedures for the business) would have to be started. With the advance of computing and process automation business processes were able to be done more and more quickly (an increase in speed, not agility.) Automated business processes were still somewhat inflexible as the data processes within the business processes would usually involve proprietary systems that could not easily interface with another automated system. This inflexibility would cause an incorporating company to spend additional time on a installing a new system or trying to combine the two systems effectively. During this time, their competition would have an advantage.

The speed of new technology and the increase in processing capacity contribute to the drive of enterprise flexibility. Greater functionality and usability are possible as a result of improvements to flexibility and agility that were not possible years ago. Smart enterprises want to be more flexible and more agile.

The speed of new technology and the increase in processing power also contributes to the drive toward utilizing real-time information. As processing capacity has increased, the demand for faster and simpler access to data and reports has increased.

Real Time Information

The speed of new technology and the increase in processing power also contributes to the drive toward utilizing real-time information.

The concept of real-time is well known in our everyday environment. Simply put - "I want it! And I want it NOW!" The increase in technology within the communications and IT fields has allowed automation of many key processes and this has given the end user the possibility of working in a system without time delays (or with almost unnoticeable time delays.)

Real-time data access and real-time reporting are determined by the speed at which information changes and when it becomes available for the user to manipulate. Users often think of real time as something that occurs immediately. Providing data in real time remains a key challenge for organizations that want to maximize the value of their application and enterprise platform investments. As soon as sales data, user data, manufacturing capacity data or other key data areas of an enterprise system become available in real-time, the enterprise is able to forecast more accurately and make changes more quickly. Real-time information requirements also force administrators and system architects to make serious decisions about the architecture and design of an information system. Bottlenecks must be decreased and data synchronization increased.

Real time information delivery allows a business greater leaps in productivity by sharing more information with more people. Organizations today face intense pressures to enable people with the information they need to make critical decisions all day, every day and deliver this information in a way that is cost-effective. To meet this challenge, we must be able to deploy information delivery systems to very large numbers of people across the enterprise - rapidly and at a low cost of ownership.

How could a business do this? Ideally this is done by:

- Providing information, in real time, to everyone who impacts key business processes - employees, management, suppliers, partners, customers and citizens.
- Delivering this information to people the way they want it - in the most commonly used formats, e-mail, Web pages, Excel, PDF and word-processing documents.

- Implementing a personalized, secure, organized approach to business intelligence that delivers information broadly throughout the enterprise.
- Increasing users' abilities to interact with information.
- Enabling faster development and deployment of business intelligence applications.
- Using Web-based technology to meet the ever-growing demands for information.

The value of real time information is the enhanced capacity of reporting and forecasting. Real time information allows an enterprise to know what is going on when it's going on and gives possible reasons for why it is happening. For this reason, access to real time information is changing how businesses operate. The lag between forecasting demand and being able to measure demand is vastly reduced by real time information and this is enabling demand driven supply networks and business intelligence processes.

In a business situation we would find real time information being used by a group of companies in a supply chain network. They would work together using a single data source that is updated as soon as a user has entered and confirmed data. In the Outdoors Company, for example, data would be shared between the supplier companies and the delivery companies in real-time. As soon as the order is ready it is entered into the database as 'delivery ready.' The delivery company immediately knows that there is an order ready because the information is delivered to it as soon as the database is updated.

The increasing need to know what is happening, when it is happening is driving the adoption of real time information processes. It is no longer acceptable to update databases overnight or create reports only at the end of the month. The ability to access information in real time is a necessary aspect of today's business world.

Just as the increase in processing capacity has influenced the demand for real-time information, so too has real-time information raised the demand for pro-active decision making. If you have up-to-date information regarding the sales of a product, then you are more likely to be able to make decisions using that data, rather than a gut feel or a market projection based on possible sales.

Pro-active Decision Making

The ability to measure trends and make predictions based on business intelligence rather than gut-feel is the defining point of pro-active decision making. By using the analysis and predictive tools available within a system, a business is able to forecast what products will have a higher demand in the near future and adjust processes to capitalize on that higher demand.

Pro-active positioning also refers to the decision made by a business to use best practice methods and models. Service Oriented Architectures in the form of demand driven supply chains and collaboration and business-to-business networks are all part of the pro-active positioning movement. Instead of merely reacting to market change and fixing business processes, we have an opportunity to plan a response and build systems that improve business process evolution.

Pro-active decision making based on factual trends and analysis has been a dream for many business owners. In the past this dream has always been out of reach due to the complexities of gathering enough data and forming statistical processes that are transferable from one situation to another. With faster computer processing power and increases in functionality of business processes this is no loner the case. It is no longer just the mega-sized corporate that can afford to analyze the world markets. The processes and information are readily available within the extended enterprise.

In the Outdoors Company we would find a business analytics software system that allowed the decision makers of the company to examine the trends in demand and supply of bicycle parts and the trends in the costs of delivery. The analysis of this information would enable them to make better decisions. The decisions would be based on measurable data, not just a feeling of where the market was going. They would be able to be pro-active about a product that was not selling by immediately reducing their manufacturing capacity of that model and using the capacity for a model that was in higher demand instead.

Conclusion

The extended enterprise is influenced by many different factors. The four that we have discussed here are relevant to our discussion of SOA and where SYSPRO e.net solutions is evolving. The world of yesterday is fast disappearing; the world of today is changing rapidly and the changes of today are calling businesses towards greater cooperation and greater involvement in networked architectures. The evolving

business must find a way to connect the people, the business processes and the strategic information that these people and processes need. Those businesses able to incorporate these changes will be the ones that prosper and grow.

Chapter 5. Connecting all the Pieces

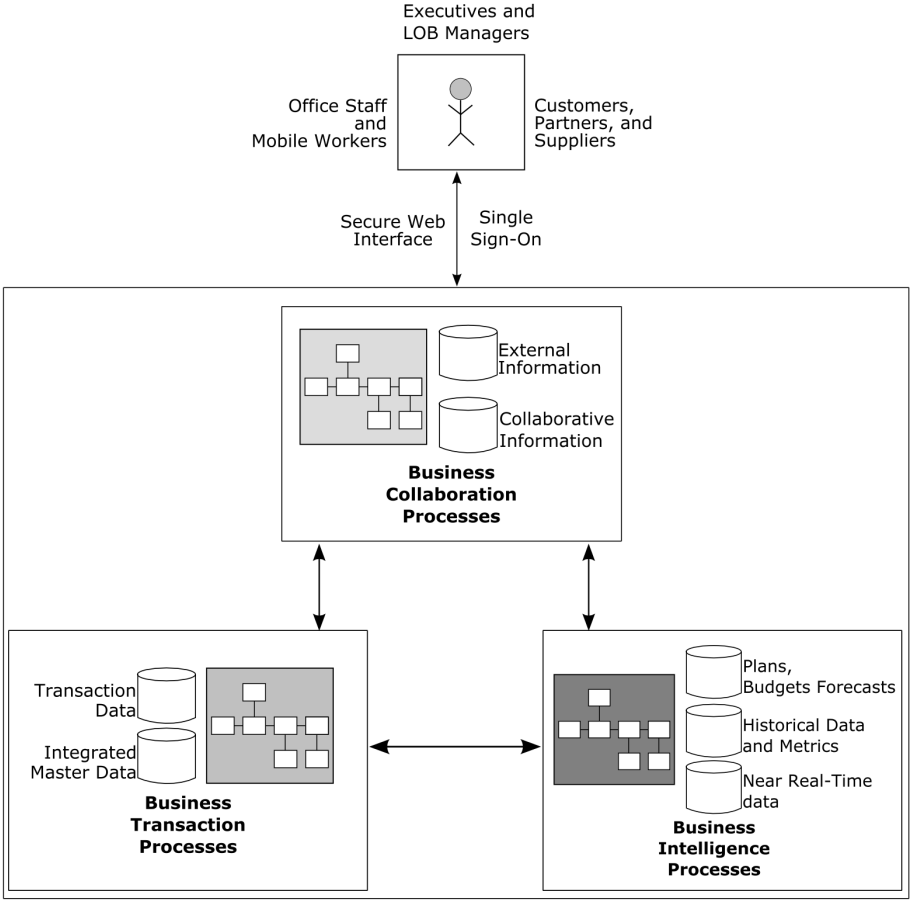
As we have seen in the conclusion of the previous section, the evolving extended enterprise requires the ability to connect people, business processes and strategic information.

No person is an island in itself, but each person is connected to many others and can only live a normal healthy life in relationship with other people. The same is true of every business. No business is an island. Each business is dependent on customers and other businesses. Even within a single business there are connections and relationships between people and between internal and external processes. The ability to connect and share information is increasingly being linked with a greater ability to compete and do business in the global arena. This sharing is enabling greater optimization of businesses systems, inventories and supply chains and is a foundational step along the path of an enterprise's ability to consider a Service Oriented Architecture. Let's take a look at the interaction of these three aspects, starting with business processes.

Business Processes

Smart businesses connect people with their business processes and business information as simply, quickly and seamlessly as possible. This enables the organization to make informed and timely decisions and optimize the business - maximizing revenues and profits. Maximizing revenues and profits is achieved through the automation of core business tasks. These tasks may involve business transaction processes (for managing day to day business operations,) business intelligence processes (for analyzing and optimizing business operations,) and business collaboration processes (allowing business users to communicate and share information about business operations.) When combined, these processes enable business users to run, optimize and communicate about all aspects of business operations.

Figure 5.1. Connecting People, Processes and Information



Each of these areas of connection and sharing increases an enterprise's ability to respond to the challenges and drivers within the global economy. To better our understanding of this process we need to examine each part of the collaborative business process:

- Transactions [29]
- Intelligence [29]
- Collaboration [30]

Transactions

Most business processes can be defined in terms of a transaction. Just like when we deposit money in the bank and receive a deposit slip detailing what has happened, so other business transactions create records of what has occurred. Some transactions are sequential and leave a trail of records. Others are once off and create a single record of what happened.

Transactional business processes run applications that support day-to-day business activities, such as receiving customer orders, managing inventory, shipping products or billing customers. The record of the data associated with these processes is stored in *transactional* data files and databases. The data may also form part of a master or integrated database, one that stores all the details of customers, products, accounts, etc.

Very often, during the course of business, the transaction processing *master data* about customers, suppliers, products, etc., becomes scattered across multiple systems. There is therefore a need to consolidate and coordinate this master data so that it can be used by other business processes, presenting a single view of the data to business users. Data integration is essential to the business collaboration process.

The sheer volume of business transactions happening in a one day period is incredibly large. If you think of a banking enterprise that has branches all around the world, then the number of transactions could reach the billions in a 24 hour period. This means that the overarching picture of what is happening in a business may be lost due to the vast quantity of transactions happening. A system of reporting, or business intelligence, is required to make sense of it all.

Intelligence

Business Analytics (also known as Business Intelligence) applications monitor and analyze business transaction processes to ensure that they are optimized to meet the business goals of the enterprise. These business goals may be operational goals that affect daily business operations, tactical goals that involve short-term programs (like marketing campaigns) or strategic goals that entail long-term objectives (like increasing revenues or reducing costs.)

Most business intelligence processes focus on reporting and analyzing business operations. (They *measure* business performance.) According to Colin White of BI (Business Intelligence) Research, this needs to change towards *"managing* business performance, by using business intelligence to align business operations with the

tactical and strategic goals of the organization." Colin White tells us that this is achieved by "extracting transactional data and integrating it in a data warehouse for processing BI performance management applications." These applications will convert the integrated (but raw) warehouse data into *actionable business information* that shows how actual business performance compares with business goals and forecasts.

Business users employ their business expertise to evaluate the actionable business information produced by the business performance management applications. This evaluation is termed *business knowledge* and is used to determine what actions (if any) need to be taken to align business activities with business goals, or what business processes need to be modified to better support those actions. This creates a so-called *closed-loop* decision making and action-taking system for managing and optimizing business operations.

An efficient and integrated closed loops system must enable a business to work smarter by closely aligning business performance to tactical and strategic business goals. This creates a feedback loop where positive activities are recognized and encouraged, while value detracting activities are either improved or eliminated.

The need for organizations to be more agile, however, requires that this closed-loop process also be employed to optimize day-to-day activities. This requires near real-time or low-latency transactional data (data that is available to the rest of the system as soon as it is entered.)

It is not merely enough to record all the transactions of an enterprise and use that information to create business intelligence reports for analysis and evaluation purposes. The smart enterprise must create an atmosphere of collaboration so that this vital information gets to the people who most need it.

Collaboration

According to Colin White, business collaboration processes "enable business users to communicate and share information about business transaction and business intelligence processes." This enables businesses to make decisions and take actions to optimize and improve business activities. Business collaboration processes are therefore essential to a closed-loops system as discussed earlier in this chapter. At present, most closed-loop processing is manual in that business users employ collaborative services to communicate and share information via e-mail, instant messaging and Web conferences. In some organizations, however, manual business collaboration processes are being automated to improve the speed of the decision-making process. This is being done by encapsulating business users'

expertise in a set of business rules, which are then used by a rules engine to automate action taking. Not all business processes lend themselves to automation, but automating activities like granting loans, issuing credit cards and processing claims for low-risk clients and customers can create a significant competitive advantage.

We have now discussed the business process aspect of smart business connections. This discussion will form the basis of our discussion of business information, for as we will find out in the next section, the end result of all business processes is business information.

Business Information

Business information is created by the processes of business transactions, business intelligence and business collaboration and results in a wide range of different types of structured and unstructured information. Business transaction processes store data in data files and databases, whereas business intelligence processes employ data warehouses and low-latency (a short time lag between one action and the next) stores to handle integrated data, transactional data and analysis results. Business collaboration processes encapsulate communications and business knowledge generated by business users in documents, spreadsheets, e-mails and other electronic media. All these various types of information are typically managed by a variety of file and database systems that are spread throughout an organization.

Business decisions usually take some time to make and may involve actions that cycle through more than one closed-loop cycle. For this reason business users need to have a single tool that enables them to access and track various types of business information over a period of time and that also allows them to share business knowledge and collaborate with other users inside and outside of the organization. This is important, not only for managing the business effectively, but also to be able to respond rapidly to legislative reporting requirements. Such a tool is often called an *enterprise portal* (discussed in the next section.)

Business processes and business information are both useless without the people or business user side of the equation. Smart businesses must utilize the right tools to connect their people (employees, clients, customers, etc.) with the correct business process and business information.

Business User

In order to make rapid and informed decisions, business users need access to personalized business content from many places at any time, using both office based and mobile devices. An *enterprise portal* satisfies this need by providing business users with an integrated and secure Web interface to enterprise wide business content, including business transaction and business intelligence applications, structured and unstructured data and collaborative documents.

An enterprise portal includes a content management system (or interfaces to external content management system) for organizing and controlling shared information that is viewed through the portal. Many portals also provide collaboration facilities that allow portal users to communicate with each other and share and exchange information.

Portals provide secure and single sign-on facilities that enable them to be employed both inside and outside of an organization. Internal portals are frequently used to improve the usability of corporate intranets by personalizing information and increasing information self-service. External portals are used to enable the interchange of business content with customers, suppliers and other key business partners.

According to Colin White of BI Research, "An enterprise portal is destined to become the business users' desktop by providing all of the business content they require to do their jobs." A portal solves the problem of locating information because it personalizes and secures business content to match each user's role in the organization, making it easier to find and access information. A portal employs a *business taxonomy* (a self describing system of cataloging and naming data) to provide an integrated view of the many different types of business information that exist in an organization.

The idea of an enterprise portal must be taken a step further. If we examine the functionality of an enterprise portal, we find that it acts as an information service provider. It allows access to particular types of information and increases the usability of that data for all those granted access. When we apply the idea of a service to the rest of the business processes, we begin to apply Service Oriented Architecture to the extended enterprise.

Making the Connections

We have seen that the smart enterprise needs to connect people, business processes and strategic information. The way to do this need not be complex nor complicated and can be managed by intelligent software like the enterprise portal. The system that uses an enterprise portal or similarly functioning integration layer forms the basis of a Service Oriented Architecture. We will see this more clearly in the next chapter.

This page intentionally left blank

Chapter 6. Service Orientation

The business drivers that we have examined and the ability to connect people, processes and information have challenged businesses to examine and adopt the concept of Service Oriented Architectures (SOA.) It is important to have understood these trends so that we can better understand:

- What SOA is;
- Where SOA came from;
- What SOA uses;
- What the benefits and challenges of SOA are;

And ultimately how to implement SOA within your enterprise.

With all of this in mind, we will start by introducing you to the concept of service oriented architetectures.

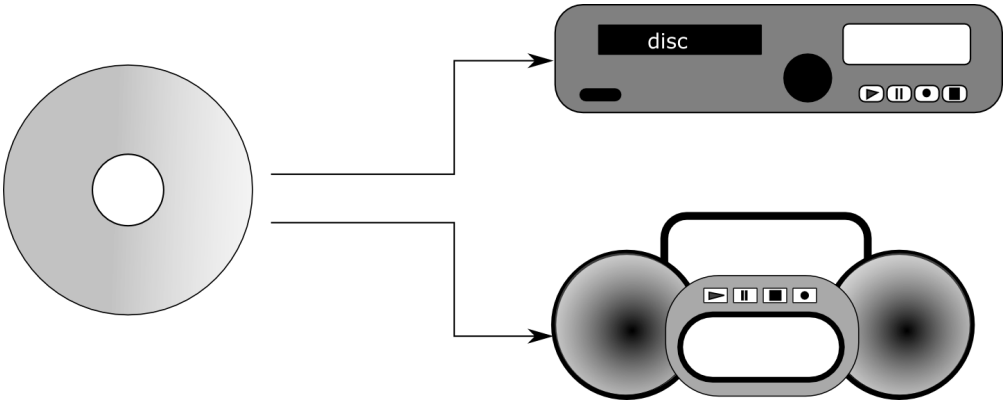
Introduction to SOA

Service Oriented Architecture is one of the top discussion areas in enterprise business software and business practice today. If we look at the trends in business, we will find that businesses that are actively engaging in changing their business environment to that of SOA are experiencing a competitive edge over those that are not. According to Michael Barnes, Daniel Sholler and Paolo Malinverno of Gartner, "Service-Oriented Architecture projects are becoming more common, especially in the financial service, telecommunications and government markets." Beth Gold-Bernstein of ebizq tells us that "Service-Oriented Architecture, considered a best practice for over two decades, is finally being embraced by many organizations seeking to increase business agility and decrease the time and cost of implementing new business solutions." An understanding of this concept is vital to our discussion of today's extended enterprise. For this reason we will focus first on defining SOA. Then we will discuss it's history, its key components and the concepts and technology that support it.

SOA Defined

In order for us to discuss the concept of a Service Oriented Architecture (SOA) we need a common definition of what we mean by SOA. There are many different definitions of SOA, ranging from highly technical definitions to those that are simple (but of limited use.) For example, we can describe the SOA concept using the CD and CD-player. If you want to play a CD, you put it into a CD player, press play and the player plays it for you. Think of the CD player as offering a 'CD playing service' and the CD as the 'business data' that needs to be processed. As such, you can replace one CD player with another - you can play the same CD on a portable player or on an expensive stereo system. They both offer the same CD playing service, but the quality of the service is different. You can also play many different CDs in one player. The CD playing service does the same thing with each CD - it plays the sound recorded on the disc. This is a thought picture that we will keep in mind as we delve deeper into the concept of SOA.

Figure 6.1. CD Player as a Service



Now, let us look at a simple definition of SOA which builds on the CD thought picture and then continue to build our understanding by discussing some of the key components of SOA. This is a basic and fairly simple definition:

- Service Oriented Architecture is a business environment style of design that guides and encourages the creation of loosely coupled business services that are inter-operable and technology-independent, enabling increased business function

and flexibility.

As you can see, within our definition we have introduced words and concepts with which you may be unfamiliar, like 'loosely coupled', 'inter-operable,' and 'technology-independent.' These are covered in the key components and supporting technologies sections of this book. (See the section called “Key Concepts in SOA” [?].)

For a better balanced view, let's take a look at another definition. The World Wide Web Consortium (W3C) Web Services Description Language (WSDL) 1.1 [http://www.w3.org/TR/wsdl] defines SOA as “A set of components which can be invoked and whose interface descriptions can be published and discovered.” Again, there are terms in the definition that must be understood in order be able to discuss the nature of SOA and how this type of architecture or environment will benefit business. The CD playing service operates on the same concept in every CD player (It is the set of components that can be invoked.) However, there are many manufacturers of CD players. They all follow the same standard and all CD players are recognizable as such because of those standards. When SOA has been implemented on the enterprise landscape there will be many 'manufacturers' of services who will put their services in directories so that many businesses can utilize them. This interoperability is what SOA brings to the business environment.

No matter what definition of SOA we use, the core of SOA deals with business and IT practices and applications. SOA is a business tool or concept built from IT. In order to better understand the relationship between IT and business in SOA we need to know more about how SOA came to be.

Quick History of SOA

There is much hype around SOA, so it often comes as a surprise when people learn that Service Oriented Architecture concepts and implementing applications have been around for well over a decade, some of them even longer. SOA has increasingly caught the attention of the current enterprise market due to the growing popularity of Web Services. It will be useful to first take a step back and look back at the evolution of SOA before we examine some of its key components. To see this quick history of SOA we simply need to look at the challenges that software developers have faced over the last few decades and observe the solutions that have been proposed to solve their problems.

Early computer programmers realized that writing software was becoming an increasingly more complex task. They needed a better way to reuse some of the code that they were continually rewriting. When researchers addressed this need, they introduced the concept of modular design. With modular design principles, programmers could write subroutines and functions and reuse 'modules' of their code. This was great for a while, but problems with the modular system were soon recognized. Developers found that they were cutting and pasting their modules into other applications and that this began to cause a maintenance nightmare. When a code problem or a 'bug' was discovered in a function somewhere, they had to track down every single application that used that function and modify the code to reflect the fix. After the fix, the deployment nightmare began. Developers didn't like that; they needed a higher level of abstraction - a way to program in concepts rather than just in code modules.

Computer programming researchers and IT specialists then proposed classes and object-oriented software (bundles of code that can be thought of as building blocks or bricks, sections of code that can be moved around and copied from one program to another without changing its usage) to solve this and many other programming problems. Again, as the software complexity grew, the developers started to see that developing and maintaining software was too complex and they now wanted a way to reuse and maintain functionality, not just code. Advancement always seems to bring a desire for more. Researchers offered yet another abstraction layer to handle this complexity -- component-based software. Component-based software was and still is today, a good solution for reuse and maintenance, but it doesn't address all of the complexities that now face developers. Today, we face complex issues like distributed software, application integration, different OS platforms, varying protocols, numerous and varied input devices, the Internet, etc. Today's software has to be able to answer the call for all of the above. New standards are being developed and new possibilities have emerged from the programming complexity. The concepts behind SOA may not be new, but the ability to mix and match operating systems, programming languages, execution environments (i.e., Java, .Net) and clearly separate the service interface from the execution technology is. This improvement allows IT departments to choose the best execution environment for each job (either a new application or an old one) and tie them together using a standardized architecture.

Don't worry if the more technical words in this brief history of SOA are unfamiliar. The central point of tracing the evolution of SOA is to show that the concepts that brought about SOA now help software programmers and business users deal with the increased complexity of the business environment, technological advances and improved IT. Service Oriented Architecture is, after all, much more of a mind set, a

development philosophy and an architecture, than it is anything specifically concrete and definitive. The success or failure of SOA is not reliant on the advances in IT or business processes but upon the ability of organizations to adapt and change their approach.

One final thought as we look at from where SOA came. In the past the same individuals in the IT departments were responsible for understanding both business and technical functions. The same person or group of people had to bridge business and technology domains. Within a SOA system this way of doing things can change. Technical staff can now reorient themselves from thinking about the entire job to doing just a piece of the job that will be used and completed by someone else.

SOA is now an enterprise best practice reality. Smart businesses have already assessed the advantages and challenges that SOA bring and are working toward implementing SOAs in order to maintain their leadership in the global market place.

You have now been introduced to Service Oriented Architectures and are ready to delve deeper into the inner workings of SOA, the central concepts at work and the terms used when describing and discussing SOA

Key Concepts in SOA

In this section we will take a closer look at the central concepts that form a Service Oriented Architecture. In the context of SOA, we use the terms service, message based technology, dynamic discovery, loose coupling, distributed networks, interoperability, Web Services, front-ends and the enterprise software bus. Each of these plays an essential role in conceptualizing what SOA is and how to apply the concept to a real time business. As the concept of the service is so central to the discussion of SOA, we will start the discussion there.

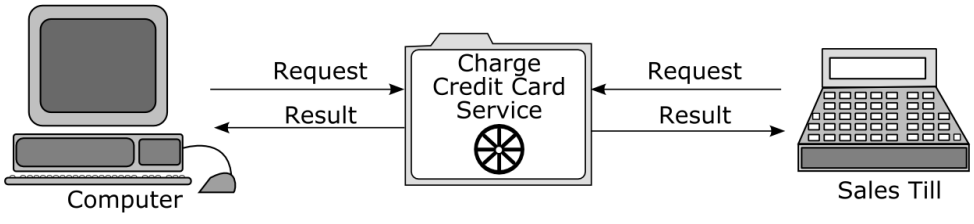
Services

When examining any SOA you will find the concept of a 'service' at its very core. Everything either enables or supports a service within the enterprise IT architecture. The 'service' is what brings all the other aspects of the architecture together. It is the most important part.

In an article at builder.com, Brian Schaffner informs us that "a central idea of SOA is the movement away from technology-oriented solutions and toward business services." Most organizations (including governments) provide some sort of business

service to their customers, so the idea of a service is not uncommon. In this context, however, our definition of a service incorporates two related fields: business systems and computer applications. Services are the central or core component of SOA, its most basic element. A service is a self-contained software module that performs a predetermined business task: i.e., "verify a customer's credit history."

Figure 6.2. Charge Credit Card Service (Verify Customer Credit History)



In SOA services are business software components created to function just by themselves, regardless of specific underlying technology. This allows the programmers to reuse the code and maintain the functionality of components within an application and share them with other applications. However, SOA also promotes application assembly because each service can be reused by different consumers. For example, in order to create a service that charges a consumer's credit card, we build a computer software component that accepts input (the credit card number and the amount charged) from an outside source, processes that input (performs a credit card transaction with the bank in question) and then returns information to the outside source (the confirmation codes or rejection.) We then setup this software component in a network environment (like the Internet or local area network,) and then we can use it from any number of applications within that network - one single instance where the software code functions as a service for all applications that need to charge a consumer's credit card. We can assemble various systems using standardized services for each different enterprise function. Remember that the "services" in SOA are business services. Updating a loan application is a business service; updating a record in a database isn't.

Another way of viewing a service in SOA terms is to define it as an exposed piece of functionality with three particular properties:

- The service interface is **platform-independent**.

  A platform-independent interface implies that a user or client from anywhere, on any operating system and in any language, can consume or use the service. This means that given the right tools in an application, anyone can connect to and make use of the business service.

- The service can be **dynamically located and invoked**.

  Dynamic discovery means that a discovery service (e.g., a published or searchable public directory) is available. Having a service available is not enough. Users have to know where it is and how to use it. A directory service enables a look-up mechanism where consumers can go to find a service based on a specific criterion. For example, if I was looking for a credit-card authorization service, I might query the directory service to find a list of service providers that could authorize a credit card for a fee. Based on the fee, I would select a service.

- The service is **self-contained**.

  Being self-contained means that the service maintains its own state and is an independent process. The service is not dependent on another process but is complete in itself.

Services are developed within a larger context than a programming object or procedure. They are designed for reuse, both in the context of other services and in combination with other services. Ideally a service will have a 'service contract' which will provide an informal specification of the purpose, functionality, constraints and usage of that service.

An easy example of a service, with the context of SOA, involves the banking system. We have all walked onto a bank, filled out the deposit form, joined the queue and been able to perform a transaction in our account through the mediation of the bank teller. When you enter a bank, you expect a particular service. You understand what information is required and where to go to get the particular service that you require. The core service or business process that you expect is the money transaction. You want to move money from one place to another. You fill out the necessary details of what you want and then hand it over to the teller, who performs the task. All the requirements and details of performing this transaction are steps along the business process, aspects or properties of the 'deposit money service.'

As we examine the other key components of SOA, you will see that most, if not all, of the other components are specific supporting aspects or properties of a service.

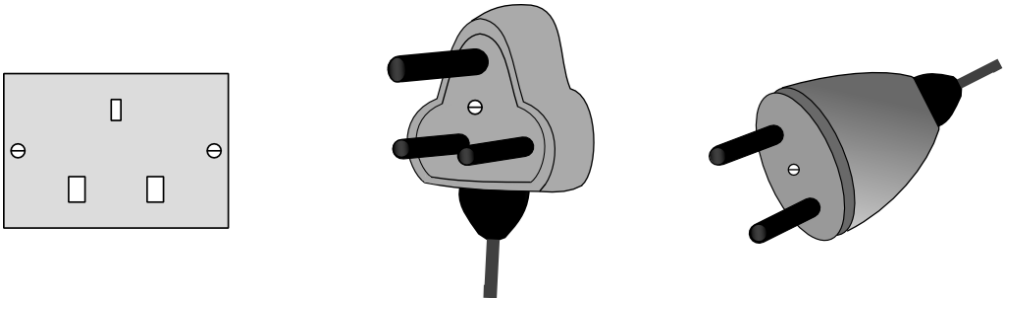
We now look at some concepts of how services and systems should interact. Remember that we are discussing the key concepts of SOA and that each of the concepts that we are discussing will work together to form a working SOA.

Loose Coupling

Coupling deals with the aspect of how software components and services fit together. As we build more and more software systems, we see similar situations and patterns appearing. Naturally, we want to reuse the functionality of existing systems rather than building them continually from scratch. The problem that loose coupling tries to deal with is that of dependencies. A real dependency is the state of affairs in which one system depends on the functionality provided by another. They cannot be separated. The trouble is that we create artificial dependencies along with real dependencies. Loose coupling creates functionality in services that are not permanently bound to a specific application or system, but are able to be moved from one platform to another, one application to another, etc.

If you travel overseas on business, you know that you must bring power plug adapters along with you or your life will be miserable. The real dependency is that you need power; the artificial dependency is that your plug must fit into the local outlet. Looking at all the varying sizes and shapes of those plugs from different countries, you would notice that some of them are small and compact while many others are big and bulky.

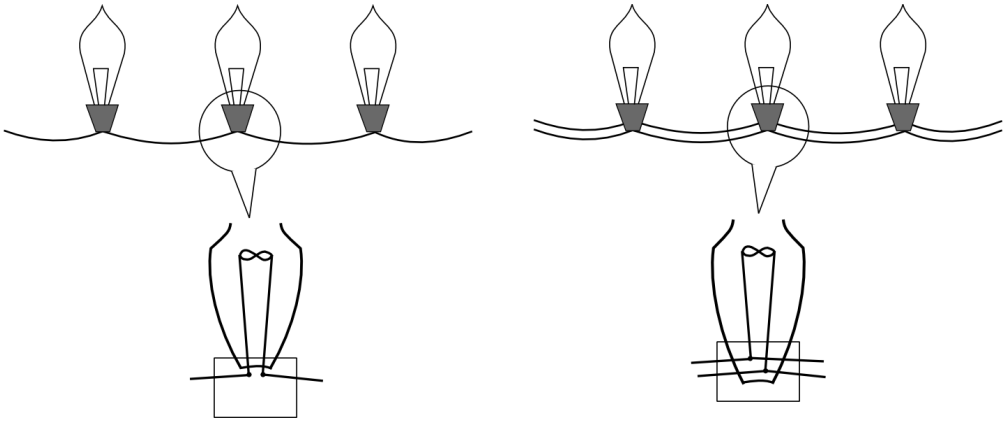
Figure 6.3. Artificial dependency - different power plugs and sockets



The lesson here is that we cannot always remove artificial dependencies, but we can reduce them. If the artificial dependencies among systems have been reduced, ideally, to their minimum, we have achieved loose coupling.

Another way of seeing loose coupling is to think of a string of Christmas lights. If you have ever had to deal with an old set of Christmas lights, you will know the frustration of having a bulb burn out. When one bulb went the whole string of lights wouldn't work. Each bulb's power supply depended on the connection of the other bulbs. This is similar to tight-coupled applications that depend on other applications and specific hardware types. In order to fix the old set of lights you had to take one bulb out and try a new one in its place, turn the power on and if it didn't work, try the next one until you found the place in the string with the broken bulb and were able to replace it. The new strings of Christmas lights are loosely coupled. If one bulb blows the rest stay on and you can immediately see which bulb needs replacing. The difference between the old and new string of lights is that the old one carries one wire connection through all the bulbs, making each bulb dependent on the other, while the new string carries a two wire connection to each bulb, connecting each bulb to the live and neutral wires and making each bulb independent.

Figure 6.4. Loose coupling advance in Christmas Lights



Within the banking system we find loose coupling at work within our deposit transaction example. One can deposit money at any branch of the bank, at an ATM or through an Electronic Fund Transfer over the Internet. The 'deposit money service' is not dependent on a particular teller or bank branch. The business process is standardized, but independent enough to take place from many different starting points.

When we consider a SOA service, we can see that loose coupling is essential, for it allows the service to be independent of operating systems and independent of other business services.

Another concept of how services and systems should interact relates to interoperability. Loose coupling and interoperability are inter-related. If a service is loosely-coupled, it will more likely be inter-operable, regardless of from what operating system it is run.

Interoperability

What do we mean by inter-operable? Interoperability involves allowing things to work together or finding ways to make them work together. Standardization practices are the tools that allow things to become more inter-operable. For example, when you fit new tires on your car, you expect tires from many different tire manufacturing companies to fit the wheels on your car. Any standard tire will fit, regardless of who made it. Tires have been standardized for many years. Within a business service environment the same should also be true. To create interoperability, one should actively be engaged in the ongoing process of ensuring that the systems, procedures and the cultures of an organization are being managed in such a way as to maximize opportunities for exchange and re-use of information, both internally and externally.

With respect to software, the term interoperability is also used to describe the capability of different programs to read and write the same file formats and utilize the same protocols. It is also true that interoperability is often more of an organizational issue. In other words, interoperability frequently has a major impact on the organization concerned, including issues of ownership (Do people want to share their data?,) staff (Are people prepared to undergo training?) and usability.

Interoperability is crucial to a SOA. Although the services are the core of a SOA, interoperability is the glue that keeps everything together and the services available to who that will use them. Without interoperability there is no possibility of integrating the systems, the data, the applications, the providers and the users of a

Service Oriented Architecture.

When you want to withdraw money from an ATM, you are encouraged by your bank to utilize their particular ATMs, but if their ATM is not in your vicinity, you can still withdraw money from your account using another bank's ATM. The ATM system allows interoperability between the banks (for a standard service fee.) The bank ATM system works together within a standards based framework to allow secure access to accounts held at banks other than the bank supplying the ATM.

We have now seen two key concepts of how services and systems interact, loose-coupling and interoperability. The next concept that we will discuss deals with how services, systems and end users interact. This involves a process of dynamically discovering or locating the service within a system.

Dynamic Discovery

When discussing the concept of a service, we saw that it can be dynamically located and invoked. This is the aspect of dynamic discovery.

Dynamic discovery is an important piece of the human element of SOA. At a high level, SOA is composed of three core service pieces:

- Service providers
- Service consumers
- Directory service

The roles of providers and consumers are apparent, but the role of the directory service needs some further explanation. The directory service is an intermediary between providers and consumers. Providers register with the directory service and consumers query the directory service to find service providers. Most directory services typically organize services based on criteria and categorize them. Consumers can then use the directory services' search capabilities to find providers.

Another term used to describe this principle is 'Service Repository.' A service repository would contain all the information about a service, including the physical location, information about the service provider, contact people, usage fees, technical constraints, security issues, access rights and the available service levels.

We are all familiar with the concept of a directory like "The Yellow Pages." Businesses advertise their presence within their specified subject areas (like

plumbing, carpeting, etc.) People then look within the directory for the category of business that they need and select a business based on the information provided. In a similar manner, the 'service repository' lists the types of services and all the 'advertising' information about that service. Users connect to the service repository and present a request for a particular type of service, then choose the service that has the best fit for their needs. The difference between a 'yellow pages' type of repository and dynamic discovery is that the service repository operates in full real-time. If one of the services goes offline it should be immediately noted in the service repository. When a new service becomes operational, it is through the service repository that it is made available to the users on the network.

These last three concepts have dealt with things that the user of a service probably will never see, but which are nevertheless essential to the definition and running of a service. The next concept that we will examine is intimately involved with the end use of a service - the front-end, that with which the person using the service interacts.

Front Ends

The clients or users of the business services created within a SOA will interact with a front-end of the service. This is a piece of software that is the interface, i.e., the Web form into which a user enters data. The visible, user entry part of an application is what is termed the 'front-end.' A front-end can also be part of a composite application (a collection of services working together.) It is at the level of the front-end that the results of using a business service will also be displayed. Another term used for front-end is *client application*.

In a Service Oriented Architecture the front-end of a service may differ greatly between one type of user and another. The same service can be accessed via different interfaces, thus it is possible to have a language translation program running between the service and the front-end (or within the front-end) and presenting the service in a completely different language.

Examples of different front-ends for the same service are the many different ways that are now available to perform a banking transaction. One can access the transfer money service at a branch of the bank and use the 'teller front-end.' You can go to an ATM and the 'ATM interface front-end,' or you can logon to your account at the bank via the Internet and perform the transaction via the 'Web based front-end.' The Web based front-end is becoming increasingly popular as it is easy to reconfigure and personalize. One can even access the transfer funds service through cell phone text messaging and other cell based communication devices.

We examine some particular front-ends currently in use later in the book.

The technology that allows a front-end to work over a network invovles the sending and recieving of particularly defined messages. This message based technology iis a critical building block on a SOA.

Message Based Technology

Message based technologies are the reason why the services of a SOA are feasible today. Standard types of text messages between software users and providers have matured through the evolution of the Internet and the technologies now supporting it. We are all familiar with sending and receiving email messages, so the idea of messages being sent between application front-ends and services should come as no surprise.

Figure 6.5. Message based technology

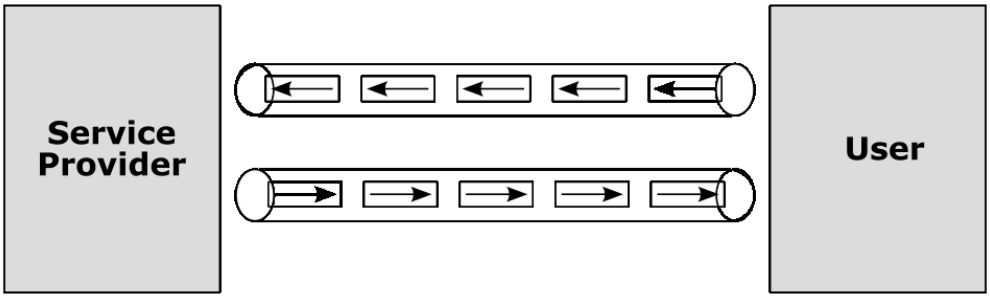
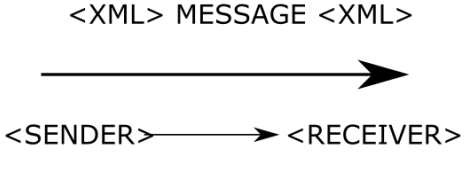


Figure 6.6. Message package



Service providers and consumers communicate via message packets. For many applications this messaging is as simple as sending specially formated text (like an XML file) over an Internet or local network connection. When a service becomes available a 'service contract' is documented. This supplies the user of the service with the standards adhered to when coding the service. The service contract defines the behavior of the service and the type of messages it accepts and returns. Because the interface contract is platform- and language-independent, the technology used to define messages must also be independent from any specific platform/language. Therefore, messages are typically constructed using XML documents (See the section called “XML” [54]) that conform to standard XML schema. XML provides all of the functionality, granularity and scalability required by messages. That is, for consumers and providers to effectively communicate they need a non-restrictive type of system to clearly define messages; XML provides this. Because consumers and providers communicate via standardized messages, the structure and design of messages should not be taken lightly. Messages need to be implemented using a technology that supports the scalability requirements of services. Having to redesign a message system will break the interface between services and users, which can be very costly.

In the Outdoors Company we would find XML messages being used to communicate content between the central database and the Web page that allows a user to configure a custom built bicycle. The user would be given a list of specific options available for each piece of the bicycle that is being created. These lists would come to the Web page from the database. Each of the data fields would be wrapped in an XML code that told the Web server program which part went into which section. When a particular component had an upgrade, the Outdoors company would only have to change the data in the database and not in all their Web pages because of this message based technology.

Messages have to travel from one place to another and the system that controls this process is the network infrastructure of the enterprise. With the concepts of loose-coupling and interoperability fresh in our minds it should come as no surprise that network technologies have grown and evolved to include many different and interconected network types and locations. The distributed network architecture is of particular interest to us as we examine these key concepts of SOA.

Distributed Network Architecture

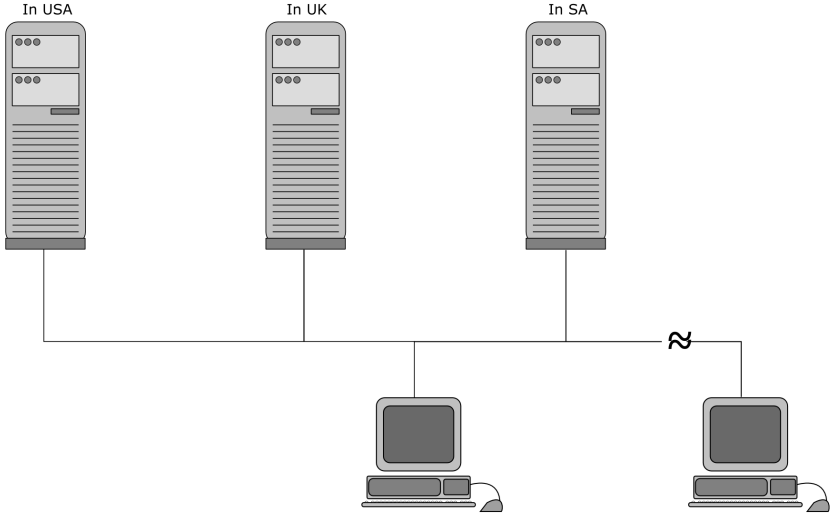
As we have explored the concept of the service and the aspects and properties that make it possible, we have seen continued references to networks. Local area networks the Internet, banking system networks and wide area networks have all

been mentioned or alluded to in discussion so far. The underlying technological concept of distributed network architectures underpins the availability of services within a SOA.

There are many different types of distributed computing systems and many challenges to overcome in successfully creating one within an enterprise. The main goal of a distributed computing system is to connect users and resources in a transparent, open and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems.

An example of a distributed system is the World Wide Web. As you are reading a Web page, you are actually using the distributed system that comprises the site. As you are browsing the Web, your Web browser running on your own computer communicates with different Web servers that provide Web pages. Possibly, your browser uses a proxy server to access the Web contents stored on Web servers faster and more securely. To find these servers, it also uses the distributed domain name system. Your Web browser communicates with all of these servers over the Internet, via a system of routers which are themselves part of a large distributed routing system.

Figure 6.7. Connections within a Distributed Network Architecture



A local intranet that connects users from many different offices, located in many different areas is another example of a distributed network. The actual computers may use any operating system and run many different programs, but the hardware backbone of the intranet is able to connect all the various systems and allow almost instantaneous communication over great distances. This becomes a distributed network when the resources are spread among the many computers, instead of being stored in a single location. The structure of the network is not located at one specific site, but is distributed throughout the network system.

The most widely known distributed network system is called the Internet. Many companies now use the internet as part of their own infrastructure and so have built services and systems that interact with and utilize the standards and processes of the Internet. These services are typically called Web Services and have been very popular with companies that are testing the concept of a SOA before fully implementing one within their enterprise systems.

Web Services

We have already discussed the Web based front-end of a service, but need to discuss Web Services in greater depth due to their increasing popularity and ease of use within a SOA.

When specifically defined, a Web Service is a software system designed to support inter-operable machine-to-machine interaction over a network. It has an interface described in a machine-processable format such as WSDL (See the section called “WSDL” [60].) Other systems interact with the Web Service in a manner prescribed by its service contract or description (i.e., using SOAP messages) typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. Software applications written in various programming languages and running on various platforms can use Web Services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between programming languages like Java and Python or between Windows and Linux applications) is due to the use of open standards.

Although the concepts behind SOA were established long before Web Services came along, Web services play a major role in a SOA. This is because Web Services are built on top of well-known and platform- independent protocols. These protocols include HTTP, XML, UDDI, WSDL and SOAP. We'll examine these in Chapter 7, *SOA Supporting Concepts and Technologies* [53]. It is the combination of these protocols that makes Web Services so attractive. Moreover, it is these protocols that

fulfill the key requirements of a SOA. That is, a SOA requires that a service be dynamically discoverable and evokable. This requirement is fulfilled by UDDI, WSDL and SOAP. SOA requires that a service have a platform-independent interface contract. This requirement is fulfilled by XML. SOA stresses interoperability. This requirement is fulfilled by HTTP. This is why Web Services lie at the heart of many SOAs.

The online booking services of many airlines are fully developed Web Services. They have been developed using the Web-based standards and security procedures necessary to their functionality. They are not only a front-end to the airline's booking system, but they perform many of the processes of that booking procedure independently. They are specifically designed for access over the TCP/IP network of the Internet. Other systems interact with these Web services (e.g. the SMS confirmation of bookings.) A Web Service is a service that has been specifically designed and created within the boundaries of the technology and standards of the World Wide Web.

From Key Concepts to Supports

All of the key concepts of a SOA that we have examined rely on many other supporting concepts, ideas, processes and technologies. They all work together to provide the system incorporated as a fully functioning SOA. These supporting concepts and technologies are also important to your understanding of what SOA is and how your enterprise could benefit from implementing a SOA. We discuss the supporting concepts and technologies in the next chapter.

This page intentionally left blank

Chapter 7. SOA Supporting Concepts and Technologies

Poles, Pegs and Ropes

In some sense, the services of a SOA are like the canvas of a tent. The main purpose of a tent is shelter, which is the purpose of the canvas. If you only have the canvas, however, you don't have a tent. You need the tent poles, the guide ropes, the pegs, the ground sheet and the various tools to put all of these things together. In a similar fashion it takes all the supporting concepts and technologies to create a usable and purpose-filled service.

There are many different ways to create and manage a service within the SOA concept. In this section we take a closer look at the technology and standards that are typically used in a SOA. Most of these technologies and standards are reliant on each other, so you may need to briefly read through each sub-section in order to gain a basic vocabulary. Many of the subsections will refer to technologies and standards defined either previously or later in this section.

The reason we have included all these supporting concepts and technologies is to give you a better understanding of them. With this information you will be better able to examine different SOA solutions and make your own decision as to their ability to meet the needs of your extended enterprise.

The 'tent poles, guide ropes and pegs' supporting SOA that we are going to examine include:

- XML
- XML Schemas
- Enterprise Software Bus (ESB)
- WSDL
- SOAP
- Grid Computing
- Ethernet
- Broadband
- GSM
- RFID
- Client Front-Ends

You may already have a very good understanding of many of these supporting concepts and technologies. Many of them have been buzz words within the ERP arena and business world. Some have been around for many years and are almost part of the furniture.

On of the 'tent poles' supporting SOA and bringing increased usability and interoperability is the technical framework of XML.

XML

XML is short for Extensible Markup Language, a specification developed by the W3C (World Wide Web Consortium. XML is a pared-down version of SGML (Standard Generalized Markup Language) and was designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation and interpretation of data between applications and between organizations

The Extensible Markup Language (XML) can also be defined as a **general-purpose** markup language for creating **special-purpose** markup languages. As such, it capable of describing many different kinds of data. Its primary purpose is to facilitate the sharing of data across different systems, particularly systems connected via the Internet. Languages based on XML (for example, RDF, RSS, MathML, XHTML and SVG) are defined in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.

The features of XML that make it well-suited for data transfer are its:

- Format is readable by humans and programs;
- Support for Unicode, allowing almost any information in any human language to be communicated;
- Ability to represent the most general computer science data structures: (records, lists and trees);
- Self-documenting format that describes structure and field names as well as specific values;
- Strict syntax and parsing requirements that allow the necessary parsing algorithms to remain simple, efficient and consistent.

XML is also heavily used as a format for document storage and processing, both online and offline and as such offers these benefits:

- Robust, logically-verifiable format based on international standards;
- Hierarchical structure suitable for most (but not all) types of documents;
- Manifests as plain text files, unencumbered by licenses or restrictions;
- It is platform-independent, thus relatively immune to changes in technology;
- It and its predecessor, SGML, has been in use since 1986, so there is extensive experience and software available.

If you use the internet for any business transactions, it is quite likely that your business data is transported via XML. Many document formats are now also utilizing XML as a base for storing information about the document, who wrote it, when it was written, how many times it has been edited and various other useful information. XML is a communication standard that is widely supported and has great potential for communicating various types of data.

For certain applications, XML also has the following weaknesses:

- Its syntax can be fairly verbose and partially redundant.

  This can hurt human readability and application efficiency and yields higher storage costs. It can also make XML difficult to apply in cases where bandwidth is limited, though compression can reduce the problem in some cases. This is particularly true for multimedia applications running on cellphones and PDAs which want to use XML to describe images and video.

- Some consider the syntax to contain a number of obscure, unnecessary features born of its legacy of SGML compatibility.

  This is unavoidable, however, as an effort to settle on a subset called "Minimal XML" led to the discovery that there was no consensus on which features were in fact obscure or unnecessary. The basic parsing requirements do not support a very wide array of data types, so interpretation sometimes involves additional work in order to process the desired data from a document. For example, there is

no provision in XML for mandating that "3.14159" is a floating-point number rather than a seven-character string variable. XML schema languages add this functionality. Modeling overlapping (non-hierarchical) data structures requires extra effort. Mapping XML to the relational or object oriented paradigms is often cumbersome. Some have argued that XML can be used as data storage only if the file is of low volume, but this is only true given particular assumptions about architecture, data, implementation and other issues.

XML usage within Web Services shows us that it's strengths are outweighing its weaknesses. The use of XML with the addition of schema languages XSL has created a data transfer system that increases inter-interoperability. It provides the basis for data standards and conversions.

Almost every Web based business application or technology produced in recent years uses XML in one form or another. Most of the data that is shared between companies is transferred via XML at one stage or another. Although you might not know it, XML is already in use within most operating systems and enterprise systems. This is one of the reasons that SOAs are able to be developed and utilized. XML is a dominant support technology for many of the innovations in recent communication innovations.

XML by itself is very useful, but its usability increases a hundredfold when coupled with an XML Schema. If we think of XML as being a tent pole, then we can certainly see XML Schemas as being the guide ropes that anchor the tent pole to the canvas and to the ground.

[For those readers who enjoy knowing trivial details, this book was originally written in XML, using the DocBook XML schemas. From the original XML document we can print a book, export it to a PDF, create a Web document, export a rich text document and export to other document formats very easily.]

XML Schemas

The W3C XML Schema Definition Language is an XML language for describing and constraining the content of XML documents. The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD (Document Type Definition.)

An XML Schema defines:

- Elements that can appear in a document;
- Attributes that can appear in documents;
- Which elements are child elements;
- The order of child elements;
- The number of child elements;
- Whether an element is empty or can include text;
- Data types for elements and attributes;
- Default and fixed values for elements and attributes.

XML Schemas are the successors of DTDs. It is possible that very soon XML Schemas will be used in most Web applications as a replacement for DTDs. Here are some reasons why DTDs are being replaced by XML Schemas:

- XML Schemas are extensible to future additions;
- XML Schemas are richer and more useful than DTDs;
- XML Schemas are written in XML;
- XML Schemas support data types;
- XML Schemas support namespaces.

XML by itself is like the audio data standard used to record CDs. Any noise can be recorded to a compact disc and played back on a CD player. XML Schemas are definitions and catalogs of what that digital audio recording contains. An XML Schema takes XML to a higher level of definition and abstraction, like a radio station's collection of music CDs, cataloged and stored in a specific manner. A radio station DJ needs to understand how the CD collection is stored in order to get the songs that will be played during a show. The DJ knows that the CDs are stored in

alphabetical order, but also indexed by a numerical sequence that allows an index to be kept listing all the songs under various categories (classical, pop, rock, etc.) Because the CD collection is specifically defined, the DJ can very quickly and easily find the music that he or she has selected. In a similar manner XML Schemas specifically define how data is cataloged, stored and indexed, making data retrieval easier, standardized and more usable.

One of the tools that utilizes its own XML and XML Schemas to perform functions is the Enterprise Software Bus. The Enterprise Software Bus may be viewed as another tent pole for the SOA canvas.

Enterprise Software Bus

The concept behind the enterprise software bus is that of controlled integration. An enterprise service bus is an emerging standard for integrating enterprise applications in an implementation-independent fashion at a coarse-grained service level (using key principles of SOA) via an event-driven and XML-based messaging engine (the bus.) An enterprise service bus generally provides an abstraction layer on top of an Enterprise Messaging System which allows integration architects to exploit the value of messaging without writing code.

Although the exact definition of an ESB varies, most agree that:

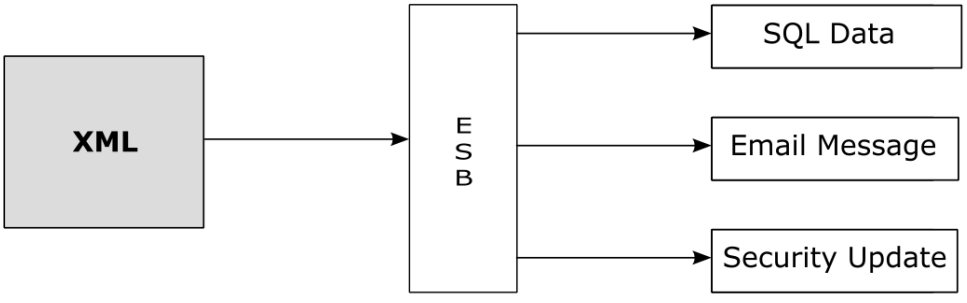
- It is a published enterprise standard - it is not merely specific to two or more systems or departments;
- It requires the clear separation of the message header and the message body;
- It is usually operating system and language independent (i.e., it should work between Java and .Net applications.);
- It often uses XML and Web services to transport messages;
- It includes adapter standards (such as J2C) for incorporating existing applications into the bus;
- It includes support for asynchronous processing;
- It includes intelligent, content-based routing;
- It includes a standardized security model to authorize, authenticate and audit use

of the ESB;

- It includes transformation services (such as XSLT) between the format of the sending application and the receiving application, including the transformation of data formats;
- It includes validation against schemas for sending and receiving messages;
- It can uniformly apply business rules, enrichment of the message from other sources, splitting and combining of multiple messages, and the handling of exceptions;
- It can conditionally route or transform messages based on a central policy;
- It is monitored for message latency and other characteristics described in a Service Level Agreement;
- It often facilitates "service classes," responding appropriately to higher and lower priority users;
- It supports queuing, holding messages if applications are temporarily unavailable;
- It handles a "publish and subscribe" messaging model, including event handling.

The benefits of using an ESB within a service-oriented architecture is that it provides faster and cheaper accommodation of existing systems, increased flexibility (easier to change as requirements change,) is standards-based and scales from point solutions to enterprise wide deployment.

Figure 7.1. An ESB translating messages



The golden dream behind the ESB is to replace proprietary integration brokers with open communication layers through which distributed services and business processes are readily exposed and easily managed. The immediate reality, however, is that it may be too soon to leave the old messaging subsystems behind. Regardless of the underlying messaging core, an ESB must somehow -- through open standards or by proprietary means -- create a foundation for reliable messaging. Until WS-* specifications for reliable messaging fall into place, that reliability continues to come from the likes of JMS (Java Message Service), homegrown messaging engines, proprietary MOM (message-oriented middleware,) and J2EE servers.

An ESB is a consummate diplomat layer. You can think of an ESB like a post office, taking all the messages from all the different sources and routing and repackaging them so that they are efficiently and effectively dealt with. For this reason a ESB is a complex piece of software in itself and choosing a ESB system needs to be done with great care. Many extended enterprise platforms, like SYSPRO e.net solutions, have an ESB like application or contain elements of an ESB within their system.

The next two technologies are specific XML formats used within particular frameworks. An ESB would most likely utilize one or the other, or both, as it interacts with other systems and messaging services.

WSDL

The Web Services Description Language (WSDL) is an XML format published for describing Web services. As of writing this book, version V 1.1 had not been endorsed by the World Wide Web Consortium (W3C); however, on May 11th, 2005

they released a draft for version 2.0 that will be a recommendation (an official standard) and thus endorsed by the W3C.

It is commonly abbreviated as WSDL in technical literature and often pronounced "Whiz-Dull." WSDL describes the public interface to the Web Service. This is an XML-based service description on how to communicate using the Web Service; namely, the protocol bindings and message formats required to interact with the Web Services listed in its directory. The supported operations and messages are described abstractly and then bound to a concrete network protocol and message format. WSDL is often used in combination with SOAP and an XML Schema to provide Web Services over the Internet. A client program connecting to a Web Service can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of an XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

Bearing in mind that the topics that we are now discussing are the supporting concepts and technologies for SOA, we could see WSDL as one of the tent pegs that keeps the concept of XML Schemas firmly grounded. We should also view SOAP as another tent peg with a similar supporting function.

SOAP

SOAP is a standard for exchanging XML-based messages over a computer network, normally using HTTP. SOAP forms the foundation layer of the Web Services stack, providing a basic messaging framework on which more abstract layers can build.

There are several different types of messaging patterns in SOAP, but by far the most common is the Remote Procedure Call (RPC) pattern, where one network node (the client) sends a request message to another node (the server,) and the server immediately sends a response message to the client. SOAP originally was an acronym for Simple Object Access Protocol, but the acronym was dropped in Version 1.2 of the SOAP specification. Originally designed by Dave Winer, Don Box, Bob Atkinson and Mohsen Al-Ghosein in 1998 with backing from Microsoft (where Atkinson and Al-Ghosein worked at the time,) the SOAP specification is currently maintained by the XML Protocol Working Group of the World Wide Web Consortium.

HTTP was chosen as the primary application layer protocol for SOAP since it works well with today's Internet infrastructure; specifically, SOAP works well with network firewalls. This is a major advantage over other distributed protocols like GIOP/IIOP or DCOM which are normally filtered by firewalls.

XML was chosen as the standard message format of SOAP because of its widespread acceptance by major corporations and open source development efforts. Additionally, a wide variety of freely available tools significantly eases the transition to a SOAP-based implementation. The somewhat lengthy syntax of XML can be seen as both a benefit and a drawback for SOAP. XML format is easy for humans to read, but can be complex and slow down processing times. GIOP and DCOM were able to use much shorter, binary message formats. On the other hand, hardware appliances are available to accelerate the processing of XML messages. Binary XML (The use of the word "XML" is controversial here.) is also being explored as a means for streamlining the throughput requirements of XML.

One can picture SOAP as a standardized addressed envelope that encloses a letter. When the letter is posted the machinery at the post office can easily sort it and send it to where it is meant to go. Because SOAP is widely accepted and known it has become one of the standards most used between clients and services on the Internet.

This brings us to the end of our discussion of the XML support group. As you have seen, the concepts and technologies are interrelated. The basis of XML is built on to create other core concepts that support and maintain a SOA . The next concept we discuss, Grid Computing, does not have to be utilized within a SOA, but its use (and the concept behind it) will bring greater flexibility and speed to the SOA, things that are crucial to the running of any real-time service.

Grid Computing

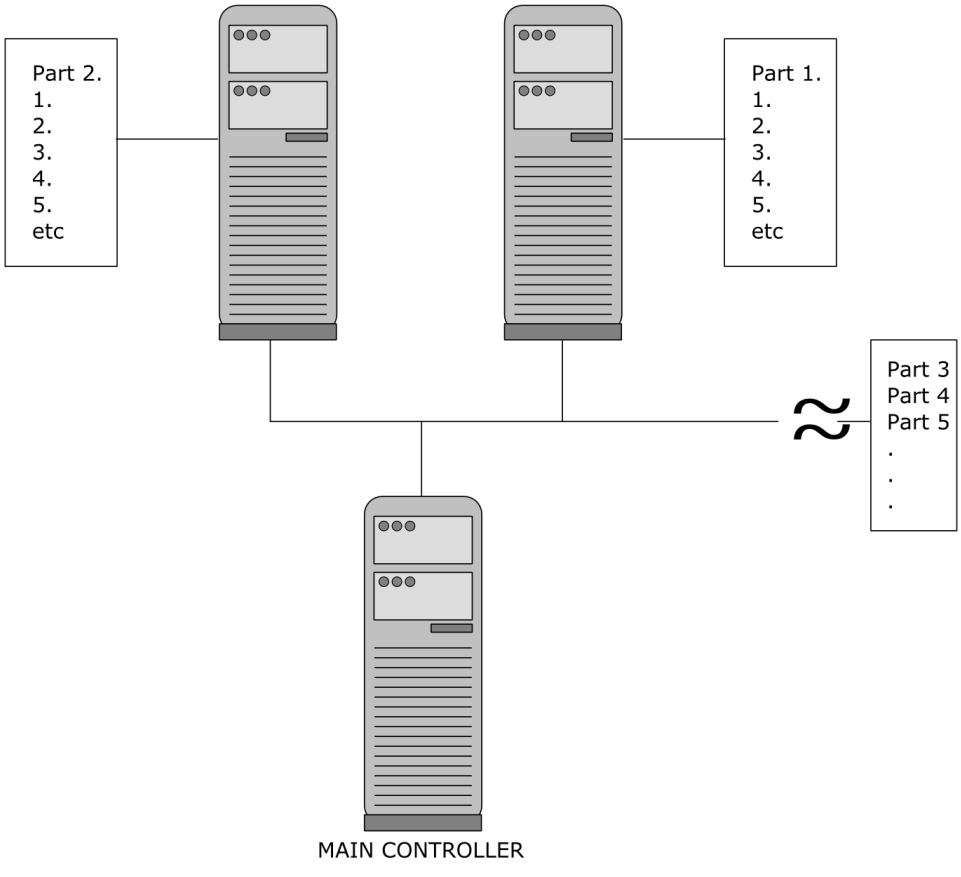
In their article at informit.com Joshy Joseph and Craig Fellenstein (writers of the book *Grid Computing*,) tell us that "in today's pervasive world of needing information anytime and anywhere, the explosive Grid Computing environments have now proven to be so significant that they are often referred to as being the world's single and most powerful computer solutions." By this we know that grid computing is important and will become even more so as the concept of grid computing gains greater understanding and support in the business world. But what is "Grid Computing?"

Grid Computing (or the use of a computational grid) is applying the resources of many computers in a network to a single problem at the same time - usually to those scientific, business or technical problems that require a great number of computer processing cycles or access to large amounts of data. A well-known example of grid computing in the public domain is the SETI (Search for Extraterrestrial Intelligence) @Home project in which thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of "rational" signals from outer space.

Grid computing allows companies to use a large number of computing resources on demand, regardless of where they are located.

In today's complex world of high speed computing, computers have become extremely powerful compared to that of (let's say) five years ago. Even the home-based PCs available on the commercial markets are powerful enough for accomplishing complex computations that we could not have imagined a decade ago. The quality and quantity requirements for some business-related advanced computing applications are also becoming more and more complex. The industry is now realizing that we have a need and are conducting numerous complex scientific experiments, advanced modeling scenarios, genome matching, astronomical research, a wide variety of simulations, complex scientific/business modeling scenarios and real-time personal portfolio management. These requirements can actually exceed the demands and availability of installed computational power within an organization.

Figure 7.2. Different parts of a process within a Grid



A Grid computing environment is created to address resource needs; the use of these resource(s) (i.e., CPU cycles, disk storage, data, software programs, peripherals, etc.) is usually characterized by their availability outside of the context of the local system. This 'external provisioning' approach entails creating a new administrative domain referred to as a 'Virtual Organization' with a distinct and separate set of administrative policies (where the home administration policies plus external resource administrative policies equals the Virtual Organization's administrative policies.) The context for Grid processes is distinguished by the requirements created when operating outside of the home administrative context. Grid technology

(sometimes also called middleware) is employed to facilitate formalizing and complying with the Grid context associated with the execution of an application or service.

Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation, or it can be a public collaboration (in which case it is also sometimes known as a form of peer-to-peer computing.) One characteristic that currently distinguishes Grid computing from distributed computing is the abstraction of a 'distributed resource' into a Grid resource. One result of abstraction is that it allows resource substitution (like swapping out a whole computer or upgrading it) to be more easily accomplished. Some of the overhead associated with this flexibility is reflected in the middleware layer and the time factor latency associated with the access of a Grid (or any distributed) resource. This overhead, especially the added time delay to access the grid resource (temporal latency,) must be evaluated in terms of the impact on computational performance when a Grid resource is employed.

The Grid Computing discipline involves the actual networking services and connections of a potentially unlimited number of computing devices within a "grid." This new innovative approach to computing can be most simply thought of as a massively large power "utility" grid, such as those that provide power to our homes and businesses each and every day. This delivery of utility-based power has become second nature to many of us worldwide. We know that by simply walking into a room and turning on the lights, the power will be directed to the proper devices of our choice for that moment in time. In this same utility fashion, Grid Computing openly seeks and is capable of adding an infinite number of computing devices into any grid environment, adding to the computing capability and problem resolution tasks within the operational grid environment.

One of the technologies that Grid Computing uses as its backbone is also a technology that a SOA would use as its backbone. Grids and Services require a network and the technology behind most networks is that of 'ethernet.'

Ethernet

Ethernet is the most widely-installed local area network (LAN) technology. Specified in a standard, IEEE 802.3, Ethernet was originally developed by Xerox and then developed further by Xerox, DEC and Intel. An Ethernet LAN typically uses coaxial cable or special grades of twisted pair wires. Ethernet can run at 10 megabytes per second (Mbps,) 100 Mbps (most common today) or ever 1 gigabyte per second of transfer.

Ethernet has become so popular that a specification for "LAN connection" or "network card" implies Ethernet without saying so. Ethernet allows connection to a company or home network as well as to a cable modem or DSL modem for Internet access. Today, almost all PCs and Macs come with 10/100 Ethernet ports which connect internally to a network adapter (NIC) that is built onto the motherboard. Ethernet can be added to older machines by plugging in a network adapter (NIC) into an empty PCI slot.

Most businesses that have a local area network with a server and workstations use Ethernet connections. Most shared Internet connections also use Ethernet connections. For this reason Ethernet is the main network enabling technology.

The speed of ethernet connections has been increasing as newer technical innovations and standards become part of the technology. With the increase in ethernet speeds comes the discussion of bandwidth and the innovation of broadband connections.

Broadband

Broadband comes from the words "broad bandwidth" and is used to describe a high-capacity, two-way link between an end user and access network suppliers capable of high volume data flows. This typically describes connections to networks having bandwidths significantly greater than that found in basic telephone networks. Broadband systems are capable of carrying a large number of moving images or a vast quantity of data simultaneously. Broadband techniques usually depend on coaxial or optical cable for transmissions and utilize multiplexing to permit the simultaneous operation of multiple channels or services on a single cable. Frequency division multiplexing or cell relay techniques can both be used in broadband transmission.

Broadband also describes a network connection that supports a very high bit rate, as

opposed to "narrowband," which supports a lower bit rate. The higher the bit rate (a measure of speed of transmission of bits per second) the faster the transmission will occur in a given period of time. Delivered via DSL or cable service, bandwidth: >128 Kilo Bits per second (KBPS.) Broadband Internet can exchange more data per second than a narrowband connection (dial-up < 57.6 kbits or ISDN <128 kbits.)

Using broadband connections, vast amounts of data can be exchanged quickly, thus increasing connectivity and enabling quicker access to services and data.

Broadband connections are becoming the data channels and Communication channels of the smart enterprise and so are more than an integral to a SOA. The next technology, Global System for Mobile Communications, is also one that is becoming more and more part of the tools used by an enterprise to communicate. For this reason it is a supporting technology, used by a SOA

GSM

Global System for Mobile Communications (GSM), is the most widely used digital mobile phone system and the de facto wireless telephone standard in Europe. Originally defined as a pan-European open standard for a digital cellular telephone network to support voice, data, text messaging and cross-border roaming, GSM is now one of the world's main 2G digital wireless standards. GSM phones are used by over a billion people across more than 200 countries. The ubiquity of the GSM standard makes international roaming very common with "roaming agreements" between mobile phone operators. GSM differs significantly from its predecessors in that both signaling and speech channels are digital, which means that it is seen as a second generation (2G) mobile phone system.

Access to the GSM networks via text messaging (SMS) and voice messaging enables businesses to connect instantly with their users or clients to present small amounts of data. Bank transfer statements, access codes, delivery schedules, special product promotions and any other text information can be easily communicated this way.

GSM together with other technologies is part of an evolution of wireless mobile telecommunication that includes High-Speed Circuit-Switched Data (HCSD,) General Packet Radio System (GPRS,) Enhanced Data GSM Environment (EDGE,) and Universal Mobile Telecommunications Service (UMTS.)

Broadband and GSM are communication technologies used as part of a SOA. Radio Frequency Identification is another communication technology that can be used, but

for identification of products within a SOA rather than communicating data or information between systems or users.

RFID

RFID is short for Radio Frequency Identification and refers to the technology that uses devices attached to objects that transmit data to an RFID receiver. These devices can be large pieces of hardware the size of a small book, like those attached to ocean containers, or very small devices inserted into a label on a package. RFID has advantages over bar codes such as the ability to hold more data, to change the stored data as processing occurs, to transfer data without line-of-sight requirements and to be very in harsh environments where bar code labels won't work. The heart of this technology is the RFID tag, a small computer chip with a miniature antenna. It can be used on transport and goods packages as well as sales units and products. When the chip receives the radio signal from a reading device, it automatically transmits the stored data wirelessly, as the system is based on radio frequencies. The chip does not need a battery because it is powered solely by the radio waves which are transmitted by the reading and writing devices.

Although first used in World War II to identify friendly aircraft, RFID technology began to materialize in the 1980s and 1990s. In 1997, Mobil introduced its Speedpass system that lets you wave the tag in front of the pump to record your transaction and debit your credit card. Automatic highway toll collection systems are RFID and General Motors' OnStar vehicle tracking system is a satellite-based version of it. RFID systems can be used just about anywhere, from clothing tags to missiles to pet tags to food -- anywhere that a unique identification system is needed. The tag can carry information as simple as a pet owner's name and address or the cleaning instruction on a sweater to complex as instructions on how to assemble a car. Some auto manufacturers use RFID systems to move cars through an assembly line. At each successive stage of production, the RFID tag tells the computers what the next step of automated assembly is.

One of the key differences between RFID and bar code technology is RFID eliminates the need for line-of-sight reading on which bar coding depends. Also, RFID scanning can be done at greater distances than bar code scanning. High frequency RFID systems (850 MHz to 950 MHz and 2.4 GHz to 2.5 GHz) offer transmission ranges of more than 90 feet, although wavelengths in this range are absorbed by water (the human body) and therefore have limitations.

RFID is enabling greater amounts of information to be gathered automatically along the various stages of a supply chain and is also enabling a greater control and security within distribution and delivery systems.

According to Lionel Carrasco in his article at ebizq.com (The Convergence of Wi-Fi and RFID,) we're also seeing another trend emerge - the convergence of active RFID and Wi-Fi. Unlike the systems mentioned above, *active RFID* involves tags with an energy source (batteries) that emit a signal to readers some distance away. The readers process the signal and determine the location of the tag, based on its presence in a particular zone or using triangulation techniques. To simplify, this kind of system provides an almost precise location of tags by coordinates within a particular physical space, very similar to GPS, but designed for use inside or in neighboring open spaces, such as parking lots or courtyards.

Active RFID technology has been available for years, using proprietary frequencies requiring their own equipment and infrastructure. But the convergence we're seeing today, based on technology developed by AeroScout, Ekahau, Cisco and others, is based on active RFID systems that use the same frequency as Wi-Fi networks, allowing end users to capitalize on existing infrastructure for wireless data networks. Not only does this reduce the overall implementation cost, but it also increases the advantages of location services for goods and people and ultimately streamlines processes in a range of industries.

Going down a layer, companies can use RFID and Wi-Fi technology to learn more about their assets than simple physical location, such as entry and exit, shortage or overflow, out of flow, friend or foe, status and more. That data can then be processed and sent to business applications, such as SYSPRO's supply chain module, or to generate notifications and alerts by e-mail, text message, PDA, etc. These "event processing platforms" have a high volume processing capacity for "location reports" that can reach up to several dozen million with thousands of tags generating information every minute, every second or even more frequently.

There is one more communication concept that we will discuss as part of this supporting concepts and technologies section, the client front-end. In many ways each of the preceding concepts and technologies is also a supporting concept or technology for the client front-end, primarily because a client front-end is a subset of the SOA front-end.

Client Front Ends

In the second section of this chapter we discussed the concept of a front-end to a service and saw the example of the various different ways to access the transfer money service from a bank. In order to gain a better understanding of how front end clients support services we will briefly examine the various client front ends currently in use.

Rich Clients

A rich client is a small piece of software that runs on the client to leverage and aggregate back-end Web Services, allowing them to appear as a single, unified, native application. A rich client runs these small applications on the desktop to work with data locally or draw on outside resources. It can communicate with a server, telling it to accept uploaded data or to send real-time data that users need. It also can reconfigure the part of the browser in which the user is working, downloading additional elements to expand the application where the user is showing interest.

Rich clients provide capabilities that thin clients are not able to, including windowing features and data navigation control such as buttons, check boxes, radio buttons, toggles and palettes. They are also able to integrate content, communications and platform-independent application interfaces for distribution through emerging SOAs. The rich client becomes a Web Services terminal of sorts, allowing applications to communicate and even execute on one another within a distributed environment.

Web Browser Clients

A Web Browser client is simply an interface to a service that is presented through the use of Web pages. This is also known as thin-client technology. Users simply log into a Web page (either on the enterprise intranet or on the Internet) using the Web browser on their machines. All the application work is handled by the server.

Web browser clients are ideal for services that require low processing capacity and many users. A cost saving benefit of using Web browser clients is that the user does not need high end computational power. Thin client machines and older computers can be used without slowing down the processes. All major operating systems come with a default Web browser, so no additional software has to be installed on the client side of the service.

Email Clients

An email client is a computer program that is used to read and send email. Protocols supported by email clients include POP3 and IMAP. IMAP and the updated IMAP4 are optimized for storage of email on the server, while the POP3 protocol generally assumes that the email is downloaded to the client. The SMTP protocol is used by most email clients to send email.

Email clients can be used to send specific information automatically. For example, an email client can be used to inform that a deposit or withdrawal has been made from an account. Email clients are usually just an information sending service.

WAP Clients

The Wireless Application Protocol is a standard for Internet access (Web, e-mail) from mobile phones. With WAP technology mobile phones are able to browse the Internet and so WAP clients allow users to use their mobile phones to access Web browser client type services.

SMS Clients

Short Message Service is used on digital GSM phone networks for sending text messages of up to 160 characters. SMS services are used in a similar fashion to email client services. When connected to an enterprise SOA, SMS messages can be automatically sent to clients informing them of the status of an order, a change in the pricing structure, or any other short piece of information that does not require confirmation.

A Whole Tent

The client front-ends and the other concepts and technologies discussed in this chapter provide the supporting framework of a SOA. Having read through the key concepts of a SOA and the discussion on supports, you have a better understanding of SOA, what it is, how it works and what it involves. Let's put this knowledge into practice by discussing the benefits and challenges of SOA . As we do this in the next chapters, use the information to evaluate the needs and capabilities of your enterprise in relation to implementing a SOA.

This page intentionally left blank

Chapter 8. Benefits of SOA

Introduction

In order for you to evaluate SOAs for yourself and for your enterprise you need more than just an understanding of what SOA is. You also need to be able to examine the possible benefits of using SOA within an enterprise.

In this chapter we will deal with SOA benefits from two different perspectives,

i. Business perspective
ii. IT perspective

We have split the benefits between these two perspectives because they are the two different aspects of the enterprise that are involved in migrating to a SOA. On many of these issues the benefit to one will also be a benefit to the other, just viewed from a different angle. For example, the increased reusability of code (an IT benefit) results in a greater return of investment (a business benefit.) According to a report by Gartner (*How Service-Oriented Architecture Will Affect SMBs* [gartner_soa_smbs] [?]) small and mid size businesses should view SOA as a natural, cost-effective progression from the client/server IT architectures that they are currently using. Migrating to SOA will help them serve their customers better and save them money by bringing processes closer together. We begin with the business benefits that implementing a SOA can bring to your enterprise.

The Business Value of SOA

In the May 2005 Gartner report,*Gartners' Positions on the Five Hottest IT Topics and Trends in 2005* [gartner_five_hottest] [249], they state that SOA and Web Services will affect every business and IT department. Businesses that successfully implement a SOA in their business environment will have a competitive edge over those that do not. They have this edge because those who have services aligned with strategic business goals can react more quickly to changing business requirements than those who just have IT systems aligned to a particular execution environment. It is easier to combine services, easier change individual service compositions and cheaper to change data flows to and from services than it is to change the executing environment.

SOA focuses on streamlining computerized business processes, which results in saving money and reusing IT assets. According to Gartner, its impact on SMBs will be huge as they buy packaged applications, develop custom ones and deploy both types. Gartner expects that through 2010 "over 80 percent of SMBs will make their business processes at least 15 percent more efficient by deploying SOA and SOBAs (service-oriented business applications .)"

Let's examine some specific benefits of a SOA from a business perspective, starting with business agility:

Agility

Business agility is by far the most important business benefit of migrating to a SOA. According to Eric Newcomer and Greg Lomow, authors of *Understanding SOA and Web Services* [newlom] [249], business agility "in the form of quickly responding to new business requirements and opportunities," is now more important than development productivity. Agility in this sense means faster time-to-market. SOA achieves this through dramatically reducing the amount of time required to assemble new business applications from existing services and IT assets. Agility also means being able to adapt the IT system wherever necessary.

In conventional monolithic enterprise systems, data and requests are exchanged via multiple technologies and a tangle of complex connections. The resulting dependencies between the systems are so numerous that any attempt at reconfiguration is often extremely time consuming and costly. In SOA business agility is provided by an increased granularity (breaking up into separate pieces) of processes enabled through the services. This results in the ability to quickly create business processes and composite applications (an application made up of ready made parts) to respond to changes in the marketplace. It also creates improved customer service as one can use services without having to worry about their underlying IT infrastructure. The improvements gained by introducing interoperability and integrating systems results in a greater ability to respond more quickly to change.

Let's examine how SOA can bring greater business agility by looking at a few scenarios:

- Agility - from finding the right service.

  Every business is reliant on the services of other businesses. It does not matter whether a particular service was found by word of mouth or by looking in the yellow pages, the vital detail is whether the right service was found quickly and efficiently. The same principle applies to services provided in a SOA - it is vital to be able to quickly and efficiently locate the right service, whether it is provided by another department or an outside partner.

- Agility - when changing service providers.

  It is sometimes necessary to change service providers. Possibly the service provider that you were using went out of business, was bought out or stopped offering the particular service that you were using. With SOA it is possible to quickly and efficiently locate an alternative service provider that provides a similar or equal capability that suits your needs and your budget.

- Agility - quickly assembling services into applications.

  Remember that an agile business can respond to new business opportunities and threats quickly. SOA Services are designed with abstract interfaces. They are not tied to a single business process, so they can be assembled (or reassembled) into new applications quickly and easily

- Agility - by supporting new service requesters and new delivery channels.

  Businesses often grow by offering their existing services to new customers through new or alternate delivery channels. A business may launch their existing services through a new web site or through a new mobile device front end. Services that are loosely coupled and platform independent can be quickly adapted to these new types of business channels.

- Agility - dynamic ability to adjust capacity.

  Every business experiences change in demand. Whether it is a spike in demand due to an event or a long term trend, the agile business is able to add or take away capacity as needed. Dynamic provisioning is possible in a SOA because the services represent discrete business functions that are dynamically discovered

and located independently. You can increase or decrease your usage of a service without major change.

- Agility - using existing services to support new requirements.

  Because SOA services are designed with an abstract interface that is not tied to a single business process, they can be easily adapted to support new and unforeseen business requirements.

SOA helps business owners to take a more direct, active role in the design of IT systems or, at the very least, the business processes enabled via IT systems. This enables IT systems to more accurately reflect business processes and the organization, improves IT systems' capability to adapt as processes change and increases the likelihood of reuse, which potentially improves productivity and process consistency.

The next business advantage offered by SOA that we will discuss is a better return on investment. A better ROI comes from many places within a SOA and as such is a general benefit.

Better Return on Investment

Any business serious about increasing or bettering a return on investment knows to examine a purchase or a change in procedure over the course of time. The return on investment benefits of SOA are not immediate, but they are significant over the course of a few years. According to Eric Newcomer and Greg Lomow [newlom] [?], "A SOA dramatically improves the ROI of existing IT assets by reusing them as services in the SOA." In this way, migrating to SOA will help maximize the value of existing IT investments while minimizing the risk.

The creation of a robust service layer also has the benefit of a better return on the investment made in the creation of the software. Because the services map to distinct business domains, they also have a greater useful life. For example, a company might create an inventory service that has all of the methods necessary to manage the inventory for the company. By putting the logic into a separate layer, the layer can exist well beyond the lifetime of any system into which it is composed. If your organization needs to create a credit card authorization service, there are basically two options. Developers will either develop the functionality as part of the application that needs it, or they will develop it as a separate component. If credit

card authorization is developed as a separate component and used as a service, then it is likely to outlive the original application.

The business value of reduced development times, increased reusability, greater scalability and specialized IT function all result in a better return of investment, as does the next benefit that we discuss, service assembly.

Service Assembly

This benefit is very closely tied with the IT benefit of code reuse and the idea of the composite application, but because it focuses more on a business' customers, we have decided to discuss it here.

When a business migrates onto a SOA, the services that developers create will become part of a catalog of reusable services. Customers who use these services will come to understand this catalog as a set of reusable assets, and they will find different ways that the services can be combined, ways not conceived by their original programmers. As these services are assembled in different ways, everyone will benefit from the new applications. The new and different assembly of the services will also allow custom applications to be developed more quickly as a result of this catalog of reusable services.

The synergy created by service assembly will also increase customer satisfaction and innovation and bring about a better return of investment.

The use of services that can be combined and assembled in different ways due to loose coupling and increased interoperability results in lower development costs.

Lower Development Costs

SOA's ease of integration - specifically the ability to reuse proven business functionality and the reduction of complexity through loose coupling - considerably lowers development costs. In fact, it reduces the development of new business solutions to an assembly exercise that doesn't require the complex technical skills needed when creating and programming from scratch.

The development costs of a new application are reduced because legacy applications can be incorporated into the new system. Code reuse service reuse and the ability to function from many different front-end clients all bring the total cost of development lower.

When looking at development costs, it is also important to look at the development cycle over a period of time. In legacy enterprise systems most of the IT focus was on patching and fixing the application, instead of looking for better ways to code or perform a business function. With SOA a development team can create and test a new service function without crashing an application and can improve existing services offline, even while the particular service is being used online.

The use of services that can be combined and assembled in different ways due to loose coupling and increased interoperability not only lowers development costs but brings the benefit of 'future proofing.'

Future Proofing

In traditional enterprise software systems, vendor and technology 'lock-in' can occur on many levels. The ERP system that you utilize depends on a particular operating system and cannot be used interchangeably with other ERP systems. The application platform (.NET, Java, Oracle,) the middleware used (CORBA, WebSphere, Tibco,) and the specific nuances of the particular program all lock you in to continued use of those products because of the great expense of changing. SOA "future proofs" an organization and dramatically reduces vendor and technology lock-in.

Newcome and Lomow (*Understanding SOA and Web Services*) explain this benefit by saying; "A SOA-centric organization bases its fundamental IT architecture on service contracts, which are aligned to business-level services and which are technology-neutral, application independent and middleware-agnostic." Within a SOA environment it doesn't matter what you used to use to perform a particular business function. SOA decouples the business process from the underlying operating system and technology, making it much easier to replace applications and technologies, without having to worry about middleware programs. It is entirely possible to replace an in-house or external service provider when needed.

We have seen that services of a SOA (combined and assembled in different ways due to loose coupling and increased interoperability) have brought many of the above benefits to the business enterprise .These properties of a service also allow for better scalability.

Better Scalability

One of the requirements of a service-oriented architecture is location transparency. To achieve location transparency, applications look up services in a directory and bind to them dynamically at run-time (dynamic discovery.) This feature promotes better scalability since a load-balancing program may forward requests to multiple instances of the service without the knowledge of the service client or front-end application. A business can start small with only one instance of the services needed, then as growth occurs they can increase those instances in line with that increased need.

This benefit is very important for small and medium sized businesses (SMBs.) They can start small, using or migrating one or two services. Then, as their businesses grow they can add greater capacity in a step by step manner. This means that SMBs budgets do not have to include full capacity of the projected service needed at the beginning of a migration to SOA, but can be increased incrementally.

The interaction between what a business needs and what a SOA offers has been described as a business benefit. We call this 'business alignment.'

Improved Business Alignment

Alignment happens when all business functions support common goals and outcomes (Newcomer and Lomow p96 [newlom] [249].) The services of a SOA can (and should) be defined so that they directly support the product that the organization provides to customers, clients, citizens and partners. When the services within a SOA are created from the business context of that enterprise, then there is improved alignment between IT and the core business function.

SOA environments make business logic more available. Due to location transparency, multiple servers may have many instances of a service running on them. In addition to this, if a network segment or a machine goes down, a dispatcher can redirect requests to another service without the client's knowledge.

SOAs also motivate the creation of services that are more meaningful within a business process. These are therefore more accessible to the end business user.

One of the direct results of better aligning IT with business goals and outcomes is an improvement in the function of the business. This in turn results in improved service to the customer.

Improved Customer Satisfaction

There are many ways that SOA improves customer satisfaction.

The benefits of SOA to a manufacturing system bring reduced waiting periods for products, resulting in greater customer satisfaction.

Improved access to a company's services and products through SOA enabled Web Services enhance customer satisfaction.

Customer-centric SOAs strive to ensure a consistent customer experience, regardless of what front-end customers are using. It doesn't matter whether a customer accesses the service in a face-to-face interaction, a Web self-service, a call center, at an ATM, from an information kiosk or on their mobile phone, the service provides a consistent experience. Newcome and Lomow *(Understanding SOA and Web Services*) tell us that "customer satisfaction drops when a customer gets contradictory answers to the same request depending on the access mechanism that they use", so by using the same SOA service for all the access mechanisms, customer satisfaction is improved.

The agility of a business due to SOA means that a business can meet a customer demand faster and at less cost than before, resulting in improved customer satisfaction.

The potential for improved customer satisfaction is contained in every other benefit, even if customers are not aware of the effect that benefit has on them or the product. Improved customer satisfaction brings greater customer retention and higher customer loyalty.

Summary

The biggest benefit from migrating to a SOA is the increase in business agility. Many of the other benefits are simply a result of better business conditions. By examining the benefits of SOA from a business perspective, we can see why a business that has successfully integrated a SOA within their enterprise environment will have a competitive edge over those businesses that do not.

The business benefits are not the only reasons why one should consider migrating a business to a SOA. The benefits from an IT perspective must also be examined and understood so that you can evaluate whether or not SOA is for you.

The IT Benefits of SOA

Having looked at the benefits of SOA from a business perspective, we will now focus on benefits from an IT perspective. SOA must be business driven, but it must also be IT enabled. For this reason it is important for you to consider these benefits and to consider which ones might be applicable to your situation, your business or your enterprise. We start our discussion with the benefit of code mobility.

Code Mobility

According to Gartner (in the Web article *Benefits and Challenges of SOA in Business Terms* [gartner_benefits_challenges] [249]) with a SOA approach "the emphasis of application development projects shifts from creating implementation-specific functionality and code toward focusing on creating stable, self-contained software modules that interact via a standards-based software infrastructure." Since location transparency is a property of a SOA, code mobility becomes a reality. The lookup and dynamic binding to a service means that the client does not care where the service is located. Therefore, an organization has the flexibility to move services to different machines or to move a service to an external provider. This frees IT managers from location specific dependencies.

Code mobility is not only from a service location perspective, but also from a code re-use perspective. The code used for one service within an application becomes an independent portion of service code when migrated to a SOA platform. The code can be re-used, modified and applied to different situations depending on the needs of the enterprise.

For example, when the code for a credit check function within a business process migrates to a SOA and is incorporated into an independent service, it doesn't matter where the code is located, as long as the service is dynamically discoverable. Any application can utilize that code without having it incorporated within the structure of the particular application. The credit check service can be moved from one computer to another and it can also be hosted by many different computers, each one running a certain number of instances of the 'credit check' service.

Code mobility also contributes to the next benefit that we discuss, greater reuse of code.

More Code Reuse

Code reuse has been the most talked about form of IT cost saving and product reuse over the last four decades of software development. Unfortunately, it is hard to achieve due to language and platform incompatibilities. Component or service reuse is much easier to achieve. Run-time service reuse is as easy as finding a service in the directory and binding to it. The developer does not have to worry about compiler versions, platforms and other incompatibilities that make code reuse difficult.

According to Simon Hayward of Gartner (*Findings From the 'All Software' Research Meeting: The Challenge of Service Optimization in SOA* [gartner_findings] [249],) "Service-oriented architecture (SOA) promises reuse, but this is much more dependent on the quality of service design (such as which elements to include) than on the characteristics of the underlying technology." It is not enough to wrap a legacy function in new code that makes it into a service that works within your enterprise system. Services must be inter-operable and dynamically discovered. They have to work on any operating system and be found by the applications and front-ends regardless of where they are being hosted on the network.

The benefits to IT of more code reuse are faster development cycles, reduced development costs, increased stability (due to the better maintainability of the code of a service,) and easier cross-team development.

Another IT benefit that relates to coding and programing involves the focus of the developers roles at each stage of a service of composite application's development.

Focused Developer Roles

A SOA will force an application to have multiple layers (service layer, application/front-end layer, communication layer, data access layer, etc..) Each layer has a set of specific roles for developers. For instance, the service layer needs developers who have experience in data formats, business logic, persistence mechanisms, transaction control, etc. A client developer needs to know multiple technologies, such as SWING, JSP or MFC. Each layer requires a complex set of skills. To the extent that developers can specialize, they will excel at their craft in a particular layer of the application. Companies will also benefit by not having to rely on highly experienced generalists for application development. They may use less experienced developers for focused development efforts.

As more and more enterprises migrate toward SOA, more and more IT developers

will have more focused roles in the development cycle. They will no longer have to be a 'jack-of-all-trades,' but will be able to focus on an area of specialty. Most IT developers are used to working in teams. Even with higher specialization, they will still need to work as part of a team, but they will have a different function. They will be active in a particular part of the development and not in other parts. This creates a more efficient use of skills and talents.

Multiple layers of development also contribute to the IT benefit of better parallelism in the development process.

Better Parallelism in Development

The benefit of multiple layers of development in SOA means that different developers can work on those layers independently. Developers can create interface contracts at the start of a project and they will then be able to create their parts independently of one another. Parallel development decreases the time needed to create and test a coding project and allows for greater flexibility of specialized staff.

Better Testing/Fewer Defects

Services have published interfaces that can be tested easily by developers by writing unit tests. Developers can use tools such as JUnit for creating test suites. These test suites can be run to validate the service independently from any application that uses the service. It is also a good practice to run the unit tests during an automated build process. There is no reason for a quality assurance tester to test an application if the unit tests did not complete successfully. More and better testing usually means fewer defects and a higher overall level of quality.

In the Outdoors Company, the service that lists the various bicycle parts that can be used to construct a custom made bike would have gone through its own testing cycle. The service programmers would have tested the network and database access functions within the service and would have made sure that they were working, before the Web front-end programmers incorporated the service into the web site and tested its functionality there. By the time a quality assurance tester logged on the web site and tested the service, the service had already been tested (and corrected when necessary) many times.

One of the end results of the multiple layers of development and focused developer roles is the SOA advantage of a service being able to interact with many different types of client front-ends. The client front-end is just another layer of the development.

Support for Multiple Client Types

By using a SOA, companies may use multiple clients and multiple client types to access a particular service. A PDA using Java technology may access a service via an Internet connection and a mobile phone client may access the same service via a text or SMS connection. Since the layers are split into client and service layers, different client types are easier to implement.

The benefits of access to a service by multiple client types are many. Customers from many sectors of the population will have greater access to the particular service. There will be a greater accuracy of information given by a single service enabled over multiple client types, as opposed to many proprietary services for each client type (each of which could have an error or bug in its system.)

The delivery tracking service of a company like the Outdoors Company would benefit greatly from support for multiple client types. Once an order has been made and the process of assembling the bicycle has begun, the client should be able to access the delivery service information from many different sources, not just on the web site from which the order was made. If an order number was given at the completion of the online order process, this number could be used to access the service via email, web site, mobile phone and even via a voice system on the telephone. This way customers are able to get information about when their orders will be ready and delivered via a medium that best suits them.

Support for multiple client types requires the multiple layers of development discussed earlier in this section. Through the adoption of SOA code, development becomes more modularized. Having various layers of code that interact with each other also increases the maintainability of that code.

Better Maintainability

The maintainability of code over a period of time has always been a hassle to programmers and developers. Updates and bug reports continue to be published for all major operating systems and business productivity suites. There is a job function known as 'software archeology' that defines the task of locating and correcting

defects in code. Users of the software also contribute toward finding bugs and errors and security risks.

By focusing on the service layer as the location for your business logic, maintainability increases because developers can more easily locate and correct defects. The code for a service is smaller than that of a whole application, so by adopting a SOA you already improve the chances of finding bugs and errors in the code. It is easier to debug the code for a service than it would be to locate the bug within the code for a large application. The function of software archaeologists is reduced and made easier by the migration to a SOA.

There is one more benefit that we need to discuss and it is related to the code having better testing, fewer defects and better maintainability. The services that are developed through an SOA will have more options for security than any single application would.

More Security

The creation of a service layer by definition means that developers have created an additional network interface that can be used by multiple applications. When systems were built using monolithic or client-server methods, security was normally handled on the front-end. Companies often did not even implement database security because it was too hard to maintain multiple security lists. Services on the other hand are used by multiple applications, so they have their own security mechanisms. An application will therefore have multi-level authentication at both the client level and the service level.

In real life this is both a challenge and a benefit. The 'credit check' service must have a security feature embedded within it. An application or front-end that wants to perform a credit check must be able to give the right security passwords or connection properties in order to use the service.

The benefit of more security within the service layer of a SOA provides enterprises with greater reliability and better controls within their system.

Summary

For many enterprises, IT is seen as a cost-house, a continued expense that is necessary for business function, but constantly breaking down, needing updates, needing more, more, more and more. While this perspective is certainly not true for

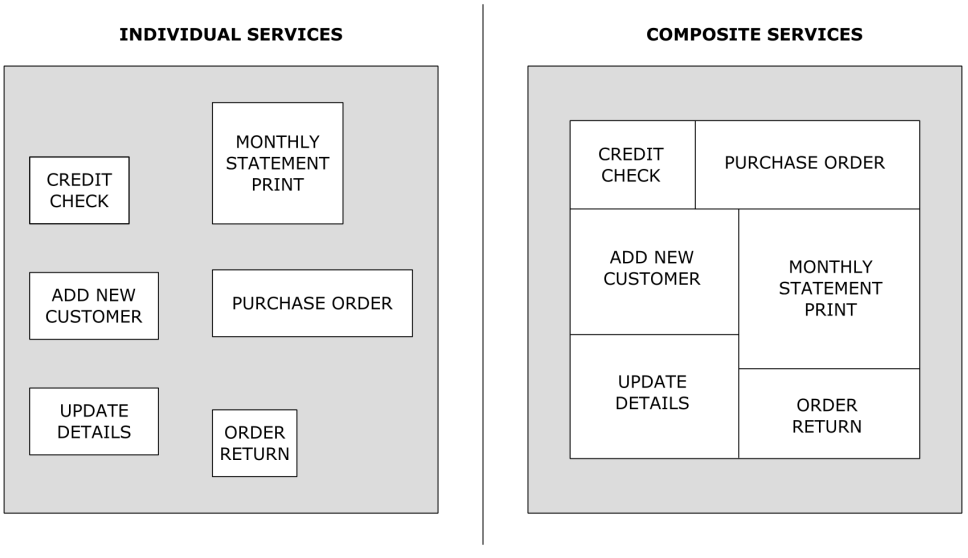
many IT divisions, it is an opinion that many people in business have. With a SOA, IT is able to take on a much more positive character. Business drives the changes in IT that are necessary for a working SOA, so IT becomes much more business focused and business aligned. IT can become the true business tool that it has had the potential to be for so long.

In the final section of this chapter we are going to focus specifically on the benefit of assembly without programing that SOA enables. This is one scenario that brings together the details many of the benefits of SOA when applied to the particular situation of evaluating an ERP or other enterprise level system that has been migrated to a SOA.

Assembly Without Programming - The Composite Application Platform

Imagine sitting down with your business and finance managers and asking them what functions they want from your Enterprise IT landscape. Then, you sit down at a computer and put together all the things that they have requested in one application - not by programming, but by choosing all the services that were required from a master list. Then, you pres one button and all those services are assembled into one interface. This is the reality of a composite application platform. Already many program vendors are producing SOA aligned products that allow you to do something like this. Most of these products will only compose an interface from the services produced within a particular company's enterprise offering, but with the continued development of repositories and service discovery applications there will soon be more composite application interfaces made from services produced by different vendors.

Figure 8.1. Combining individual services into a composite application



When services are published, marketed and served to any business that wants to incorporate their functions into its architecture, then the goals of SOA are being realized. It is a good step forward to have the business functions of your ERP platform transformed into business services that are then available on an as-needed basis. Businesses can then utilize what they specifically need and add more services to their enterprise interfaces from other vendors be necessary.

The composite application platform is the direct result of code reusability and interoperability. By programming services for reuse, the functionality and code are made available as a 'plug and play' type of assembly. As we see an increase in understanding of the concepts behind SOA, we will also see a rise in the services that will be available for a composite application, both within an organization and between organizations.

Grid computing, metadata repositories, enterprise software buses, interoperability standards and the continued transformation of legacy business logic into services all support the notion of assembly without programming. The pieces are already created

and operational, they just need to be fitted together in order to meet the specific needs of the specific enterprise.

There are a few things that we need to discuss on slightly more detail; the use of dynamic meta data repositories, the continued refining of services and the creation of visual tools that help define interoperability. These three topics will help you understand more about the concept of composite application platform. Let's start with the dynamic meta data repository.

Dynamic Meta Data Repository

Meta data is all the data that a system produces. A composite application platform will have a data repository that automatically discovers, catalogs, and organizes meta data regarding existing raw services in its dynamic repository (often called the Application Dictionary.) Whenever the meta data of a raw service is refined or enhanced within the system, those changes are dynamically recorded and added to the repository. The Application Dictionary is the critical piece of infrastructure for enabling the assembly of new composite applications, since it maintains and provides current views of all the building blocks that are available to the assembly environment.

In other words, this is a dynamic cataloging and storing system that is at the core of the composite application. It brings order to an otherwise chaotic data system by keeping track of all of the services available and how they all interact and communicate. In doing this is also refines the raw services into usable objects or combinations.

Refining Services

A composite application platform automatically refines all raw services in the Application Dictionary repository into a common business object format that is easy to understand - a format that is based on both Web Services standards and the principles of object-oriented design. From there, visual tools are provided for simplifying service interfaces or combining multiple services into composite services. These further refinements create more easily used building blocks that are also automatically cataloged and organized in the Application Dictionary.

The final step that makes a composite application feasible is the creation of tools that allow a non-programmer to create the links and communication channels between the various services. We call this 'defining interoperability.'

Defining Interoperability

The composite application platform provides visual tools that automatically capture any implicit relationships between the services from different application systems. Those tools also assist in defining new interoperability relationships between radically disparate applications and those new relationships become part of the meta data about the services in the Application Dictionary repository. The underlying technology that enables this crucial capability to define and leverage interoperability is sometime referred to as Application Algebra.

Assembly with no programming is a current reality and will evolve and improve as more and more businesses start using composite application platforms as part of their journey toward SOA. The visual tools and wizards for application assembly enable a variety of personnel from both IT and business functions to create new composite applications without requiring any programming. With refined components available and interoperability relationships predefined, the Application Dictionary repository hides the technical complexities of the underlying systems so that assemblers can focus on the business need of the composite application. The programming is done, now some assembly is required.

Conclusion

We have seen the benefits of migrating to a SOA , both in terms of business benefits and IT benefits. You had the opportunity to engage and use the knowledge you have gained in reading the introduction to SOA and should now be able to evaluate your enterprise and your needs in relation to what SOA can offer you. Before you do so, however, we still need to look at some of the challenges of SOA. Discussing the challenges will continue to solidify your understanding of SOA and will show you some of the areas that you might need to pay particular attention to, as you migrate to a SOA.

This page intentionally left blank

Chapter 9. Challenges of SOA

Introduction

Wherever you go, you will meet people of different viewpoints and different levels of optimism or pessimism. People will see a half glass of water as either half full or half empty. When people examine Service Oriented Architectures, they will also approach it from their particular view points. A realist will tell you that there is just a half glass of water, half full if you've just put water in up to that level and half empty if you've just drunk the other half. When looking at any challenges, it is better to think like a realist and acknowledge that the glass being half empty or half full depends on a person's perspective and the immediate history of the water in the glass.

In this chapter we deal with some of the challenges that enterprises face when migrating to SOA. Some of these may be less of a challenge to your particular business, but in thinking about how to migrate your business to SOA you will need to consider all of them.

Our approach to these challenges will be similar to the one we used when discussing the benefits of SOA. We will look at the challenges in two different areas: the technical challenges and the business challenges

Technical Challenges

Whenever a system changes, there will be all sorts of challenges. When an IT architecture changes, there are issues that have to be addressed before the architectural changes can take place. These are the technical challenges that an enterprise faces when migrating to a SOA:

- Data Rationalization;
- Business Service Enablement;
- Abstraction Compatibility;
- Service Accessibility;
- Constrained Innovation;
- Standards Interoperability.

We begin our discussion with the classic IT challenge of data rationalization. It seems that, no matter what project or system IT and business face together, data and how to call the data is a central theme.

Data Rationalization

Databases and data transfers are at the core of most (if not all) ERP systems. The underlying structure of all enterprise systems will have a database of some kind, either connected to it or as a central feature. For this reason, one of the biggest challenges that enterprises face when migrating to a SOA is how to manage their data. Data rationalization involves dealing with the condition of the data within these databases, with semantic incompatibilities (different definitions of the same data,) redundancies (the same data repeated in many different places,) and discrepancies in definition that may exist in the repositories that services will use.

Integrating data models in a complex enterprise can be difficult because the IT infrastructure often contains massive duplication and redundancy. Refining this situation without loss of data definition (semantic loss) is a tough nut to crack. Gartner, Inc., advises organizations wishing to fully exploit service-oriented business applications to focus on integrating the processes and underlying data models, rather than on integrating individual application components. Failure to integrate these aspects will place the organization at a competitive disadvantage. Gartner believes that such metadata management is "essential to reducing the escalating complexity of management and maintenance of integrated software platforms."

This is a challenge that must be addressed in consultation with your database and XML specialists. For the purposes of this book, you need to know that incorporating and rationalizing the data stored by your business and your business partners is a challenge that must be addressed. In the planning section of this book the section called “Data Transformation and Rationalization” [154] we will look at some questions that need to be answered to meet this challenge.

Any business that migrates to a SOA has to face the challenge of data rationalization. The next challenge that we discuss also must be faced by each enterprise as they plan their migration to SOA, the process of business service enablement.

Business Service Enablement

Many existing services are currently only thin veneers on a non-service based API making them unsuitable for the service requirements of SOA. They typically have been auto generated by commodity tools that are provided as extensions of tightly bound applications or have been coded by programmers for a once off programmatic use (i.e., quick fix.) In order to truly enable business services the whole enterprise needs to be aware and involved in transforming to a SOA. It is no longer feasible to coat a business service within an application with XML messaging and try to incorporate it as a true service.

The move to a SOA provides an opportunity to reuse what is working and get rid of what has not worked, while at the same time incorporating more agility and scalability within an enterprise system. Business service enablement requires that all levels of the business are involved and have an input into the process. For example, when transforming a bicycle manufacturing and sales business along the lines of a SOA you need to involve and enable marketing and sales, warehousing, distribution, manufacturing, assembly, accounting, staff management and executive functions. SOA allows an enterprise to start small and then grow to incorporate all these aspects of their business.

The road to full business service involvement requires the following steps:

- SOA assessment - a clear vision of how your business can benefit from a SOA.
- SOA vision definition - an outline of how a SOA can help solve your specific business problems.
- SOA transition plan - helping you map how your organization will make the transition to a SOA.
- Internal SOA center of excellence - a group of people with the right skills to form an internal SOA expert team.
- SOA organization and governance - defining rules surrounding your SOA.
- Comprehensive IT plan, including investment planning and IT activity prioritization.

When a business has been able to follow these steps and involve all levels of their

business in the exercise, then the process of true business service enablement has been accomplished.

The above steps will also aid your enterprises assessment of the levels of abstraction needed for your enterprise to make service compatible on all levels..

Abstraction Incompatibility

One of the goals of SOA is to reach a level of abstraction wherein all the services are inter-operable. Along the road to that interoperability is the problem of abstraction incompatibility. Simply put, there are many mismatches between Web Services that were designed at different levels of abstraction and this makes them difficult to inter-operate. SOA as a whole process must be utilized in order to deal with this problem. In the security section we posed the problem of one service only using a particular standard of encryption which the client application did not use. This was solved with the use of a conversion software layer. Abstraction incompatibilities need to be discovered and solved in the planning stages of implementing SOA so that they do not become a barrier to the implementation of the SOA.

Once the levels of abstraction have been resolved, the next challenge to address is that of service accessibility.

Service Accessibility

The SOA concept of maximizing service reuse necessitates a meta data repository to catalog and enhance the knowledge and understanding of available services. Such a requirement entails the provision of a repository that describes service capabilities in familiar business-object terminology; organizes (maybe through pictorial representation) related services in a meaningful context; manages secure access to services; and offers advance searching features so that users can quickly assess the capabilities of the available services. This is a huge undertaking and requires a new way of looking at data and networks. The composite application platform idea that we saw at the ned of the last chapter gives us a good idea of how to resolve this technical challenge.

The next challenge is also related to the levels of abstraction within a SOA, but within the framework of users becoming locked-in to a particular level of abstraction through the tools designed at that level.

Constrained Innovation

Within the context of a SOA the abstraction of core business services opens them to a greater audience or client base. Personnel inside and outside the core IT function who are closer to the business problems that need to be addressed require user-friendly tools that enable them to create innovative and integrated solutions. They will not have the experience nor the knowledge to approach the needs from a programming perspective but will be able to see from the list of services the solution to their needs.

When they are able to utilize tools to integrate the needed services, they will be less constrained and more innovative within the enterprise environment. This increases the service reusability and the organization's return on their SOA environment. The challenge comes when the user-friendly tools do not address business processes that the enterprise needs to incorporate within their system. They then need to go back to the programming layer and find or create a new business service that can be incorporated as a tool within the system.

The final technical challenge that we will discuss in this section deals with the level of abstraction that defines the standards used by services and questions the level of interoperability currently obtainable.

Standards Interoperability

There are currently many different standards and this makes it difficult to produce services that are fully inter-operable. As these standards evolve and merge through middleware and 'translation' software, this becomes less of a technical challenge.

The problem is that standards don't guarantee interoperability. By their very nature, standards are designed to support many different uses across many different types of systems and organizations. For example, security standards allow many different types of credentials including how a user is identified. If the sender and receiver don't both understand the same types of credentials, they can't talk together. The common language breaks down. Narrowing down the set of acceptable ways to use standards from all the available options is something your organization needs to address with policies and procedures. Of course, many of these policies and procedures will naturally need to change over time as regulatory or other compliance requirements change. So when thinking about the total cost of ownership, you need to factor in the inevitable cost of change.

We have now seen six specific challenges that enterprises face when migrating to a SOA. Let's now move on to the business challenges and discuss four more specific challenges.

Business Challenges

In the previous chapter we saw many benefits of SOA from a business perspective. There are also a few challenges that are worth discussing to provide you with a better overall picture of what you're dealing with when assessing SOAs. We will discuss the following four challenge topics in this section:

- Transfer Pricing Model
- Customer Service Levels
- Consequences of Change
- Ownership Issue

These topics will provide you with the information that will help evaluate a SOA and decide as to the effectiveness of the system for your enterprise. We start where most business issues start (where the money comes from,) the pricing model.

Transfer Pricing Model

The challenge within the SOA arena regarding pricing is the question of who pays for a service shared by many applications. With traditional line of business applications, figuring out who pays is easy - since one team owns everything. For a shared service, ideally each line of business should pay proportional to their use. Those who use it most should pay the most. Essentially this is a transfer-pricing model. Beyond the accounting considerations of how to implement transfer pricing policies and procedures effectively across lines of business, you must also consider how to track usage by each line of business accurately - if you can't measure usage, you can't charge for it!

In some ways charging for a SOA service should be like charging for any other shared service. You pay for your electricity service based on how much electricity you use. You pay for the telephone service based on the number or distance of the calls that you make.

Ensuring Customers Service Levels

To end users (the customers of the IT infrastructure) an order management application is still just an order management application whether it's built as a monolith or it leverages shared services of a SOA. In either case it must meet the same expectations of performance, availability and functionality. Conversely, if the same user utilizes two different applications, he/she may have different expectations of each - even if both applications leverage the same set of shared services.

While shared services serve many applications, they need to be approached in the context of the applications they serve to ensure the "customer experience." For example, changes may lead to unintended consequences. For instance, if an application using a service has an unexpected load increase, it consumes the service's capacity most likely reducing the service level for all the other applications. The use of shared services means that understanding the nature of the interdependencies plays a key role in understanding the effects of any changes.

The effects of any change will be felt throughout the whole enterprise, and so the consequences of change must be examined and understood.

Consequences of Change

In every environment change will bring stress. The faster the pace of change, the higher the possible stress factors become. SOA advocates a lot of changes. It challenges the way enterprise organizations think, plan and do their business. It reorients IT functions and changes ownerships. It is not a new way of doing things, but a new way of understanding things. As such, it can bring dramatic change to an organization. This change needs to be handled in a positive way, bringing together those with high and low stakes in the business and creating cohesion within the SOA vision.

The consequences of change can be drastic. In the planning section later in this book we will deal with some of the processes that can be planned when dealing with this change.

The final business challenge that we need to discuss is the issue of ownership and responsibility. In some ways this challenge is similar to the transfer pricing model as it involves a new way of looking at an IT product.

Ownership Issues

The ownership and responsibility for services may become an issue at various points along the migration route to a SOA or the development of a SOA service. As one team begins leveraging services built by other teams, they no longer have visibility into all the moving parts that make up their overall applications. This factor has a number of implications. For example, how does an application team ensure that its overall application is secure, when it's built using services from other teams? If its application is exposed to partners and customers via the Internet, how does the team ensure that the services it's using aren't vulnerable to malicious attacks aimed at stealing or corrupting information? Maybe the services the team is leveraging haven't been built with the same requirements in mind. Similarly, when something goes wrong, how can you figure out if the problem is due to the application itself or to services leveraged by the application? And who dictates security and business policy as it relates to shared services?

These are some of the topics and questions that you will face as you examine and evaluate SOA core services and platforms. If we are realistic about the concept of SOA we will find that there are many advantages of migrating an enterprise system.

In the next section we will approach the idea of challenges from a different perspective and bring checks and balances into the discussion as we look at the whole topic of the road toward a SOA .

Roadblocks Ahead

The road toward a fully developed SOA will be different for each business because SOA involves a mind set, not just a new tool for the job. The SOA vision will undoubtedly be adopted by IT functions in all the major industries, but the difference between the vision and practical reality of building composite applications must be addressed. We have seen that the service-enablement via Web Service wrappers around existing applications works. What IT architects must now identify and create is the kind of services that will fully utilize the capabilities of a service-oriented architecture. They need to advance the services to a higher level.

The balance between coarse and fine grained services must also be maintained. Good services must provide business value and be attractive to the user, particularly those who will be building a composite application from services. Meaningful business services are usually more coarse grained because they encapsulate more business value. The danger is that the service is likely to be overloaded with

functionality and data, which could be wasteful of resources, difficult to use in certain instances and most likely would contain too much irrelevant capability for the end user. Meaningfulness is often in the eye of the user, so different user needs will often motivate a more varied selection of services to accommodate the various demands.

The difficulty of quantifying the benefit of business flexibility and agility may be another roadblock that prevents a full realization of SOA. While these benefits are desired by all, the challenge for both users and vendors is to view results in the long term. SOA adoption will either require a strong executive level mandate or a grassroots justification based on enabling specific business capabilities or specific cost reductions.

Those companies that have started out developing and testing specific Web Services and utilizing XML for specific applications need to make the leap from this starting point to a full Service Oriented Architecture. The cost of a SOA includes enterprise-wide education and training and the dissemination of tools and documentation, etc.

Another potential roadblock is the current immaturity of some industry standards. While much progress has been made in establishing standards, the industry needs to progress beyond these first-generation standards, i.e., SOAP and WSDL. The challenge to the industry is to keep these basic standards and improve on them without adding too much overhead to the services.

Security is both a roadblock to fully implementing SOA and an IT benefit of migrating to a SOA. For this reason we will deal with security later in the book in a chapter of its own Chapter 10, *Security in SOA* [107].

The Build vs. Buy Argument

The fundamental investment questions of whether to build something in-house or buy an application to perform the function applies to all aspects of IT. It applies particularly when an organization is considering the implementation of a SOA. Both sides of the choice have their pros and cons. A decision that favors either route depends on a number of factors whose influence on the decision process varies between organizations. As a result, there is no golden rule that can help an enterprise choose one route over the other. Whatever the route, organizations generally inherit all that is good and bad about that option and are subsequently excluded from the good and bad of the alternative. Furthermore, once embarking on a route it is difficult and expensive to change course.

In this section we discuss the decision factors weighed by customers in the build versus buy debate and explore a third option, one that lies somewhere between build or buy and offers the best of both choices.

Background

Prior to the advent of packaged software, organizations wishing to implement software solutions had no choice but to build a custom application that met their needs. This situation changed in the nineteen eighties as software developers realized the commercial opportunity in developing applications that addressed the common processes and needs of many organizations. Since then, the software industry has seen the continuous return of the build versus buy debate and vendors in both camps periodically switch sides, depending on which option enjoyed the strongest support and had the greatest potential to drive increased revenues at the time.

During the nineteen nineties, accelerated technology advancement in market niches increased the number of business applications and core building blocks available in the market. While this resulted in greater choice for customers, it also led to greater confusion and complexity.

Still, these options did little to squelch the build versus buy debate. Now, instead of buying the technology stack and applications of a single vendor, customers were deciding on a level of vendor-centricity and augmenting this with other best-of-breed packaged solutions. If anything, this only increased the number of arguments instead of improving the customer's ability to take advantage of the benefits of both camps.

For sake of brevity, we will refrain from delving too deeply into the differences between these options. Fully explaining both sides of every option and comparing them against one another will take too long and in all likelihood will just fuel an ongoing argument. Instead, we will focus only on the basic options which all other options fall under to a lesser or greater degree: build or buy-packaged solutions versus custom applications. We will then illustrate the benefits and abilities of new emerging technologies that offer a more "middle of the road" approach that should benefit all.

Buy (Packaged Solution)

The advantage of the buy strategy is that, in theory, a package will enable the organization to go live with broad functionality in a short period. This is primarily

because the vendor has already developed the functionality and because it is easier to plan an implementation timetable.

Post-implementation factors also have to be taken into consideration. When purchasing a package solution and outsourcing the implementation, the customer is, in effect, shifting several aspects of risk. Generally, the subject of risk is associated only with the implementation. Yet, in the long run, taking a longer view, customers are also exempting themselves from the task of maintenance, effectively placing the responsibility and cost for keeping abreast of technological advancement and market trends on the vendor. Though leaving the vendor responsible for technological advancement can be a risk in itself, the buyer's risk is far reduced.

These factors, combined with the fact that the majority of resources engaged for implementation are outsourced, makes packaged solutions easier to budget for and appear cheaper on paper. The level of control, accountability and reduced risk makes for an attractive proposition when compared with custom development. Generally, it is difficult, if not near impossible, to enjoy all of these benefits when it comes to building custom applications, in-house or outsourced.

However, for all these positives, packaged solutions also have their drawbacks.

The most well known of these is that packaged solutions are never exactly tailored to meet all customer needs. Additionally, customers almost always end up purchasing and paying for more functionality than they actually need. As a result, they often complain that their total cost of ownership (TCO) is higher than expected for the smaller portion of functionality they actually use.

For the most part these complaints are valid and expected. Packaged solutions, by definition, strive to be a "one-size-fits-all" or an "off-the-peg" solution. They are, therefore, designed to fit the broadest set of requirements and are void of industry or sector specific functionality. Obtaining these enhancements often drives up cost significantly, resulting in a cost to functionality ratio that is difficult to justify. However, in some instances where vendors recognize that an enhancement will have broad appeal in their customer base, they will include the feature or functionality in the implementation at a much-reduced cost. This is, however, rare. Customers with shallow pockets soon learn that they are at the mercy of the vendor when it comes to specialist requests.

It is then that customers wish they had taken the build route. They start to source alternative packaged solutions or reconsider the build option with the intention of integrating the new solution to their existing system. Integration is, however, not always possible and often carries a heavy price tag especially when proprietary

technologies, such as electronic data interchange (EDI), are involved. Integration also has the nasty side effect of increasing the costs of future upgrades, since changes in any part of a system may cause the integration layer to no longer operate as initially designed. A simple addition or modification to a field may cause exceptions and even result in data corruption. The total cost of ownership is, therefore, forced higher, flexibility is reduced and often it may have been a cheaper long-term decision to have the main vendor develop the special requirements from the onset.

Understandably, this situation is undesirable from the customer's perspective and often leads to tensions in the relationship or worse, a general disillusionment with ERP as a whole.

The customer expects to purchase a packaged solution free from the constraints of the vendor and retain the ability to develop deep industry or sector specific functionality that will improve their overall level of execution to better customer relations, increase productivity and profitability-all the items for which they bought ERP in the first place.

Build (Custom Application)

In the face of advanced packaged solutions, support for building custom applications has waned but did not die a silent death and it is, therefore, still possible to find organizations that prefer to build their applications. When asked why they prefer this route, the answer most commonly given is, "When we surveyed the market, we did not find a packaged solution that met all of our business requirements or could integrate to the existing systems."

The key advantage of the build option is that organizations can create applications that match 100 percent of the nuances particular to their business. Most custom applications are also more closely aligned with user requirements and therefore increase usability, reduce training and generally promote the user experience and level of satisfaction with the system.

The additional advantage is that organizations only develop the functionality they require and are, therefore, not paying for features or enhancements for which they do not have a use. This comes with the added benefit of not having to pay ongoing annual license fees (ALFs.)

Object-oriented development techniques that enable a building block approach to creating applications have significantly reduced development time-scales. These

factors, combined with cost-effective, off-shore coding shops, have in many cases taken the shine off packaged solutions. The result being that the build option is once again enjoying a rise in popularity.

Now, for the bad news. As mentioned in the previous section, it is difficult to accurately place a timeline on custom development initiatives. Estimation of project duration and costs is, therefore, often speculative. Functionality must be built from scratch and project scope and process modeling need to be completed and understood before development begins. Getting either of these wrong may result in significant overruns to timelines and budgets.

Organizations on the build route also assume the risks of implementation and maintenance and have the responsibility for keeping abreast of technological advancement and market trends. Failing to do the latter may often result in the custom system being rendered inept from technical and business perspectives. When this happens, the result is often a scrapping of millions of dollars in labor in favor of a packaged solution that can be quickly implemented in order to avoid market fallout.

So the argument for and against either option is "six of one, half a dozen of the other." Both routes offer great benefits, but without any ability to reduce or eliminate their negative qualities.

What customers need is the ability to leverage the pros of both build and buy options while being able to avoid the negative qualities. They need something in between.

Somewhere In Between

Does such a place really exist?

Yes, new technologies have emerged that are widely considered to be sufficiently mature as technology platforms that can be used for development and solving business problems. These technologies include

- Extensible mark-up language (XML) - which we have already discussed in detail
- Managed code (realized in Sun J2EE and Microsoft .NET)
- Component architectures

Leading vendors of packaged solutions have already incorporated these key technologies into their product offerings, resulting in what can be described as a system and platform providing the best of both the build or buy strategies. Ironically, the drive behind providing customers with this choice has not been due to the problems described in previous sections. Instead, the driver has come from the customers' demands for collaborative commerce (c-commerce.)

C-commerce involves collaborative, electronically-enabled, business integration among an organization's internal personnel, business partners and customers throughout a trading community. The trading community could be an industry, industry segment, supply chain or supply event segment.

C-commerce is made possible by means of the extended enterprise, a SOA in which functionality, technology and architecture are used to enable the deployment and interoperation of enterprise applications both internal and external to the virtual boundaries of the enterprise network.

The demand for "something in between" combined with the new environmental requirements and demands of c-commerce have had the following effect on vendors. It has made them aware that it is impossible for any single vendor to develop packaged solutions that cater to the thousands of imaginable business applications that will deliver deep industry specific functionality in all vertical industries while continuing to assimilate applications and functionality in the core applications.

However, in order for vendors to qualify for entry to the extended enterprise arena, they are required to provide customers with a way to create what many call "next generation" applications. These factors demand change to the functionality, architecture and data of resource planning systems to accommodate the characteristics of "next generation" applications that will:

- Make extensive use of existing enterprise applications and business logic as a business process infrastructure.
- Make significant use of freely available, open and standards based integration and connectivity resources.
- Have a small technological footprint relative to the core technology stack or core applications.

Service Oriented Architectures are finally helping customers to realize the

"somewhere in between." They give customers the basic functionality, while enabling an extension or expansion in functionality from this position. Furthermore, they give the enterprise the agility and flexibility to develop (build) or choose other best-of-breed vendors (buy) and so craft a system that can be easily adapted to business requirements and next generation applications.

Conclusion

Next generation applications will undoubtedly become nothing less than the future of enterprise software. As we move into this era, we will be increasingly forced to rethink some of our conventional wisdoms and traditional approaches to application development. From the limited scope of this section of the book, perhaps the most important of these is the "80/20 principle."

This principle advocates that nearly 80 percent of any custom development effort is focused on overcoming problems related to core technologies, data access, security, query and user interface. The remaining 20 percent is spent on actually developing the business end of the solution. These new technologies and ERP II compliant systems reverse this principle so that 80 percent of the development time can be focused on developing the business solution.

The result should be custom business solutions built on the business logic inherent to packaged solutions that more accurately meet both the business and user requirements.

The final chapter in our discussion of the extended enterprise and SOA is the facet of security. We have chosen to speak about security in a SOA separately from the benefits and challenges so that you can focus your attention more clearly on what is currently available. Security within any system is something that is constantly being upgraded and hardened. Everything that you have learned about a SOA so far will be useful to you as you examine the aspect of security.

This page intentionally left blank

Chapter 10. Security in SOA

Security Concerns

As we continue to examine what a SOA is and help you understand it better (so that you can evaluate software and different systems and services) we need to focus on one more key area: the security options of a SOA.

The distributed and inter-connected nature of SOA makes addressing security concerns a critical success factor. The primary security concern in SOA is to establish an interoperable framework that enables security for services, applications and users in a trusted environment and complies with established corporate policies. We will find that the standards and techniques to provide security in a SOA are still evolving rapidly.

In the conventional enterprise application security realm, there is an underlying concept of a trusted computing base (TCB.) The TCB provides mechanisms for enforcing a security policy that protects the resources within the controlled enterprise environment. This is equivalent to providing a security fence or wall that protects valuable resources. Services, particularly Web Services, don't have a clear policy within a security perimeter. SOAs approach to distributed computing allows an application's functionality to be abstracted as business services that are location-independent and discoverable on a network. As such, the architecture allows service composition, which may engage many different service providers distributed across different platforms and enterprises. In such an open environment, distinguishing legitimate service requests from illegitimate ones becomes a challenge.

A secure SOA will need to address message security, trust policies, distribution policies, identity management and interoperability, all within an understanding of security standards appropriate to the technology being used.

Trust

As we have already mentioned, a SOA involves the loose coupling of services. This loose coupling makes the issue of trust quite explicit. The WS-Security specification defines trust as, "the characteristic that one entity is willing to rely upon another entity to execute a set of actions." The trust issue has two aspects: one is to specify the trust policies and the second is to broker trust between different security domains (also known as spheres of trust.)

In a distributed environment like SOA, the two sides (client and service) need to establish trust before they can interact with each other. One such way is WS-Trust, a Microsoft, IBM and VeriSign specification. WS-Trust defines extensions to WS Security to provide mechanisms to get security tokens and to establish trust relationships. For example, a client can send only X.509 security tokens and the Web Service can accept only SAML security tokens. WS-Trust provides a protocol to get the SAML security token by presenting the X.509 security token. By doing so, WS-Trust resolves the token format mismatch; trust between client and Web Service can be established. This provides a great benefit as different security infrastructures can interoperate with each other without significant changes. To some extent an enterprise software bus can also provide a translation of security protocols.

Trust within a networked environment is something that has to be negotiated. When a new service is started on the network, its 'trust' features are part of the description that is kept in the service repository. This information is also accessed as the service is dynamically accessed and discovered. Trust policies can change as a system grows and as more users become part of the SOA.

Message Security

Within the foundations of a secure SOA are the security aspects of each message. XML features and basic Internet (TCP/IP) security provide a basis for this security, but it must be taken further.

Message security involves providing message confidentiality, integrity, non-repudiation and the exchanging of security credentials between client and service. A message may have to hop through various intermediaries to reach the final destination. It becomes critical to maintain the confidentiality of the sensitive information so that no unauthorized entity can gain access to it. It may also be important to evaluate whether or not the message was not modified in transit and ensure that the integrity of the message is maintained. In most of the

Business-2-Business scenarios, it becomes important to establish the message authentication, which guarantees that the message was created by the claimed identity. Non-repudiation is also an important requirement in B2B applications. Non-repudiation supplies the guarantee that the sender can not later repudiate and claim that he never sent the message. Non-repudiation is required due to malicious senders who can later disavow having created and sent a particular message. Resolving non-repudiation issues requires message authentication and sender authentication simultaneously. This can be achieved by using XML signatures for message authentication and SSL-based sender authentication.

Due to the fact that most message security happens without our noticing, it is important to find out what type of message security a particular enterprise system employs. The information that we have given you here is very brief, but it is enough to make you aware of the importance of message security within a SOA, or indeed in any business/IT system.

Distributed Policies

In a typical SOA, where the client and the service may not be in the same security domain (or even on the same network system,) policies enforce security rules on the outgoing (client side) and incoming (service) messages. Smart enterprises will make sure that any communication with the outside is compliant with the policies they have established.

In a secure SOA policies are attached to all the related entities, client, service and discovery or service repository. Policies are used to describe a broad range of service requirements and capabilities. For example, an organization may use a policy to define security requirements such as encryption. Policies are validated at the time of interaction and within the context of the interaction.

For example, the results of a business analysis system may be available for specific shareholders and business partners to access, but within the distribution policies of the company this sensitive information requires a user authenticated login to the system and a particular password for access. Within the service description there would be the definition of what type of authentication was needed, what type of encryption to use and what the specifications on the type of password that can be set up and used to access the business analytics reports.

Identity Management

A client uses an identity (most of the time it is a user identity) to gain access to the service it needs. A typical SOA solution is distributed over multiple security domains and there could be several identities attached to a single user in different security domains. This poses some problems with traditional security approaches. The security infrastructures may vary among the various backend systems, so users may need to be authenticated for each system. The other problem is related to the SOA service composition layer, which might be calling many different atomic services falling under different security domains. The absence of an overall security context makes it difficult to associate multiple user identities.

SOA environments are typically highly decentralized in nature and so identity management becomes a significant challenge (particularly for Web Services.) Identities can be stored in many directories as well as many different types of directories, including proprietary username/password repositories, LDAP, Active Directory and X.509 certificate stores. An additional challenge is that SOAs may have requests that result in additional requests to many different applications at once. A SOA-ready service may be composed of many service operations from many different services that each has its own identity. As part of a single transaction, many different services may be touched whether in parallel or in serial process. Being able to authenticate and be authorized across all of these systems seamlessly improves the user experience as well as performance - driving the need for a federated identity solution for SOA environments

For the smart business, this means that identity management must be planned from the beginning. System architects and programmers need to have this feature well defined before implementing the rest of a SOA.

Interoperability

Although interoperability is not a security issue per se, it is a very important factor in making a typical SOA solution work. Most of the Web Services specifications provide a number of mechanisms to do the same thing, which leading to interoperability issues. For example, let's assume that a sender can encrypt data using Triple DES, AES 128, AES 192, or AES 256 algorithms and the receiver can only decrypt in AES 128. One does not need to know what all these encryption standards are in order to see that there will be a problem if the same encryption standards are not being used. In general, wide spread adoption of security standards increases interoperability, but even different implementations of the same

specification may not inter-operate in some cases. Core specifications (WS Security, SAML, etc.) are designed in a way to be extensible and provide a number of options for doing the same thing.

Many of the security algorithms change fairly regularly, as they are still being developed and evolving to suit the needs of different business environments. You need to be aware that security features should be as interoperable as possible, so that your services will be accessible over a wider range of systems and networks.

Current Web Based Security Standards

The Internet industry has a set of existing and widely accepted transport-layer security mechanisms, such as SSL (Secure Sockets Layer) and TLS (Transport Layer Security.) Despite their popularity, SSL and TLS (which is a minor update of SSL) have some limitations when it comes to Web based services.

SSL is designed to provide point-to-point security, which is not enough for Web Services because end-to-end security is required. In a typical Web Services environment where XML-based business documents are routed through multiple intermediaries, it becomes difficult for those intermediaries to participate in security operations in an integrated fashion. SSL-based communication provides security (confidentiality, integrity) at the transport level rather than at the message level. As a result, messages are protected only while in transit on the wire. For example, sensitive data received on SSL channel, once displayed, is not generally protected unless one applies some encryption technology. Finally, SSL does not provide element-wise signing and encryption. For example, if there is a large purchase order XML document and a need to sign or encrypt only a credit card element, signing or encrypting only that element with SSL is not possible.

All of these limitations make SSL and TLS unsuitable for most of the security needs of Web services. In the last couple of years, this gap has been identified by the industry and considerable effort has been invested to provide a new security architecture to address the needs of SOA. It is important to realize that the new service oriented security architecture must adhere to the essential characteristics of SOA.

Organizations have existing security infrastructures in place, which protect the resources on diverse platforms. The fundamental goal of SOA is to enable the

existing infrastructures to interoperate through services. The solution is to enable a security spanning layer over existing security infrastructures with a SOA.

Using SOA enables organizations to build a set of reusable security services that can be used by business applications. The resulting reusability ensures consistent security policies across platforms with reduced development costs. For example, an authentication function can be offered as service. Due to the distributed nature of SOA, it is important that the standards-based architecture is adopted, which delivers interoperability and lets the infrastructure operate across organizational boundaries. In the last couple of years, much work has been done to create security standards, which is very promising.

WS-I Basic Security Profile is a WS-I initiative. WS-I isn't a standards development organization, but it works closely with a number of standards bodies - W3C, OASIS - to promote and utilize the right set of technologies in business scenarios. The Basic Security Profile will be a guide for the use of Web Services security standards and technologies in the development of interoperable Web services. The Basic Security Profile is an interoperability profile that addresses transport security, SOAP messaging security and other security considerations. The profile provides specific security requirements, which can be tested on the sender as well as receiver side.

SOA Security Summary

Despite the number of complex issues surrounding security in a SOA, the emerging security standards have a lot to offer. A significant amount of work has been done in designing security standards in a modular, complementary and evolutionary manner. As the domain of security standards is becoming mature, major Web Services platform vendors (IBM, Microsoft, Sun, BEA Systems, SAP) have started adopting these technologies. The future of security standards revolves around three major topics: first the standardization of various WS-* specifications (WS-Trust, WS-Policy, WS-Federation) at least at the functional level; second, the convergence of the identity federation specifications (SAML/Liberty Alliance and WS-Federation,) and third, the increased focus on interoperability.

Entrepreneurs and business owners do not need to know all the fine details regarding the security of a SOA, as long as they have experts to help them. They do need to know the basic principles and the current practices. In examining security in SOA we have examined some basic principles of security and seen some of the current practices and technologies.

Chapter 11. Summary of SOA

Through the last chapters we have examined SOA in some depth and have discussed the benefits, the challenges and the potential of security in a SOA. Before we go on to look at some models of the SOA platform in action, let us take a step back and bring the discussion of SOA back into a single focus through summarizing the main points.

We have seen that service-oriented architecture is a style of design that guides all the aspects of creating using business services. SOA guides the whole life-cycle of a business service, from its conception to retirement. It is also a way to define and create an IT infrastructure that allows different applications to exchange data and participate in business processes, regardless of the operating system and programming languages underlying those applications.

The core building block of SOA is the 'service.' The other core building blocks within SOA are the dynamic discovery of these services (the searchable directory of services available,) and the front-end or client side of the services (where you the user see the service in action on the screen in front of you.) The concepts enabling these building blocks are the technological advances in message based communication (the standardization of complex types of data messages sent between systems,) loose coupling (making a service as independent as possible,) distributed networks (local intranets and systems like the internet,) interoperability (standardization of how services can operate together - the glue that holds the services together,) Web Services (a service enabled within the internet environment,) and the enterprise software bus (an integration tool that converts messages and documents between services.)

All of these work together to produce an architecture that operates on services. It is a business service driven enterprise system that enables the business greater flexibility and agility in regard to change. Therefore the business can respond faster to market changes. This in turn brings greater efficiency and quicker response to market demand - with the end result of greater competitive advantage and increased margin of profit.

We then discussed some of the underlying standards, technologies and supporting concepts. We saw that the use of XML (Extensible Markup Language) was enabling data to be communicated in more and more intelligent ways, particularly with the addition of XML Schemas, the Web Services Descriptive Language (WSDL) and the Simple Object Access Protocol (SOAP.)

Knowing all this, we then looked a some benefits, both from a business perspective and an IT perspective. Business agility is one of the key business benefits of SOA today. The need for agility is not new. The way business was conducted even a decade ago is no longer acceptable if a business wishes to remain competitive. The changes include how the business interacts with customers, how it manufactures goods, how it is organized and managed. The changes in business are fundamental and pervasive. We also looked at some of the challenges that need to be faced and at the concept of security within a SOA. These are the areas within SOA that are still evolving and part of the process of incorporating the concepts of SOA in the enterprise environment.

SOA is a key component of the extended enterprise. It is an architecture that brings together and extends the business processes and technological advances that are foundational to business today. Some of the challenges that we have examined need to be addressed while implementing a SOA, but some may also be viewed as opportunities to create something better.

At the core of the concept of SOA is the understanding that SOA provides a conceptual structure for the enterprise to interact with change. By creating services that encapsulate business processes and enabling them to be used on different platforms, we create a reconfigurable structure. This structure involves the people and the technology because it is conceptual; its functions are created for business users. All the things that we have discussed allow us to conclude that SOA brings business and IT together in such a way that they are no longer just individual parts of the enterprise. They are integrated. Technology and business processes work together to improve the enterprise's ability to respond to internal and external change and therefore releases incredible potential business energy that can be harnessed to produce the competitive edge about which we have talked.

Part III. SOA Models

In this part of the book we will present two model applications of the SOA concept: Demand Driven Supply Networks and Collaborative Planning Forecasting and Replenishment. These are discussed with definitions, examples and detailed business implications.

By the end of this part of the book you will understand how a SOA enables demand driven supply networks with collaborative supply chains, real time demand management, supply excellence and the role of continuous innovation within a DDSN. You will better understand the collaborative planning forcasting and replenishment model that SOA supports, including the basic CPFR model, the role of strategy and planning, demand and supply management, real time execution and analysis of the whole process.

Chapter 12. Demand Driven Supply Networks

The concept of SOA can be applied to different sectors of the market and different models discussed for those sectors. One such sector is the relationship between demand and supply.

Introduction to DDSN

As a manufacturer, wouldn't you welcome a system where there are no warehouses, inventories or paper invoices, just plug-ins that monitor your supplier network automatically, in real-time, everywhere, simultaneously? The synchronized execution of manufacturing and supply across a dynamically re-configurable supply chain network (to profitably meet demand) is an ideal scenario. Optimized Demand Driven Supply Chain Network (DDSN) is enabling companies to achieve this. The mantra is moving from just measuring performance, on internal cost and efficiency, to external processes targeting customer-satisfaction at the shelf. Having a well-harmonized network maintaining high operating margin - not just profits or growth rate - is the goal of any organization.

The global market place is increasing the need to manage changes across geographies and maintaining steady demand not only during a product's life-cycle but also during periods of fluctuation. With advanced technology and know-how, product life-cycles have shortened while product mix is getting more complex. In this changed scenario, DDSN seems to be the ideal solution. Strategizing for managing all resources aimed at meeting customer demand and setting up metrics to monitor and manage the supply chain are all important elements in the entire process.

DDSN is not a brand new product, but rather a new method of using older products that are supplemented with new technologies, new data and new analytics. Supply Chain Management products used supply and cost focused data. DDSN uses this data but places demand management at the core of the value chain optimization. DDSN also utilizes the new technologies discussed within the SOA section to cross organizational boundaries (i.e., Web Services) incorporating it's function within a Service Oriented Architecture.

A general lack of understanding of the requirements is a big barrier to the creation of

systems that can sense and respond to demand visibility. Unfortunately, without knowing it, most of the current thinking with supply chain experts operates in a supply -- not a demand -- paradigm. Most vendors talk demand-driven systems while answering with a system for constraint-based supply theory. For most vendors, this change will require listening, understanding and truly building new applications for the demand-driven supply chain.

Collaborative Supply-Chain

A supply chain consists of a set of firms or businesses that are acting to design, manufacture, engineer and distribute products and services within the market.

Within the extended enterprise, due to the increase in network connectivity, trading partners are increasingly willing and able to participate in decision making processes up and down the supply chain. This is the essence of a collaborative supply-chain. In this context, collaboration synchronizes decision making among the participants in the supply chain through an electronic medium that analyzes the market and sets communication standards between the parties. Successful collaboration, in the business sense, means that two or more groups or companies are working jointly to:

- Derive shared information;
- Plan based on that shared information;
- Execute with greater success than when acting independently;
- Measure performance; and
- Reward success.

Management of the supply chain moves from a departmental view to a team-focused approach, incorporating people from all parts of the supply chain. This requires a parallel flow of information rather than a linear flow (i.e., all those involved in the supply chain have access to the relevant data as soon as it is entered into the system. They do not have to wait for a manager to release the data to them.) Thus, the focus of information technology within the collaborative environment moves from application centric to data centric. Synchronizing events within the supply chain requires that certain data exists and is available to all the relevant parties. Manufacturers must learn to collaborate internally and externally. This builds the

culture of information sharing that empowers everyone across the board. Collaboration makes it easier for companies to adjust to changing scenarios.

Successful collaborative supply chain management streamlines business processes by helping enforce schedules, optimize stock levels, avoid errors, improve performance and automate processes.

Synchronized Swimming

Have you ever watched an Olympic synchronized swimming team go through their presentation? It is a truly awesome sight to see. The grace and beauty of the synchronized movements add to the grace and beauty of each individual movement. Sometimes they move in perfect unity, at other times in perfected syncopated movement, like a flower opening all its petals. The swimming team achieves this through much practice, but it is the original choreography that unites the team and awes the judges. DDSN can be viewed in a very similar manner. The planning stages are crucial and without a good plan the expected end result will not be achieved. A good plan followed by great team work and cooperation will bring about a system that works and one that brings the competitive edge to those enterprises engaged in the supply chain. There are three areas that work together to bring about the results of this synchronized team work: Demand Management; Supply Excellence; and Continuous Innovation.

Demand Management

The classic answer in dealing with demand variability has been inventory. This buffer allowed enterprises to absorb sudden surges in demand. DDSN shortens the cycle between demand signals (POS data, movements from the back of the store to the front, or distribution center to store movements,) production and replenishment. In other than seasonal items or promotions, buffer stocks are seen as proof that the demand signal to replenishment cycle can still be shortened.

Tools dealing in forecasting, price and revenue optimization and promotions management deepen an enterprise's ability to manage the supply/demand balance by tapping into the demand variability as a resource. Collaboration facilitates real-time focus on inventory levels, capacity outlooks and new technology drivers, which, in turn, helps in better management of demand. Setting up a collaborative network will help build effective supply chain components.

Supply Excellence

Businesses with demand-driven supply networks get paid 70 days sooner and bring new products to market 70% faster than their less enlightened rivals, says AMR. The consultancy (which advises corporate customers on creating such networks) also found that companies with a demand-driven approach to production have a 92% perfect-order rate, as opposed to 81% for their competitors. Overall, companies with a demand-centric-as opposed to factory-centric-bent are adding 5% to their top lines.

Continuous Innovation

The demand-driven supply chain network is best thought of as a dynamic networking of organizations and supply chains that work together in collaboration to provide a seamless pipeline of products and demand information from source suppliers through to end consumers. The challenge here is to build alliances that help reach the customer faster. So, collaboration is the key. Companies need to identify areas for improvement that have significant cost reduction potential and high probability of successful implementation.

DDSN Capability Model

Creating a demand driven supply network within your enterprise must start with analyzing what you currently have and what you want from a DDSN. Once you have this analysis, you can begin planning to implement DDSN features within your current system and incorporate other parts of the supply chain along the way. A DDSN capability model must be examined and used to bring about the systematic and enterprise changes that need to take place in order to achieve a truly effective DDSN.

AMR Research's Demand Driven Supply Network capability model is an online tool (http://www.amrresearch.com/) that allows companies to benchmark their journey toward becoming demand driven and powering effective supply networks. Companies are at different stages of capability, according to our DDSN model and there is a clear correlation between DDSN progress and company performance.

A DDSN system is more than just an improvement in supply chain performance. It is a change management endeavor that starts with supply chain excellence and results in supply networks that respond quickly to demand. Creating an organization that is demand driven and successfully manages supply networks is not a quick process. It is a three to five year journey. DDSN capability is described in four stages: reacting,

anticipating, collaborating and orchestrating. Placement in one of the four capability stages is based upon where a company stands in six assessment categories: business process, organization, technology, measurement, continuous improvement and culture. The move toward supply chain excellence represents stages 1 and 2; becoming truly demand driven is achieved in stages 3 and 4.

Based upon interviews with DDSN leaders, AMR Research has built the capability model upon the following tenets:

1. Leadership is essential.

   DDSN is most successful when there is a DDSN leadership team and the focus is on DDSN as a change management initiative. This team clearly understands the differences of supply chain management and DDSN.

2. This is not a step change or the program of the month.

   The most successful DDSN organizations craft their programs as phased approaches, not as step changes. In these organizations, there is board-level support for the entire journey, not just the first year or two.

3. Don't undervalue the need for a visionary.

   A visionary is vital to success. This is someone who can carry out a phased plan with key milestones tied to business drivers.

4. Technology is not a driver.

   Because many improvements can be made within the boundaries of existing technology, the least important element of the DDSN capability model is technology. Technology is needed, but companies cannot automate their way to DDSN success.

5. Culture is the hardest element; business process fusion is the starting point.

   The starting point is the business process -- the fusion of demand management, supply network design and product innovation. Organizational structure, measurement practices and continuous improvement programs help and are refined throughout the journey.

In conclusion, a clear correlation exists between becoming demand driven, establishing effective supply networks and company performance. Companies must move from whiteboard to reality in the next six months or risk being left behind by DDSN leaders.

The DDSN model demostraights the felxibility and agilty of a SOA system, as well as the real-time access and data connectivity benefits necessarry for a demand driven supply chain. We will also see these benefits fitting the needs of the collaborative planning, forceasting and replenishemnt model.

Chapter 13. Collaborative Planning, Forecasting and Replenishment

Introduction to CPFR

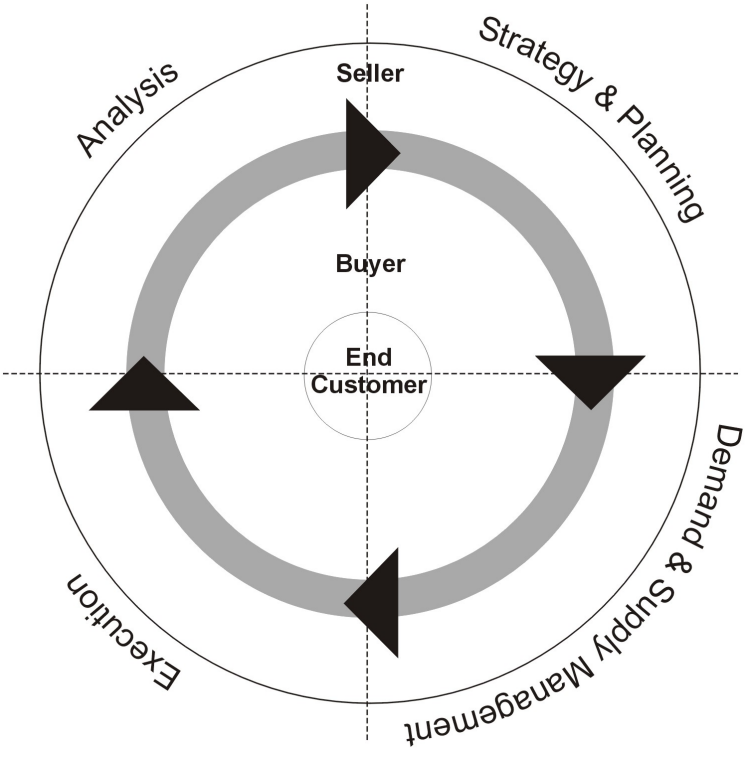
CPFR has been defined as "a business process for value chain partners to coordinate plans in order to reduce the variance between supply and demand." In a CPFR partnership, the buyer and the seller have a collaborative agreement with specific business goals like in-stock %, order fill rate, DC service level and forecast accuracy.

Collaborative Planning, Forecasting and Replenishment (CPFR) is the latest in a long line of supply chain initiatives that have promised greater profits through improved efficiencies and increased collaboration between trading partners. Some of CPFR's predecessors include electronic data interchange (EDI,) vendor-managed inventory (VMI) and efficient consumer response (ECR.) However, CPFR differs from its predecessors in that it is designed to link the supply and demand processes, allowing for a more consumer driven supply chain. CPFR aims to seamlessly link the CPG industry from manufacturer to consumer, allowing trading partners to see the entire supply chain from one end to the other. CPFR calls for complete collaboration and information sharing between trading partners, including the merchandising process, item/category selection and seasonal and promotional planning. Combined with real-time updates based on hourly activity, trading partners will be able to engage in total supply chain visibility and forecasting.

The CPFR Model

The CPFR model provides a framework comprised of four activities. Businesses involved, or wanting to be, in a collaborative supply chain will cooperate on these activities with the aid of technology. Figure 13.1, “Framework of the CPFR Model” [124] shows the model as a logical cycle. In reality businesses will be involved in all activities, to a greater or leaser degree, at any moment in time because each activity can impact on the other.

Figure 13.1. Framework of the CPFR Model



In the document "Collaborative Planning, Forecasting and Replenishment (CPFR) - An Overview" [ref] these activities are further broken down into the specific tasks in which buyers and sellers will engage. We will not go into detail on each of these

tasks. It is enough to understand that each activity mandates a number of tasks. There are three task categories, including:

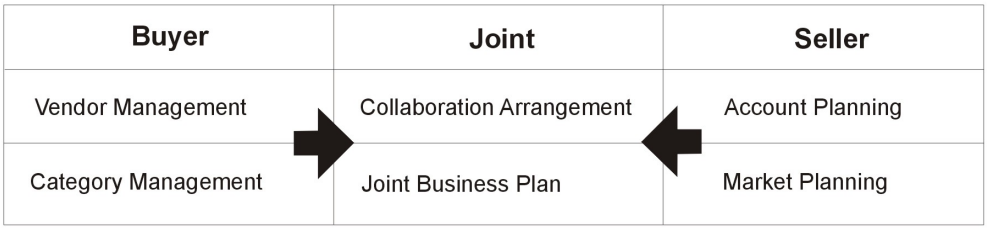
- Seller tasks
- Joint tasks
- Buyer tasks

These tasks are listed, but not detailed, in the following sections. Seller and buyer tasks are independently executed and provide input to the joint tasks.

Strategy and Planning

This activity is concerned with establishing the ground rules for the collaborative relationship. Buyers and sellers are required to agree on topics, such as product mix and placement and develop event plans for the period. Figure 13.2, “Strategy and Planning Tasks” [125] lists the tasks in which buyers and sellers must engage to provide input to the joint tasks that enable collaboration.

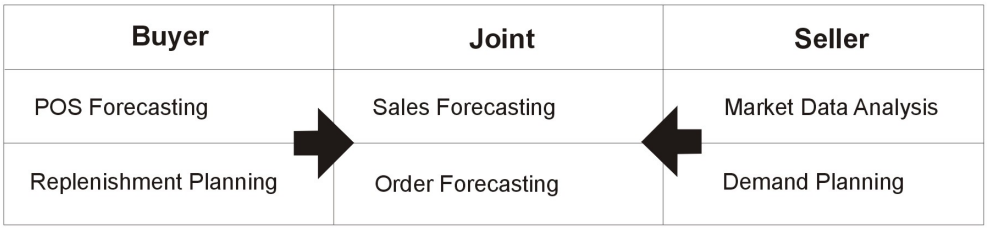
Figure 13.2. Strategy and Planning Tasks



Demand and Supply Management

This activity is concerned with projecting the consumers' demands, orders and shipping requirements for the period. Figure 13.3, “Demand and Supply Management” [126] lists the tasks in which buyers and sellers must engage to provide input to the joint tasks that enable collaboration.

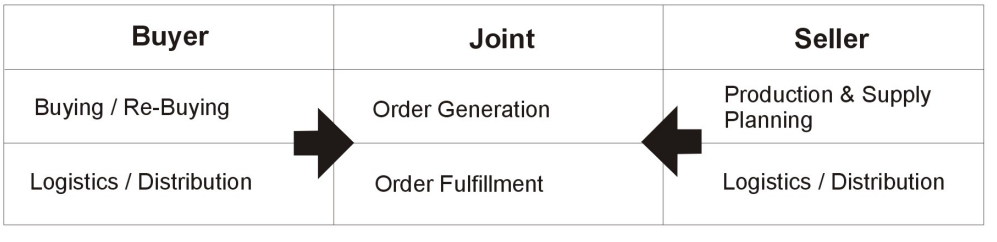
Figure 13.3. Demand and Supply Management



Execution

This activity is concerned with order placement, preparation and delivery of shipments, receiving and stocking of products, recording of sales and making of payments. Figure 13.4, “Execution” [126] lists the tasks in which buyers and sellers must engage to provide input to the joint tasks that enable collaboration.

Figure 13.4. Execution

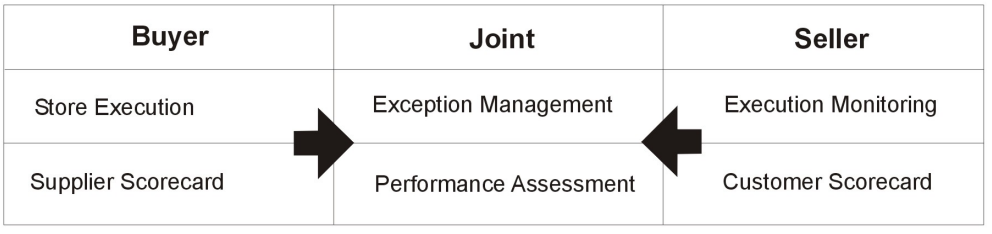


Analysis

This activity is concerned with the monitoring of planning and execution activities so that results, aggregated across the supply chain, can be used to produce *Key Performance Indicators* (KPI.) The shared results can therefore provide insights that will enable everyone to identify problems and adjust their plans. Figure 13.5, “Analysis” [127] lists the tasks in which buyers and sellers must engage in order to

provide input to the joint tasks that enable collaboration.

Figure 13.5. Analysis



This page intentionally left blank

Part IV. Implementation

In this part of the book we deal with the process of evaluating and creating Service Oriented Architectures and provides steps and models that can be followed along the way.

By the end of this part of the book you will better understand the process of integration when transforming your enterprise to a SOA. You will understand more about the technologies and tools available, as well as the methods and models that are being used to implement SOA within the enterprise environment. You will also better understand the impact of change and the process of planning for that impact, both in the people resources and the technology resources of the enterprise. You will understand more about data transformation and the process of architecting an effective solution for your business.

You will also better understand the process of adapting to change. You will understand how a SOA enables a business to increase its responsiveness to general change and change in the market within which it operates. You will understand the effects of change within a SOA and the need to be careful of some of the impacts of change. Finally, you will better understand how to setup a SOA to maximize the effectiveness of positive change and decrease the impact of negative change.

Chapter 14. Introduction

You might have thought that architecting a SOA deals primarily with fast and efficient services, minimalist service interfaces, optimized database access and, adequate network bandwidth and processing power. These are certainly among the architectural considerations that bear upon the ability of a SOA to deliver value. But these are technical issues that are well recognized and solvable. In fact, while ideally these technical issues should be planned for prior to implementing the SOA, they can be successfully addressed during the building of the architecture or even after it is in active use. In other words, getting these right only brings you into the game; it doesn't make you a winner. The issue is not in building the services themselves, but rather the architecting of the SOA on the whole. And, unlike most architectures, this is more of a business challenge than a technical one. Most SOAs fail to achieve their desired goal because of their inability to tie in to the business objectives.

Other SOA architectural issues become a part of the infrastructure and can't be fixed after the fact. They have more to do with the definition of the services than their implementation. In other words, you may build perfect services, except they may not be the services that bring benefits to your organization. Think of it this way: Services define the essential business practices of an organization and applications call those services to fulfill those practices. For example, one such practice might be ordering a product. You might say that all businesses do this in the same way, but there's always something unique about how it is implemented. In this case, a visitor to the web site selects an item and at that point a Web Service checks the inventory database to see whether or not that item is in stock. If it is in stock, that Web Service calls another service that prepares pricing and shipping information. That result is sent back to the first Web Service and the availability, pricing and shipping information is delivered to the user.

The set of Web Services mimics the manual process that has been in place in the business for any number of years. But then there's a problem. Let's say that some time in the future the business changes this particular practice. There are now alternative methods of shipping and the user must select a shipping method in order to complete the transaction. Yet, the existing Web Services don't easily support that change in business practice. At a minimum, it's clear that the second service would have to be called independently by the application, rather than by the first service. In the worst case, both services may have to be partially or completely rewritten in order to support a single change in the business practice.

This particular example is somewhat artificial and simplistic. Any real-life analogy

will be more complex and difficult to resolve and it's likely that an obvious and effective solution won't exist. In other words, this is an example of the easiest problem you will have with a poorly architectured SOA. It represents a cautionary tale of anyone embarking on a SOA strategy. Building services, even computationally efficient services, is not a guarantee of success, at least not success in terms of improving the current and future business prospects. And that is really the key. There is no reason to pursue a SOA strategy just to improve operations in the present. The importance of a SOA lies in its ability to also prepare you for the future.

Chapter 15. Implementation Methodologies

Technologies and Tools

SOA success is highly correlated to the choice of technology and tools utilized in both design-time and run-time environments, as well as the way in which those technologies and tools are designed to interplay and deliver a true service-oriented solution.

In the design-time environment, SOA readiness is impacted by how well the integrated development environment (IDE) extends to support rapidly emerging SOA implementation standards, especially Web services protocols. Design, development, testing and deployment tools must support unique protocols and attributes of SOA and critical infrastructure is necessary to support publication and discovery of new and existing services. In the run-time environment, SOA readiness demands an infrastructure that supports secure, interoperable and reliable messaging between services and is capable of orchestrating service interactions to support business process demands.

Choosing the appropriate tools and technologies is not sufficient to achieving SOA success alone. The use of appropriate design and message exchange patterns is critically important. These enabling patterns allow organizations to deliver and utilize services in a repeatable manner to ensure consistency and adherence to best practices necessary to scaling SOA across the organization.

Examples of SOA success factors pertinent to technology and tools include:

- **Identity**

  Appropriate use of security protocols, centralized identity credential stores, identity provisioning and key management systems, as well as consideration of federated identity where applicable

- **Registration/Discovery**

  Implementation of registry and repository technology to enable publication of

service records, metadata and support and escalation procedures

- **Service API**

  Appropriate use of WSDL and XML schema definition for description of technical binding specifications and data definition

- **Tiering/Layering**

  Separation of end-to-end processing into a number of tiers and implementation of each tier in a layered manner to increase architectural flexibility and reduce management and support burden

- **Loose Coupling**

  Appropriate use of interface styles and messaging patterns that reduce the potential for tight coupling based on underlying platform and programming model exposure

- **Pattern Usage**

  Identification and usage of architectural patterns specific to SOA, including message exchange patterns (MEPs,) appropriate Java 2 Platform, Enterprise Edition (J2EE) core patterns (Web Broker, others) and MicroArchitectures (well-aligned aggregated pattern sets)

- **Creation and Deployment**

  Use of the IDE and toolset that enables easy exposure of business logic as Web Services, including SOAP facades and WSDL generation, integrates well into the service deployment platform of choice and is supported by an ecosystem of compatible plug-ins to easily add new functionality and standards support

- **Standardized Data Model**

  Standardization across the organization on a common XML schema-based data model - in alignment with industry-specific and cross-industry data schema standards efforts - to minimize overhead and problems associated with data translations and transformations between services

- **Separation of Business and IT Services**

Differentiation of mission-aligned business functions from common enabling IT functions, such as authentication, authorization, logging, and notification, to enable greater specialization of services and reduce time to market (TTM) of new business services

- **Interoperability and Open Standards Basis**

  Use of true open standards for interfaces between services, as well as coordinated standards usage guidance, especially WS-I profiles, to increase probability of interoperability between services; choice of a programming platform with proven commitment to support existing and emerging WS-I profiles in a comprehensive, timely manner; choice of a deployment platform vendor that also displays a commitment to support key SOA interoperability standards

The tools and technologies used are only part of the equation of of well constructed SOA. The underlying methods of using and integrating the tools and technologies are equally important. For this reason it is important that we also consider methodologies and processes that assist a productive implementation of SOA within an enterprise environment.

Methods

As with the adoption of any system or concept, there are best practices and methods of migrating to and incorporating the system. We know that SOA is not just about the technology and programming code of the services, but also about the people within the enterprise and those connected to the enterprise through the functioning of the SOA. With this in mind, it is important to consider what methods are being discussed and what processes are available to help implement the changes.

The adoption of a formal methodology is critical to supporting quick, efficient, service-oriented analysis and to designing life-cycle activities that complement SOA. Also, formal governance and operations procedures make an organization more likely to adapt to SOA. New techniques for model-driven architecture that heavily involve visual business process and service interaction modeling approaches are beginning to emerge and are also important to evaluate.

Here are two examples of methodology and process:

- **Governance Model**

  Policies and procedures for the identification of necessary services, coordination of complementary or conflicting service development efforts and full life-cycle management of services from proposal to support

- **Model-Driven Architecture**

  Familiarity and/or usage of emerging approaches and technologies that enable development of service-oriented models to facilitate interaction and communication of requirements and logic between business analysts, architects, developers and testers

It is not within the scope of this book to communicate in the necessary depth what methods should be considered or used, but we can make some recommendations that will allow you to think through the process. This will also help you evaluate the information in other sources on SOA methodology. Our recommended approach for any organization looking to moving to SOA is to form a strategy that involves the following four major activities:

- **Education** - Gain an understanding of key SOA architectural principles, concepts, best practices and technologies

- **Assessment** - Determine the current state of your organization's readiness for moving to SOA by identifying existing best practices and gaps, as well as major opportunities for realization of benefits from SOA

- **Planning** - Develop a phased SOA migration plan that makes sense for the organization, mitigating business and architectural risks while measuring and delivering significant return on investment (ROI) through increased flexibility and responsiveness to changing market demands, as well as decreased design, development, integration and support costs

- **Execution** - Deliver prototypes, pilots, infrastructure and services consistent with the phased SOA migration plan, seeding and embedding SOA perspective and best practices throughout the business and technology groups within the organization as well as among key customers, partners and suppliers.

While each of these activities is equally important, one of the best ways

organizations can get started today in participating in the SOA value proposition is to perform an assessment of how well-aligned their current environment is with SOA principles and best practices. Evaluating key areas of impact and the success criteria specific to each will enable an organization to identify both existing best practices as well as areas of concern that must be addressed to begin the migration to SOA. Conducting such an assessment of an organization's SOA readiness is best done by engaging experts who have developed formal evaluation criteria, metrics and best practices to compare against and provide a basis for developing a set of tactical and strategic recommendations.

In order to help you deal effectively with system experts we will now quickly discuss some ideas from a business model perspective. The perspective of a business model will also help you integrate your knowledge of SOA with the process needed to implement a SOA within your enterprise.

This page intentionally left blank

Chapter 16. Business Models

Why is the SOA approach to application design so difficult to get right? It requires building a set of components that not only models business practices today, but also is versatile enough to provide support for future unknown business practices. In fact, a well-designed SOA might just make the creation of those future processes obvious when the time comes to make that change.

Being able to assess what business practices will look like in several years is key to creating a successful SOA. Clearly a part of it is a guess, so the trick is to base your guess on knowledge of the business. Among the things you have to consider are scenarios in which the product or service mix that constitutes the core of the business may change, the number and types of competitors may be significantly different or the entire business model may change.

This focus on the future means that the architecting of the SOA has to be done in conjunction with enterprise strategic planning. As the strategic planning process outlines scenarios for the future, the business and technology architects project processes will support those scenarios. Then you have to map application components that support existing business processes into those future processes. That's where the SOA components should fall out of the mix. Then, you're ready to go forward with the architecture. Following this approach gives you the best shot at building services that deliver flexibility and the ability to support new applications while providing the critical pieces that deliver value to future applications.

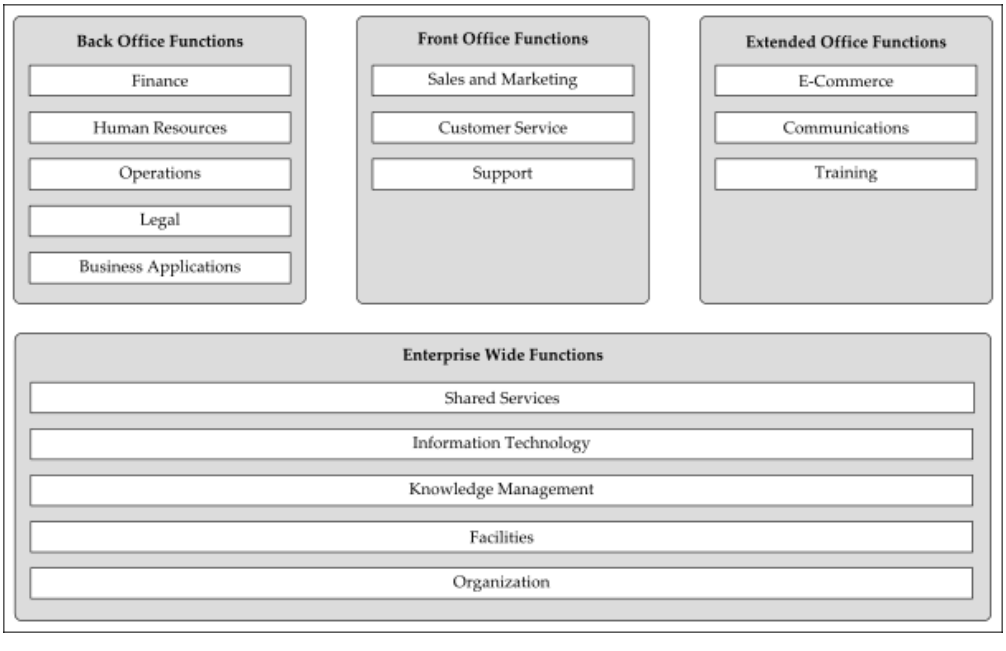
Alternatively, simply taking applications supporting existing processes and encapsulating them into Web Services may look good on paper and may win kudos for creating successful Web Services, but you haven't built an architecture that will serve the organization well into the future.

Either approach delivers an enterprise SOA. The latter leads to a proliferation of services that deliver marginal or no value going forward and requires the development and maintenance of an increasing number of services to support new applications. Tying a SOA with the enterprise strategy, however, means writing and maintaining fewer services over time. It may be a more difficult challenge initially, but it pays off increasingly in the future.

Defining a SOA is not a trivial task. It is a discipline that requires heavy doses of analysis, strategy and understanding of the business. Most SOAs are built from the blueprints of an enterprise framework that defines the processes and components that make up the business model.

The enterprise framework is a tool for understanding the business model at a very high level. As Figure 16.1, “The Enterprise Framework” [140] shows, it defines the taxonomy for business processes and how they are organized. In more mature environments, it can also define the priority of the framework components.

Figure 16.1. The Enterprise Framework



The framework is a view of the components that make up the business model and may include entities outside of the corporation. A true business model does not operate in a vacuum; therefore, the enterprise framework should include touch-points for partners, customers, vendors and others who interact to create the business model. In the above model, customers interact with the front office components, while vendors and partners are hooked into the extended office components.

Chapter 17. Planning

In this chapter we will cover a lot of different ideas on how to plan for implementing a SOA in your enterprise. We will focus our discussion on six specific areas.

- The Impact of Change

  A discussion of logical and technical migration issues.

- Web Service Management

- Service Continuity

  A six step process of achieving control of your SOA.

- Achieving Integration

  A discussion of user integration, business process integration application integration and data integration.

- Data Transformation and Rationalization

- Architecting an Effective Solution

As we go through each of the six areas pertaining to planning a migration to SOA we will find many of the concepts that we have already discussed in the previous chapters being discussed here from a very different point of view. Some of the benefits and challenges of SOA that we have seen should be addressed in the planning stages and so you will be familiar with many of the ideas presented here.

We start our discussion on planning by examining the impact of change.

Examining the Impact of Change

One of the facts that life teaches us is that there will always be change. No matter how much we want things to stay the same, we know that given enough time there will be change. In spite of knowing that things will always change, we fear change and try to prevent it. When dealing with SOA, we need to examine and plan for the

impact of change, both in the technical regions of the enterprise and in the minds of the people and in the 'concept' regions of the enterprise.

Logical Migration

Any change within a system will affect the people that use the system. It is therefore necessary to examine the logical migration, the change in the minds of the people, as we examine the impact of change. We can architect the technology for change, but changing the people is another matter altogether. The greatest change management challenge facing organizations today is the breaking down of the business/technology dichotomy. Now more than ever, business people must be tech-savvy and techies must be business-savvy.

Much has been written on the subject of change management regarding staff member and end user education. For our purposes much of this information is unnecessary, but it would be a good idea for you to investigate the subject and maybe even hire a change management consultant when you implement a service-oriented architecture within your business.

Service Oriented Architecture is as much a mental system or process as it is a technical one. If you do not change the mindset of a company to embrace the ideas behind SOA any advantage brought to your business from a technical installation along SOA lines will be short lived and the likelihood of achieving the benefits of SOA is greatly reduced. Jason Bloomberg of ZapThink [bloomberg] [250]tells us that: "As enterprise architecture, SOA addresses a range of business problems through the application of IT resources. As such, it is essential to a successful SOA roll out that the business drive the initiative." By this, he informs us that the business people within an enterprise must be the driving force behind the change within the company.

Such change doesn't happen automatically or overnight. People are naturally resistant to change and some are more resistant than others. It is absolutely essential, however, that companies foster increasing numbers of specialists who are both business- and tech-savvy. First are the enterprise architects, who must maintain a broad picture of the structure of the entire extended enterprise, including both business and technology. Second are the business analysts, who are tech-savvy business people who understand the intricacies of business processes and are able to translate business requirements into process changes and service descriptions. The business analyst role has been a nebulous one up to this point at many companies, where many individuals with the title have little if any tech-savviness. In the service-oriented world, however, companies will need business analysts who can

work with architects to define services and incorporate them into service-oriented processes. These specialists can be the champions of the change that is necessary within the enterprise.

If logical migration is not tackled early on, then end users of services within an enterprise will not buy-in to the change. They will resist it. Managing change entails more than just tools and training - it also requires a commitment to the process. SOA is as much about change management, governance and skills acquisition as it is about technology. This is the point that most people miss.

Technical Migration

Technical migration is usually one of the first things people consider when thinking and planning a SOA. We like to think in terms of physical changes that will need to take place; new computers and network infrastructure, new software packages to evaluate and implement, decisions about which cross platform architecture to invest in and how to incorporate what is already part of the enterprise system.

Most of the key concepts of SOA involve some sort of technical migration. New software is often needed in order to increase interoperability and loose-coupling. New computers and new network structures are needed in order to create the necessary service repositories and services. Examining all of these options needs to be incorporated into the planning stages of migrating to a SOA.

IT departments will also undergo change as an enterprise moves toward a SOA. The question of whether to build, buy or adapt systems has to be answered while planning the migration. Specialists, such as the system architect, along with the chief information officer (CIO) and managers from the finance and business operations should be involved in assessing and managing the technical migration.

Once the planning of the hardware has been done, then the implementation of the technical migration can take place. The process of taking out old systems and installing new ones must also be carefully thought out and planned with emphasis given to reducing system downtime and maximizing system stability.

Migrating to a SOA will be done in phases, so a technical migration does not have to involve everything within the enterprise on a particular day. We cannot say, "Today we installed the SOA." SOA is a concept, a guiding business view that utilizes technology and business processes.

We do not need to go into technical migration in any great depth, expect to note that

in order for a SOA to function correctly the right tools must be available. The cost of putting these tools in place must be seen in light of the benefits of SOA, particularly the return of investment and cost savings. If you do not understand how migrating to a SOA can help your business, then you will not see the benefit of installing any new technology or upgrading your existing systems. If you do not understand the benefits of SOA your business, will lose the competitive edge to those who do understand the benefits of SOA and are in the process of implementing their migration.

The impact of change will vary from organization to organization. This makes it even more important to examine the impact of change on the people of the organization as well as on the technical equipment of the enterprise. As many SOAs have a technical focus on Web Services, we will now deal with some of the planning processes and requirements that need to be addressed within a Web Services platform.

Web Services Management Platform Requirements

During the planning and operational stages of migrating to a SOA it is useful to separate out the management of the Web Service from the management of the platform on which the service runs (the Web services platform.) Think of the Web Services platform as the interpreter of SOAP messages on their way to the Web Service itself. By making use of its understanding of SOAP messages, the Web Services platform can help the developer by taking over parts of the management functions.

Many vendors have Web Services platform products on which developers will build their Web services and those vendors will also supply some management capabilities with their products. However, developers cannot rely solely on the management features that come with the platform; you'll need to build some extra functionality to ensure your Web Service is indeed manageable.

The key concerns in managing Web Services begin with runtime instantiation and responsiveness to requests. These issues include:

- How the Web Service can be instantiated or started up;
- How multiple instances of the Web Service may be handled concurrently;

- How the load is being dealt with-by one instance or shared across those instances;
- How any instance of a Web Service may be shut down;
- How the operations staff will monitor the stability of a running Web Service at any time;
- How the loading characteristics on that Web Service can be discovered and presented to a manager;
- How the operation of the Web Service can be changed.

In order for management data to be extracted, a Web Service must conform to certain interfaces or rules. These rules will be the "management interface" to the Web Service and are made up of appropriate responses to messages for:

1. Starting up the service;
2. Shutting down the service;
3. Checking that the service is alive and responding;
4. Re-initializing the data structures or component objects that make up the internals of the service;
5. Requesting the service to report the number of concurrent messages it is processing at any time;
6. Requesting the service to report the number of messages in its queues (inbound/outbound) at any time (if it implements queuing of requests);
7. Requesting that the Web Service display its identifications, including its current version number;
8. Requesting that the service display its current set of dependencies (other Web Services);
9. Requesting that the service send any error messages to a named target or file;
10. Producing current load statistics-including the "weight" of any type of message

dealt with, such as its byte size (This is particularly relevant in document exchange scenarios.);

11. Producing unsolicited events or warnings of occurrences within the service that require attention;

12. Decommissioning the Web Service at the end of its lifetime.

Requirements 1 through 3, 5 through 8 and 12 may be performed by the Web Services platform rather than the Web Service itself. However, the Web Service developer must implement the cases where the management platform should be notified of unexpected internal events. This is also true of certain internal properties of the service, such as in requirements 9 or 10. The operations shown above may be included in a common management WSDL that all managed services should obey.

In the planning stages of a SOA the management team needs to be aware of many things. All the stages mentioned above are merely pointers to help you deal with the subject matter of Web Services in a SOA. We will take service management to a deeper level in the next section, as we discuss service continuity.

Service Continuity - Improving and Keeping Everything Going

While core Web Services standards have successfully addressed the mechanics of getting applications to communicate to one another, the success of SOA implementation must address the challenges that lie beyond the pure mechanics of communication. The number of complexities can be overwhelming: -- stakeholders with different agendas, policies with cross-functional implications, service levels that must be maintained at all costs, complex interrelationships between services and no lines painted on the data center floor to connect any of the dots. If we left it to grow on its own, a network of Web Services would quickly degenerate into a tangled spaghetti of brittle, single-use integrated applications and fail to achieve the economies of scale or the cost and flexibility benefits of SOA.

These challenges call for a new breed of solution, one that addresses the various technical, business and organizational requirements and coalesces all pertinent knowledge in the SOA into a form that's understandable and actionable. Dan Foody (an active participant in the Web Services standards community, including WS-I and OASIS) proposes a solution that he terms 'SOA Command and Control.' By

examining this solution, we will see a clearer picture of service continuity

In his article on SOA Command and Control [foody] [250] , Foody presents five key imperatives of SOA Command and Control: align, comply, observe, respond, optimize. These five imperatives encapsulate the required and the associated value, of SOA Command and Control. Let's discuss each of them in some detail.

Align

IT is successful only if the business is successful, so IT must always be aligned with the business. The SOA Command and Control solution must allow a company to measure SOA activity against its business objectives in order to understand its current impact on the business, to determine how it's trending and to predict where it will go in future. SOA Command and Control should also let you customize or tailor the services offered by your SOA quickly and easily to address tactical business opportunities. With SOA Command and Control you can:

- Understand which of your most important consumers, regions or divisions are getting the best service;
- Provision services to meet the unique requirements of a new high-value consumer;
- Plan and execute a migration to a new version of a service so that your standard customers roll onto the new version and live test it before your most important customers.

Comply

True SOA Command and Control empowers stakeholders to move from a passive role to an active role of driving policy changes immediately and automatically across the organization. In addition, stakeholders gain visibility as to where and when the policies and procedures are being applied and/or violated. With SOA Command and Control you can:

- Uniformly apply and enforce a security policy across many services without a service-by-service development effort;

- Become compliant with changed privacy regulations;
- Define what service levels can be promised to different consumers.

Observe

By definition, SOA Command and Control provides both detailed and at-a-glance visibility into the inner workings of your SOA at any point in time - automatically, without expensive, time-consuming manual configuration. This facility lets you understand SOA-wide patterns and trends that would never be uncovered with solutions that provide simple statistics and only threshold, rule-based or predictive alerting. With SOA Command and Control you can:

- Understand what your service level is right now, broken down by geographical region, customer segment and service;
- Discover who depends on which service and what their different usage patterns are;
- Determine what "hotspots" are most vulnerable to attack or which will be the first to fail as loads increase.

Respond

When a symptom of a problem is detected - whether it's a functional issue with a newly rolled-out version of a service, a malicious attack or a performance issue - you need to be able to determine the root cause of the problem and how to respond to it effectively. Root cause analysis is important in a SOA because symptoms rarely appear at the location of the root cause and the root cause may be a service owned by a different group. With SOA Command and Control, you can accurately and automatically determine the root cause of problems, without expensive, time-consuming manual configuration of rules or relationships. And, once you determine the root cause, you can respond in one of many different ways such as notifying administrators, black-listing users, rolling back service changes, rationing capacity or modifying documents in transit. These responses can be triggered manually, fully automated or even manually overridden when automated responses don't produce the desired result. With SOA Command and Control you can:

- Roll back consumers to use a previous version of a service if a problem is encountered during the migration;
- Enrich or cleanse incomplete or ill-formed documents rather than reject them;
- Continue to meet the service-level expectations of your most important customers while under unexpectedly heavy loads.

Optimize

As with any IT infrastructure, services have a finite capacity to process consumer requests. Determining the capacity requirements of services is especially complicated because each consumer has a different pattern to use with different kinds of requests and different peak usage periods. And, as new consumers come online, they consume capacity from the service and potentially affect the service level of everybody else. SOA Command and Control lets you both pro-actively and reactively optimize the allocation of scarce service resources. With SOA Command and Control you can:

- Offload resource intensive processing for frequently changing policies;
- Predict what changes will cause your service to become a bottleneck, breaching the service-level expectations of consumers;
- Ration capacity, giving greater capacity to more important consumers or reserving specific servers for high-value customers.

Achieving SOA Command and Control

Traditionally, the five imperatives of SOA Command and Control have been addressed directly by development or line-of-business application teams as part of the applications they build. Because application teams are intimately familiar with their own applications, the strategy of building security or operational policy into the applications was often effective.

In this environment, the role of cross-functional IT stakeholders, such as security officers or deployment architects, is to write documents that outline the policies that application teams must follow. Unfortunately, by delegating the implementation of

policy to each application team, you lose economies of scale and control over whether the policy has been correctly understood and implemented. Worse still, to change policy requires a complex one-off project for each and every application or service.

Because of the limitations of this model, products have emerged that attempt to empower cross-functional teams to gain complete control over policies that span applications and services. Unfortunately, while this model appears effective on the surface, it breaks down because these cross-functional teams don't understand the business context of the different applications and services. For example, if a change to privacy policy requires that all personal identities be signed and encrypted, the cross-functional teams will have no idea what data in any of the tens to thousands of documents are classified as personal identities.

What is required instead is a fundamentally new approach to addressing this challenge. Instead of providing tools that let one team or another be responsible for every aspect of the problem, you need a solution that enables the different stakeholders to collaborate effectively - each maintaining control over their own areas of responsibility, while seamlessly sharing the knowledge necessary to let the others do their job once.

With an effective SOA Command and Control infrastructure, policies will not only be defined once, centrally, but will also automatically be enforced in the fabric of the network itself. The capabilities of an effective SOA Command and Control platform lets organizations bypass the knowledge gap and successfully achieve the economies of scale as well as the critical cost, time and flexibility benefits of SOA.

The next topic that we need to discuss is the whole arena of integration. Within the planning phase of implementing a SOA it is important to recognize the various levels of integration needed.

Achieving Integration

The functional goal of SOA is integrating the business logic of complex applications into re-usable chunks that are easily accessed, used and understood. No company really wants to deal with integration. The only reason for anybody to spend money on integration is because software systems as a rule don't integrate by themselves. But no executive thinks that spending money on integration addresses a strategic need of the business. Instead, money spent on integration goes for fixing something that really shouldn't have been broken in the first place. The sad fact of the matter is that in the forty-plus year history of distributed computing, integration has

constantly been a cost for every company with two or more computers that need to talk with each other. With SOA this is beginning to change.

As we go through each of the sub-sections on integration, we will be looking at the various parts of a SOA from the inside. In order for us to understand the parts more clearly we will be using the example of the bicycle manufacturing and sale enterprise that has embarked on improving its business functions through implementing a SOA.

User Interaction Integration

In the past there have been many single-use programs to collect user input and use that input within an application. With the development of shared and reused services within a SOA there has been a standardization of user input and user interaction opportunities. Typically, these are controlled by the front-ends of services or the composite applications that access the various services used by the enterprise SOA. Fully integrating user interaction within a system will enable increased levels of abstraction and interoperability, enabling increased usability and user satisfaction.

In our Bicycle Company example we can imagine a web site that advertises custom built bicycles. The web site also takes orders and initiates the payment services, assembly services and delivery services. The user who visits the web site will see one screen, but multiple services will be integrated within the context of that screen. The user selects the bicycle and enters all the details needed for the business processes to start.

The user also needs to be integrated into the tracking system, either through the user logging back in, on the web site or via email or sms (text message) technology, in order to determine where the bicycle is along the chain of manufacture, assembly and delivery. It may be that some parts for the bicycle are made in a factory in another country and are still in transit. The bicycle assembly is therefore not finished and the user can find out when it will be completed. Because users are fully integrated into the system through a complete data rationalization and process integration, they are able to discover what is happening with their order in real time.

In any enterprise it is the end user or customer that provides the market for the business product, whatever it is (item or service.) By fully integrating the user interaction into the SOA, you gain increased customer satisfaction. In addition, you will also have a better understanding of your customer's needs and wants, making your enterprise more able to respond to specific market demands.

Business Process Integration

Companies have long sought to solve business process issues by implementing Business Process Management (BPM) approaches that consider systems and IT assets as activities or tasks that participate in well-coordinated and centrally organized business processes. Traditionally, the challenge of BPM is that while it is possible to construct processes that achieve integration goals, enterprises typically use BPM tools only at design time, modeling processes as they used to be or processes as they should be, but rarely processes as they actually are in the IT environment.

So, while BPM solutions can craft orchestrated processes that are composed of fine-grained services, they don't contain the runtime environment necessary for the loosely coupled, asynchronous service interactions required by a typical SOA. At the very least, a BPM solution could be used in conjunction with a loosely-coupled integration approach to make the business processes runtime activities that coordinate integration. Thus, by itself, BPM solutions aren't sufficient to meet SOA requirements. They also need to evolve within the SOA arena.

Business process integration is very important within a fully functioning SOA. It is within this layer where most of the challenges faced by SOA lie. For the Bicycle Company, business process integration brought about a different way of doing business. Before becoming service oriented, the company focused on assembling various types of bicycles and storing them, waiting for an order. With the integration of all their business processes the company was able to shift into an order driven assembly that required no storage of the assembled bicycle. By integrating other companies into its SOA, the bicycle company was able to produce one meta system that tracked all the parts advertised on the web site from many different factories and optimize ordering, delivering and assembling processes. Within its SOA the company incorporated orders, payments, credit facilities, parts delivery, assembly, final product delivery and customer feedback. The services needed by the business processes were able to be integrated, even though they involved many different companies and systems.

The end result of business process integration within a SOA is enhanced business flexibility, reduced overheads and clearer function definition. This gives the enterprise the competitive edge that it needs in the international market economy of today.

Application Integration

SOA allows the functions of legacy applications to be integrated along side new business logic in a system. By providing ways of wrapping applications and creating ways for them to interact with other processes and applications, SOA encourages the creators of applications to integrate the various levels and functions of those applications. This may be technically difficult for many proprietary applications, but the use of common standards and middleware has now made application integration much easier than it was before.

Application integration is an IT challenge and it is one that the programmers shake their heads over until they come to grips with the advantages offered within a SOA. It is at this level where 'middleware' is mostly being used. All the various applications, operating on different systems in different places, need to be integrated within the SOA system. They do not all need to communicate with each other and be fully interoperable with every other application, but they all need to be connected with the enterprise function.

It was not necessary for the Bicycle Company to fully integrate the delivery tracking system on its web site. All the company needed to do was to create a service that supplied the user with the tracking number of the order and a Web link to the delivery company's tracking site. The user doesn't need to know how these applications were integrated, but the enterprise administrators do. This integration was planned and modeled from the beginning. The system programmers created a Web Service between the delivery company and their own site, matched the order number with a tracking number and emailed that to the client as well in addition to displaying it on a Web page that the user could log into using the order number.

The Bicycle Company had many other applications to integrate, from accounting packages to inventory systems. The important part of this integration was the process management that resulted from it. Not only was the customer able to track what was happening with the order, but the Bicycle Company could also track and correct things that needed correcting.

Although application integration is an IT challenge, it is also a major business benefit offered by the SOA model. Not only can you reuse legacy applications along side new ones, you can also integrate applications running on different operating systems, written in different programming languages, using different security protocols and different data storage facilities. By creating a system of services, many parts create a specific whole.

Data Integration

The next section the section called “Data Transformation and Rationalization” [154] deals with data transformation and rationalization, key elements of data integration. Data integration is an important step in the process of migrating to a SOA.

Data integration can bring many benefits to the enterprise in terms of report generation and market trend analysis. It does, however, require a full understanding of data modeling and metadata facilities when used on an enterprise wide level.

Choosing the Right Integration Solution

There are many software companies offering integration tools and platforms. Metadata services and the Enterprise Service Bus are both central tools for this solution. When looking at an integration solution, we need to focus both on what is needed now and what will be needed in the future. We understand that we cannot know the future for certain, but by utilizing an integration solution that allows greater flexibility and does not lock the services into a particular narrow application, we can hedge our bets and allow for change. We need to integrate the core necessary functions of the enterprise and then allow for flexibility and interoperability with other systems.

Choosing the right integration solution should be part of your SOA strategy plan. As you look at your own organization and the market in which you operate, you will be able define your core needs and match those needs to services that integrate them at the user, business process, application and data levels.

Data Transformation and Rationalization

Databases and data transfers are at the core of most (if not all) enterprise resource planning systems. The underlying structure of all enterprise systems will have a database of some kind, either connected to it or as a central feature. For this reason, one of the biggest challenges that enterprises face when migrating to a SOA is how to manage their data. Data rationalization involves dealing with the condition of the data within these databases, with semantic incompatibilities (different definitions of the same data,) redundancies and discrepancies in definition that may exist in the data repositories that services will use.

Integrating data models in a complex enterprise can be difficult because the IT infrastructure often contains massive duplication and redundancy. Refining this situation without loss of data definition (semantic loss) is a tough nut to crack. Gartner, Inc. advises organizations wishing to fully exploit service-oriented business applications to focus on integrating the processes and underlying data models, rather than on integrating individual application components. Failure to integrate these aspects will place the organization at a competitive disadvantage. Gartner believes that such metadata management is "essential to reducing the escalating complexity of management and maintenance of integrated software platforms."

When looking at integrated data models to use to transform and rationalize an organization's data, these are some of the challenging questions that need to be addressed:

- Is an integrated data model a real, tangible model, or is it a logical concept only?

  An integrated data model must be real. In order to build the layers needed by a SOA you should collect in one place the metadata that describes all the data defining the SOA. For example, transformations in the interface layer are schema-to-schema mappings that can only be defined if the source and target schemas actually exist. Where you store it and how you manage it are very important questions that you will need to answer before implementing the SOA. Your decisions at this stage will have long-term consequences on the maintainability of the implementation of your SOA.

  It is not enough to merely have an integrated data model. It must be based on what is already available and on what is really possible with the introduction of new components.

- What modeling or schema language (flavor) should the integrated data model use?

  The integrated data model should ideally be an XML data model that is best described in XML Schema; after all, service payloads are constrained by XML Schema. Note that the expression of a data model - that is, how you choose to deploy it as something tangible and usable (i.e., a family of XSD files) - is not necessarily the same as the development image of the model. For example, some organizations capture everything in UML (...def..) and export the resulting model as XML Schemas. However, it is good advice to stay as close to the final implementation as possible.

This question needs to be answered in consultation with someone who specializes in database management and XML Schemas. For the purposes of this book and your being able to evaluate SOA offerings, only be aware that your enterprise needs to decide which XML schema will best suit your needs.

- How do you create the integrated data model?

  To achieve an integrated data model for an organization, a single, overarching model must be created that fully represents all the relevant underlying data models, including any schemas used for trading partners or industry standards. This is a serious data modeling exercise that typically requires the input of highly experienced analysts and architects. The end result is a set of custom standards for your enterprise. This might sound as if it goes against the principle of 'loose coupling,' but this overarching data model is the layer in which the data is 'coupled.' It is at this layer that all the data must be interoperable and standardized.

  The starting point is often a 'metadata-gathering' phase, where all the existing models are imported into a central place (usually a data repository.) Exposing metadata is not always straightforward, as some applications do not publish interfaces or schemas and others require some interpretation or embellishment along the way. For example, database schemas in an Oracle database can be exported as DTDs or XML Schema files using the XSL utility (XML-SQL Utility,) but ensuring that relationships and constraints are properly expressed in the resulting schemas requires a manual reading of the metadata.

  Be aware that importing metadata from existing systems creates an application-specific view of the landscape, warts and all. There is little point in creating application- specific integration models, as this does not abstract us high enough above the underlying application specifics. This exercise must therefore be accompanied by a broader analysis that models the actual data requirements of the enterprise, preferably with a very forward-looking appraisal of the expectations facing the business. You should look at this as a bonus. As Thomas Erl says in his article "Best Practices for Transition Planning" in the November 2004 issue (WSJ Vol. 4, issue 11,) "In planning a migration to a standardized adoption of SOA you have an opportunity to erase some of the neglect from the past." This is a speculative analysis action that certainly applies to the data-modeling phase.

  The data modeling phase also provides an excellent opportunity not only to erase some of the neglect from the past, but also to introduce other essential best

practices into your SOA, such as a service-oriented security model. At this metadata gathering stage, some organizations use metadata management tools, equipped with the drivers and connectors, to facilitate the import of metadata from various systems in the application landscape. However, before you rush out and buy a metadata repository, be sure you understand exactly what your long-term requirements of such a system might be, paying particularly good attention to questions of maintenance and future evolution.

Your current ERP system may already contain everything needed for a metadata repository.

- How do you resolve duplication and redundancy?

  When data models are integrated, some translation and rationalization are inevitable because duplication and redundancies must be resolved. It is essential that no object properties are lost or diminished during the rationalization phase, as this would result in a reduction of the metadata and therefore of the potential functionality. Rather, the resulting integrated data model should be more descriptive and functional than a simple sum of the component models.

  When rationalizing data models, we often have to interpret and manage the fact that two objects in separate application models are essentially the same object. How we deal with this depends on the requirements. If two objects exist because different teams of developers have made up their own different names for the object - and this is a very common problem currently in XML-driven systems - you can either straighten out the problem in the underlying models before integrating them or create a new "alias" object in your integrated model that can map to each variant.

  Another common problem is the use of the same name in underlying models for what are essentially different objects. The solution here is similar to the previous problem: either solve it before integrating, or integrate by creating new objects at a higher level. Note that when working with XML Schemas, namespaces can determine how you proceed. Clearly, if two teams have used the same name in the same namespace for different purposes and the act of integrating the data models exposes that problem for the first time, there is no way you can allow both objects to continue coexisting with different meanings.

  When two objects that are essentially the same object need to coexist for transformation purposes, the integrated model must describe both objects rather than attempt to resolve them into one, otherwise it is not possible to create the

transformation. For example, in a trading partner scenario where a legacy system from a supplier processes a credit card payment according to constraints described in a DTD (and your system speaks an XML Schema equivalent,) the correct way to integrate your data models would be to load both the DTD and the XML Schema equivalent and then create the mappings between the constituent objects.

- Is an integrated data model a passive reflection or an active master?

  Active metadata is business driven; passive metadata is technology driven. Once we have an integrated data model, the metadata takes on an active role. Prior to this stage, the role of metadata was passive because it served no further purpose other than to describe and constrain data. It reflected existing systems and application components. Active metadata drives a new development effort from within the metadata. In other words, when we need to modify the payload of a service to satisfy a changing business requirement, our starting point must be the externalized schema that describes the payload. This is a very serious consideration for tooling and evolution management. From this point onward, any changes to the way the business functions will force developers to go to the metadata first - that is, the integrated data model - and make their changes there before changing code.

- Where do you store the integrated data model?

  Bearing in mind the active nature of the metadata in your SOA, it is essential that you store the model in an environment that supports the concept of change. Repositories that provide container functionality and business analysis support are essential parts of the IT landscape before you implement your SOA. Once you start implementing the SOA, the integrated data model must be managed in a model-driven environment with full developer support. This allows changes to be made centrally and deployed out to the environment in the form of version-controlled schemas, transformations and so on.

- How do you manage the life cycle of the integrated data model?

  The term that is applicable to managing the life cycle of a data model is "metadata evolution management." This is at the heart of any SOA, because SOA is a development- time and deployment-time concept that requires an integration platform that orchestrates Web services. Web Services are described by and carry payloads that are constrained by metadata that is expressed in XML Schema. When systems evolve, the metadata must also evolve, thus making

XML-metadata evolution management an essential part of the infrastructure.

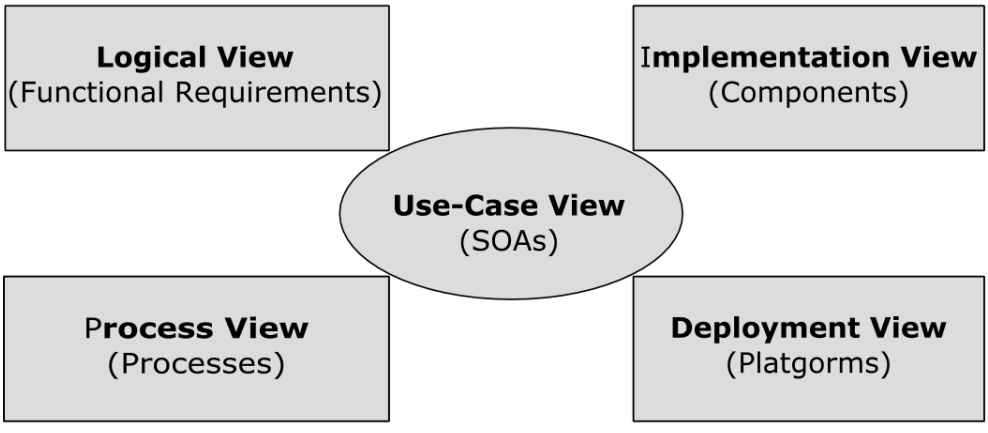
Suffice to say, managing the life cycle of your Web Services development, particularly from the perspective of the evolution of metadata, is not simply a schema-versioning problem. Versioning schemas are about technical constructs and development processes, not about the management of metadata evolution. Metadata evolution management is the real problem facing the long-term life-cycle management of Web Services development projects.

In presenting ideas regarding the planning stages of implementing and SOA, we have discussed the impact of change within an enterprise, the role of Web Service management, service continuity, integration and data transformation. We now bring everything back together and look at the big picture as we discuss some models for planning and architecting an effective SOA solution for your enterprise

Architecting an Effective Solution for your Enterprise

It is important to see the big picture as you plan a Service Oriented Architecture. It is also important to maintain the view of the big picture as you create the architecture and migrate your enterprise. Philippe Kruchten of ZapThink proposes a '5 view model' of architecting a SOA.

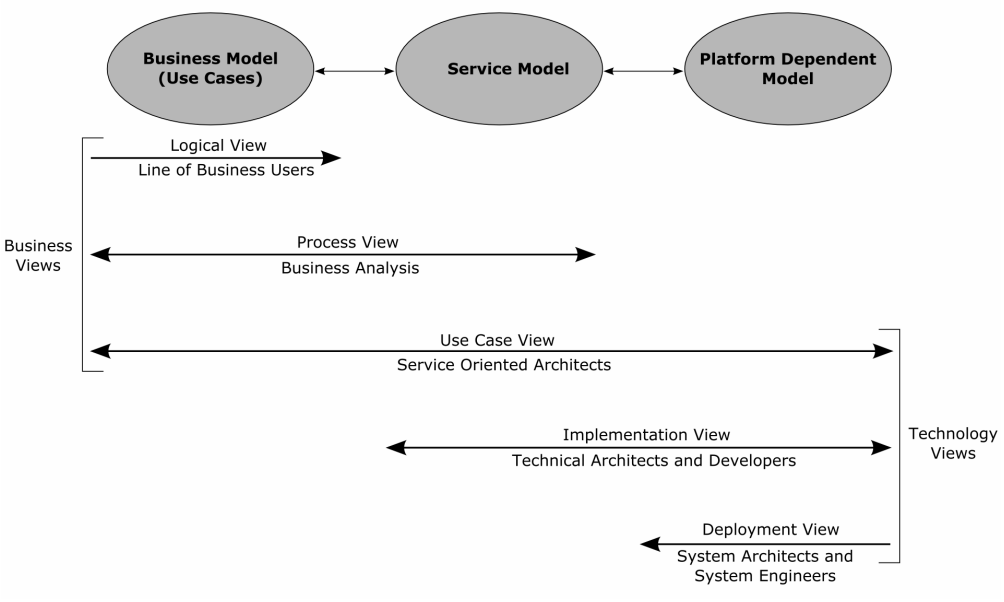
Figure 17.1. The 5 View Model



The four rectangles in Figure 17.1, “The 5 View Model” [160] represent different ways of looking at an architecture or the perspectives of the involved stakeholders. The fifth view, the use-case view, overlaps with the other views and plays a special role with regard to the architecture. The deployment view maps software to the underlying platforms and associated hardware, the way that systems specialists view the architecture. The implementation view describes the organization of the software code and is the view favored by programmers. Business analysts work with the process view, which addresses the software's runtime issues. Finally, the logical view represents the users' functional requirements. In the case of SOA, the service-oriented architect must be able to connect the users to the services and the services to the underlying technology, following the threads of use cases in the use-case view.

To show how the service-oriented architect must work with each of these views, let's put them in the context of a SOA meta-model, as shown in the following figure.

Figure 17.2. Practice of SOA



This image shows the two overlapping realms of SOA: the business realm represented by the business model and service model ovals on the left and the technology realm represented by the service model and platform-dependent model ovals on the right. The service model thus becomes the point of contact between the business and technology realms and acts as the channel for communication across the enterprise. Business users who use the logical and process views work with coarse-grained business services, making them into processes as needed, depending on the fluctuating requirements of the business. Technologists, on the other hand, work to build and maintain the abstraction layer between the services and the underlying technology. The central model, representing the services themselves, acts as the axis around which the business moves.

Remember that true SOAs are always responding, adapting and changing - always in a state of flux. This environment of constant change is the cornerstone of the practice of SOA shown in Figure 17.2, “Practice of SOA” [161]. The stakeholders shown in this figure continue to effect changes throughout the architecture on an as-needed basis. The line between design time and runtime blurs, as the technologists respond to changing business requirements as a normal part of day-to-day operations.

Now that we have a better understanding of the planning of a SOA, let me briefly outline some guideline steps to put this into effect:

1. Develop and maintain an architectural vision;
2. Don't assume you can build your SOA just by evolving the technology you already have;
3. Establish the infrastructure to make your SOA mission-critical early in the process;
4. Plan for new developments in the future;
5. Be prepared to move forward incrementally;
6. Get started sooner rather than later.

Planning is an integral step along the path of SOA migration. Many things that we have discussed in this chapter can be tackled later in the migration process, but it would be easier to address them (or at least be aware of them) as your enterprise examines various SOA offerings and as you assess your specific business needs.

As we continue to discuss the implementation of a SOA within the enterprise environment, we will deal with the ability of a SOA to adapt within the world of change. This property of a SOA brings a profound advantage to any enterprise, but there are also inherent dangers, which we will discuss and show examples of in the next chapter.

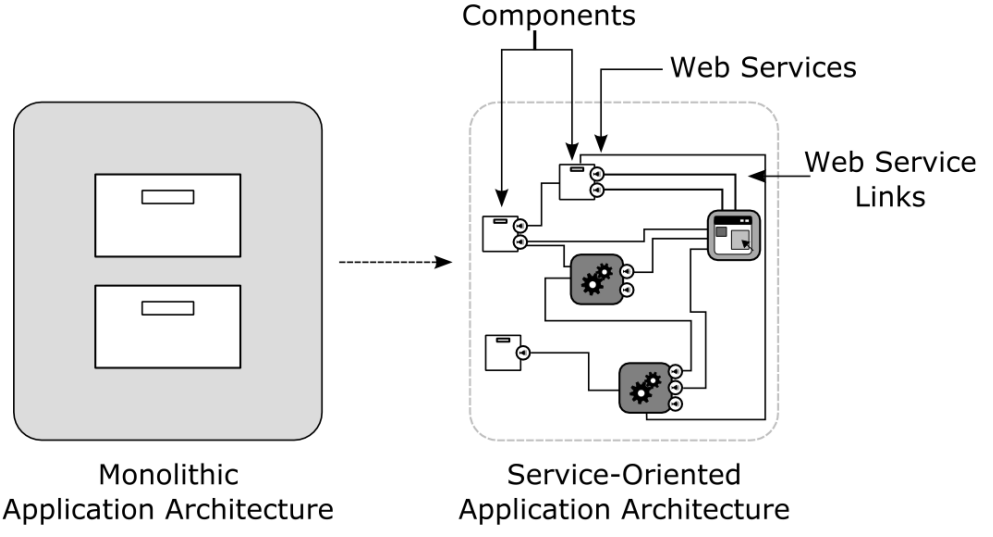
Chapter 18. SOA Adapting to Change

It is important to understand that any major changes at the enterprise level of business require an understanding of the process of change and the results of change. The article "Managing the impact of change in an Enterprise Services Network" [philjam] [250] in the online publication *Enterprise Architect*, covers these points extremely well. We will use the following points and issues raised within the article to increase your knowledge and understanding of the process and results of changing and moving toward a Service Oriented Architecture.

Increased Enterprise Responsiveness to Change

As we have seen, the service-oriented approach to enterprise software architecture replaces large, complex, monolithic applications with applications composed of loosely coupled collections of services and modular software components linked through well-defined interfaces.

Figure 18.1. From monolithic to loosely-coupled, simpler, modular software components



For the business, the result of a Service Oriented Architecture is movement toward what is often referred to as the "real-time enterprise." Because applications have well defined service-based integration points and because these services are built on standards embraced by every major enterprise software vendor, connecting software systems together will be orders of magnitude easier than in the past.

When an enterprise is able to move the right information to the right people and systems at the right time, it has increased its ability to identify and interpret changes in their markets and to respond by adjusting their processes, operating models or structure.

Increased Responsiveness to Market Change

The information systems that run the business are often modified as the business responds to changing market conditions. Frequently, these modifications may even require the creation or integration of entirely new applications. For the IT organization tasked with these activities, the result of adopting SOA is a decrease in cost and a dramatic increase in the rate with which these changes can be made.

Cost is reduced in the application development cycle because existing application modules and services can be re-used as building blocks, eliminating the costly and often repetitive effort that slows many of today's IT initiatives. In addition, because application modules have well-defined service interfaces that facilitate plugging and unplugging, it is far easier to initially (or eventually) outsource an application component or to purchase and integrate packaged application or business logic.

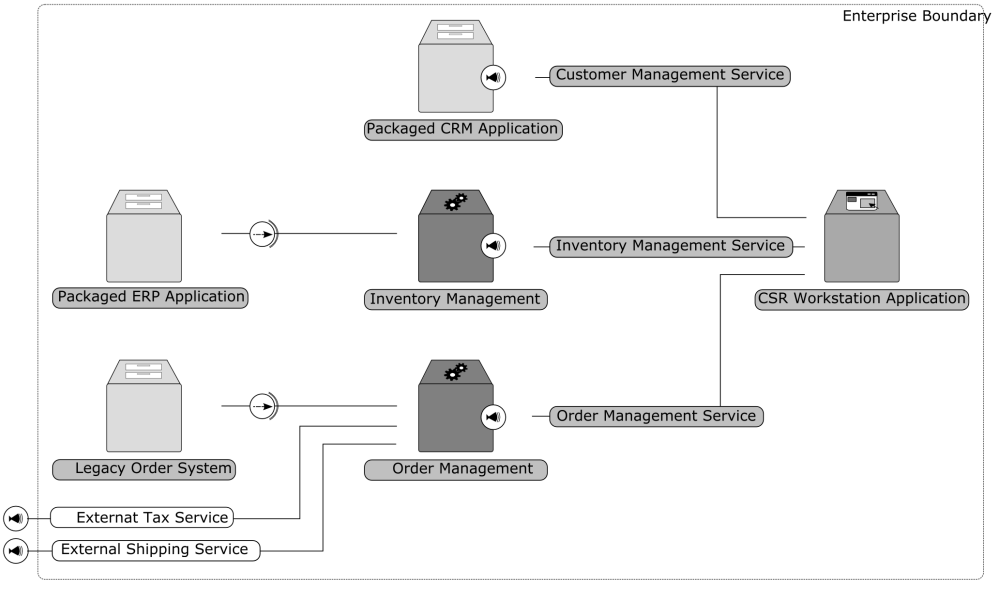
Velocity (the rate at which the system can change) improvements also result from using a SOA. Because the development of an application becomes more like an "application assembly" when building on existing services, development time is vastly reduced. Whereas application complexity was high with monolithic systems design, the modularity and relative simplicity of application components in a SOA makes them easier to understand and therefore also easier to enhance and change. In addition to lowering costs, the ability to outsource or purchase packaged solutions and then easily integrate them into the application environment also speeds up the rate of change or response.

The following scenario is presented in James Phillips' article "Managing the impact of change in an Enterprise Services Network" [philjam] [250]and brings these concepts to life by following the evolution of an enterprise service network. A manufactured goods enterprise has begun to build up excess inventory in a perishable product line. It must be cleared to avoid a material write-off. A business decision is made to discount the price on this line of products and to promote this offer to customers likely to purchase.

The business unit IT organization was tasked with creating a supporting application. Figure 18.2, “Initial project built with Web Services: Clearing excess inventory” [166] shows the result. The company had recently implemented a CRM system with a set of native Web Service interfaces providing access to customer information. The existing ERP system was not accessible via a Web Services interface, so an

application server was used (with an adapter providing ERP system access) to make inventory information available through a Web Service. The company's homegrown mainframe-based order management system was combined with two externally provided Web Services in another application server to create a Web Service for order management - query, add, delete.

Figure 18.2. Initial project built with Web Services: Clearing excess inventory



With the services in place, an application was rapidly assembled to execute on the customer service representative (CSR) workstation that would: query the order history of an inbound caller, examine the caller's historical buying patterns, look up inventory levels for the products most likely of interest and calculate an appropriate discount offer based on an adjustable algorithm. The offer was then presented to the CSR in the first five seconds of the phone call where it could be worked into the call script.

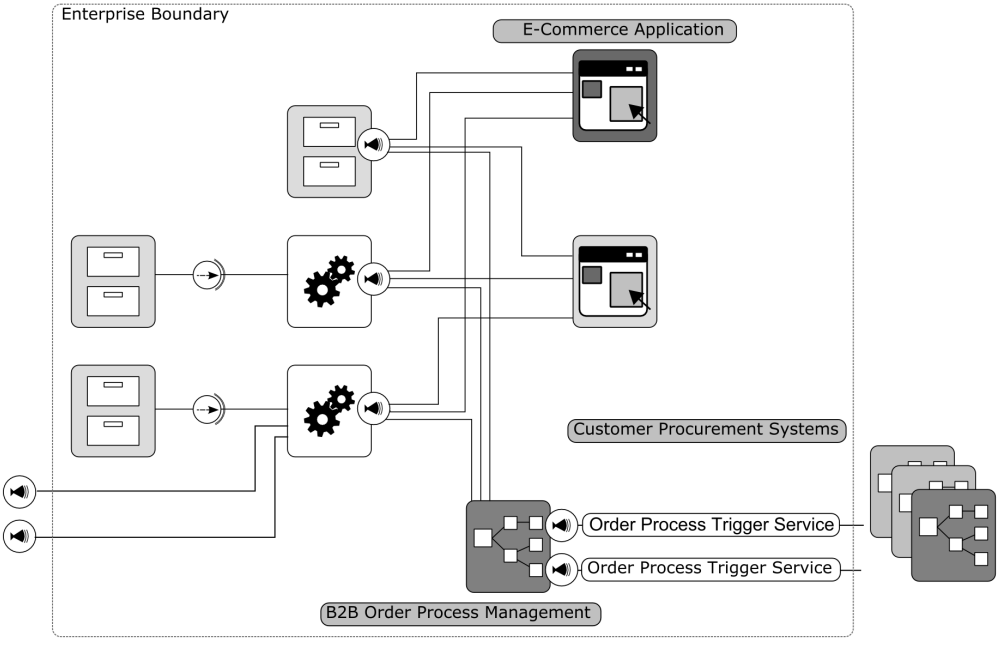
In this case, some services had to be created where they did not exist. But once created, the application development process was literally a matter of days.

Following on the heels of this success in an effort to build a more competitive cost

structure and to increase customer satisfaction, the decision was made to build an e-commerce site through which customers could browse and directly order products. In addition, the top five percent of this company's customers, based on order volume, would have their procurement systems directly connected to the company's order processing system - both accelerating the ordering process and creating modest switching costs for these valued customers (though this was not highlighted for the customers.)

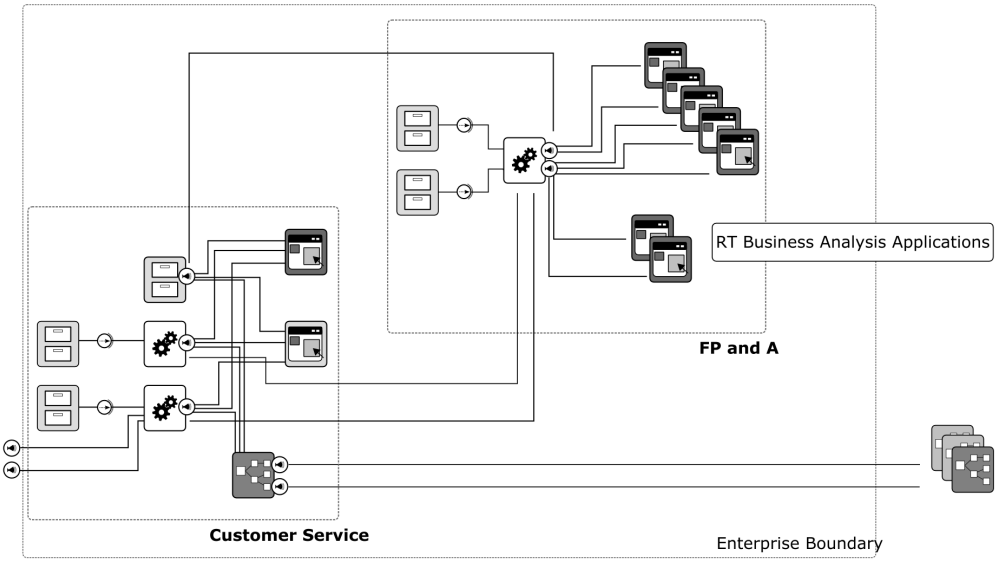
Figure 18.3, “Customers are quickly provided direct ordering access through e-commerce” [167] illustrates this solution. The IT organization has leveraged the same set of Web Services initially used to help pare inventory. For the procurement ties, a traditional EAI (process management) solution was used to manage the order process between companies. A process in the EAI platform leverages the available order, customer and inventory Web services to execute the business process that is itself triggered by the arrival of an order message to a new Web Service created in and exposed by, the EAI platform.

Figure 18.3. Customers are quickly provided direct ordering access through e- commerce



To prevent future surprises, like excess inventory accumulation, a decision was made to build a set of planning and analysis applications to analyze real-time business activity. Leveraging the inventory, customer and order management services already created and in use by the customer service organization, the Finance, Planning and Analysis IT organization built the system shown in Figure 18.4, “Web Services leveraged across the enterprise to power real-time business activity monitoring” [168]. The same services that allowed an aggressive response to an unfavorable business condition now help keep that same business condition in check.

Figure 18.4. Web Services leveraged across the enterprise to power real-time business activity monitoring

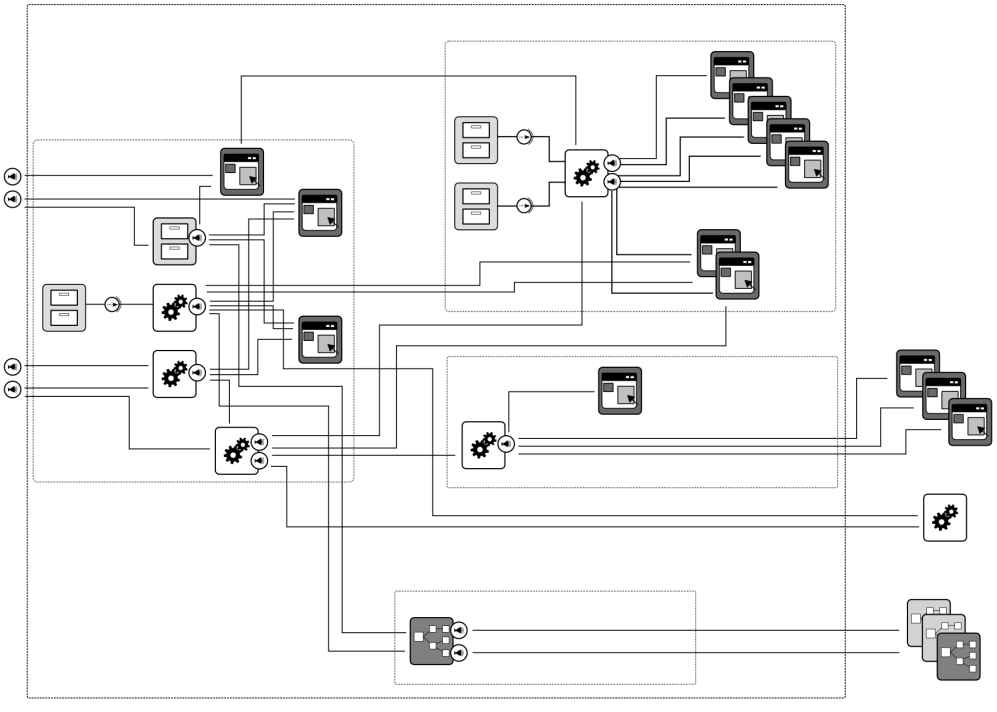


Building and introducing this system was done in a fraction of the time and cost that would have been required without the availability of these building block Web Services. And the real-time flow of critical information between these applications, required to drive this solution, was historically difficult to achieve at any cost.

Following this same pattern of innovation, systems are built, adjusted, configured and reconfigured in support of business decisions made in response to changing market conditions. Figure 18.5, “New services and applications continue to be built,

enhanced and connected" [169] provides a snapshot of this enterprise service network after more time passes, but while it is still on the leading edge of growth.

Figure 18.5. New services and applications continue to be built, enhanced and connected



Relative to past enterprise application architecture strategies, the ability of the IT organization to support and even enable changes to the operating model of the business is markedly improved.

In a SOA world, change is (relatively) easy and affordable.

Enhancing the ability to change the enterprise application landscape cost-effectively, rapidly and in support of changing business requirements is precisely why businesses are adopting the SOA approach.

Enterprise SOA - Large, Dynamic and Rife with Interdependencies

If change is made easier, then change will naturally occur at an accelerated pace. More new systems are completed more quickly. The applications of acquired companies are integrated into the IT landscape more rapidly. Connections with customers, suppliers and partners are built and rebuilt with greater speed. Enterprise service networks are, by nature, highly dynamic.

Enterprise service networks rapidly get big. The number of application components in a service network is much larger than the number of applications in an enterprise where monolithic applications reign. One of the primary reasons the service-oriented approach works is that it breaks monolithic applications into smaller, more focused application modules that are easily understood, modified and linked to form and reform applications as the needs of the business change.

One look at Figure 18.5, “New services and applications continue to be built, enhanced and connected” [169] will show that enterprise service networks are also characterized by complex interdependencies. The very notion of an "application" is changed in a service network. In the past, monolithic applications had very clear boundaries. In a service network, "applications" have fluid boundaries - they overlap and they are often assembled by leveraging components and services provided by other organizations within the enterprise, or even across enterprise boundaries.

Dependencies can be direct, indirect and circular. Application modules often both consume and provide Web Services. Applications are assembled by leveraging services provided by modules that themselves may rely on services provided by other modules that themselves - you get the picture - it's the one in Figure 18.5, “New services and applications continue to be built, enhanced and connected” [169].

The Impact of Change within a SOA

What do you get when you combine change, big networks of application components linked together in complex chains and interdependencies that are often unclear? (Other than a migraine.)

You get exploding costs associated with keeping the network running and meeting performance expectations - all while continuing to facilitate change (the reason for

embarking on a SOA strategy in the first place.)

As a service network grows, the marginal cost of change in the network grows at an accelerating pace. Three types of change in an enterprise service network dominate these rising costs:

- Uexpected change to a service in the network;
- Planned change to a service in the network;
- Planned simultaneous change to many services in the network.

Unexpected Change to a Networked Service

In a growing network of connected applications, service changes can occur suddenly and unexpectedly. Examples include:

- **Change:** A manufacturer faces a sudden increase in market demand for a product. Traffic increases to an e-commerce site as customers place orders. More calls come in to the call center where customer service representatives take product orders. Both of these systems link to a customer management Web Service being provided by a CRM system.
- **Result:** There is an unexpected decrease in the response time of the customer management Web Service.

- **Change:** A brokerage is using an external provider of market quotes for equity securities. The quotes are retrieved via a Web Service. The quote supplier's system providing the Web Service unexpectedly crashes.
- **Result:** The quote Web Service is no longer available.

- **Change:** The finance organization in a large retail chain has built a dashboard

application to monitor key business metrics. This application links to an inventory Web Service provided by a homegrown warehouse management system in one division. That division replaces the application with a packaged inventory management application.

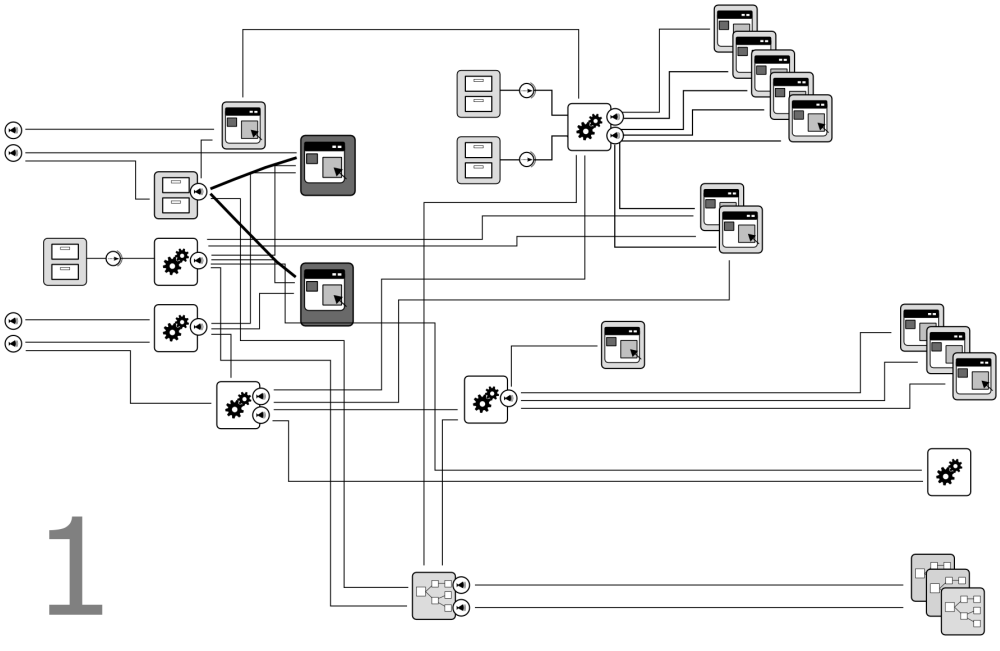
- **Result:** The inventory management Web Service has moved and the contract (WSDL) for that Web Service has changed slightly in the new application from that provided by the original system.

The impact of each of these examples to the service network is similar. As a further example of the impact of change, we will now trace the impact through six stages. The problem starts with the degraded performance of the customer management Web Service. That change ripples throughout the service network, eventually impacting systems that are several times removed from the initial problematic service.

Example of the Impact of Change

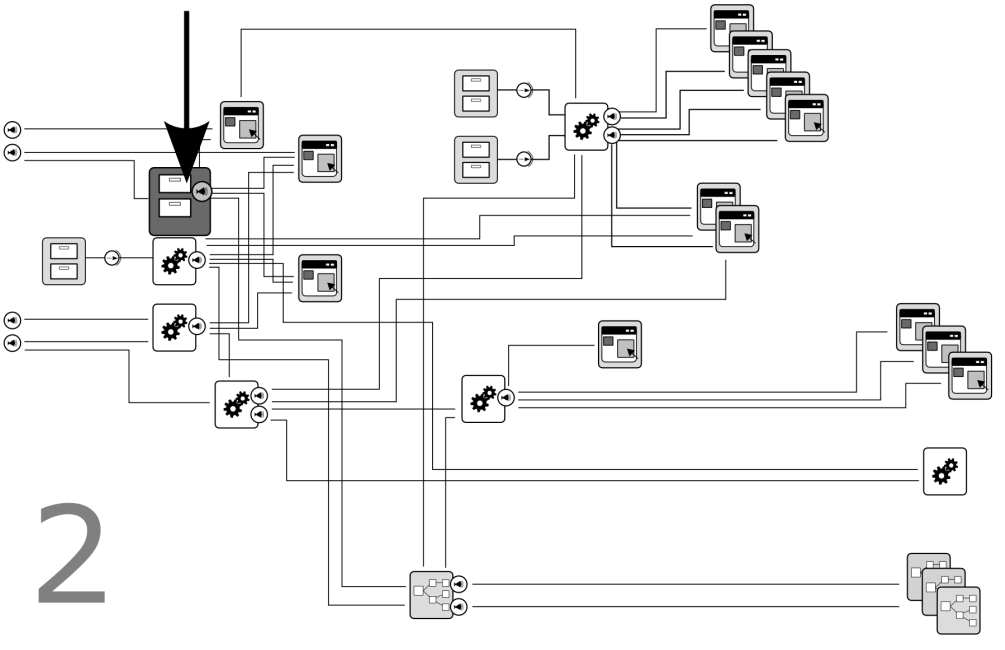
1. Market condition change: An increase in demand leads to increased utilization of the e-commerce and call enter order entry applications. These applications leverage the customer management services provided by a packaged CRM system. As application utilization increases, so does the number of messages sent to the customer management service.

Figure 18.6. First Impact



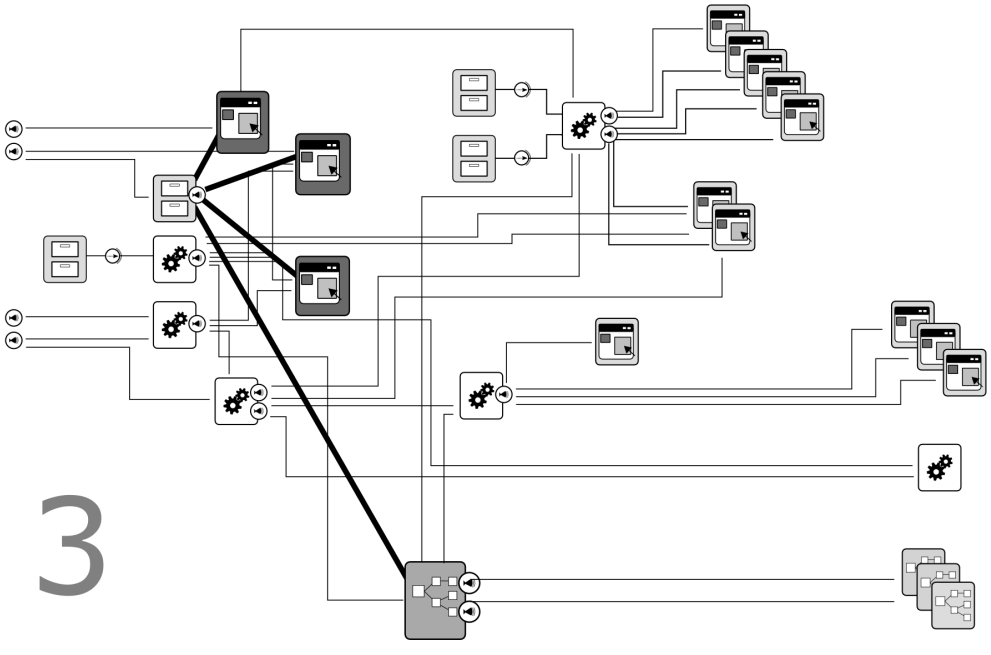
2. Impact to the service network: The increase in message volume to the customer management service degrades the performance of the CRM system. The result is a decrease in response time and throughput out of the customer management Web Service.

Figure 18.7. Second Impact



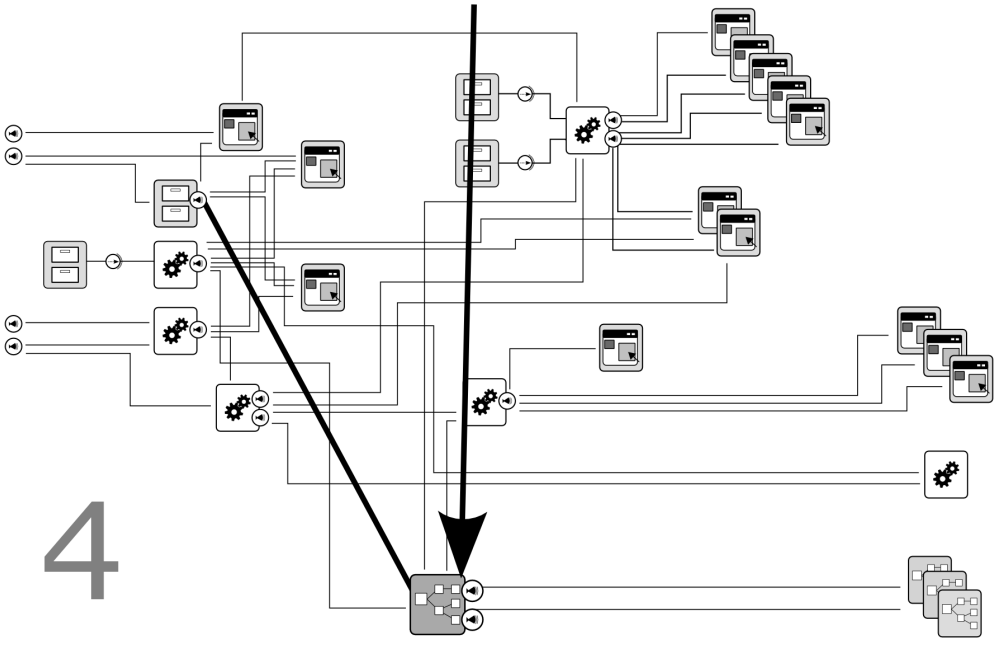
3. Impact to linked applications: Any application that depends on the customer management Web Service will now begin to experience performance degradation.

Figure 18.8. Third Impact



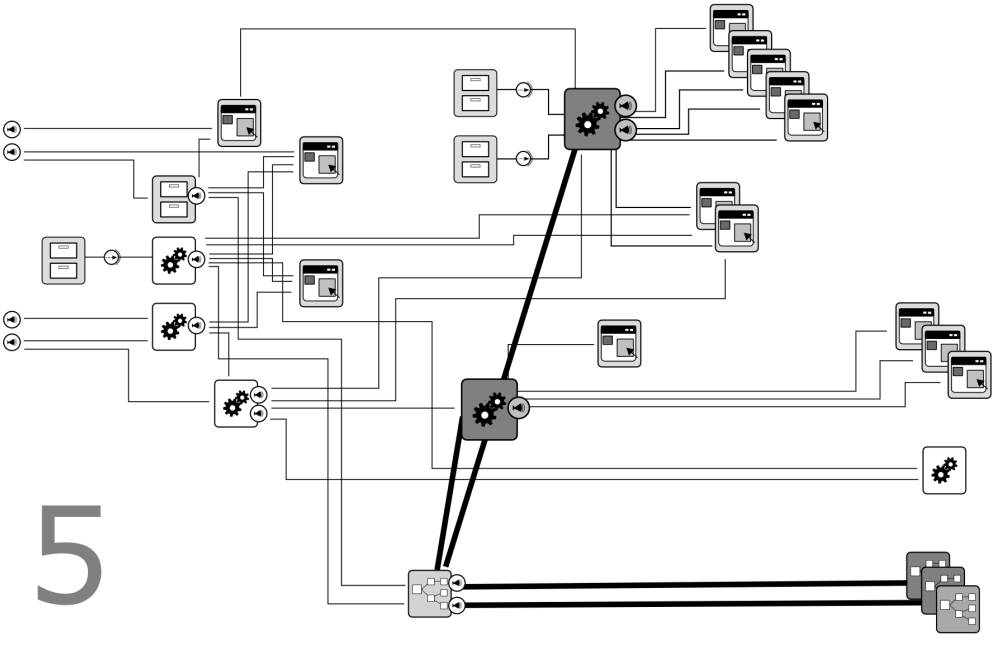
4. Ripple effect service network impact: Any upstream service that depends on the (now performance degraded) customer management service may now itself display a performance degradation in response time. In this case the EUA system providing the partner accepted order process Web Service trigger is impacted.

Figure 18.9. Fourth Impact



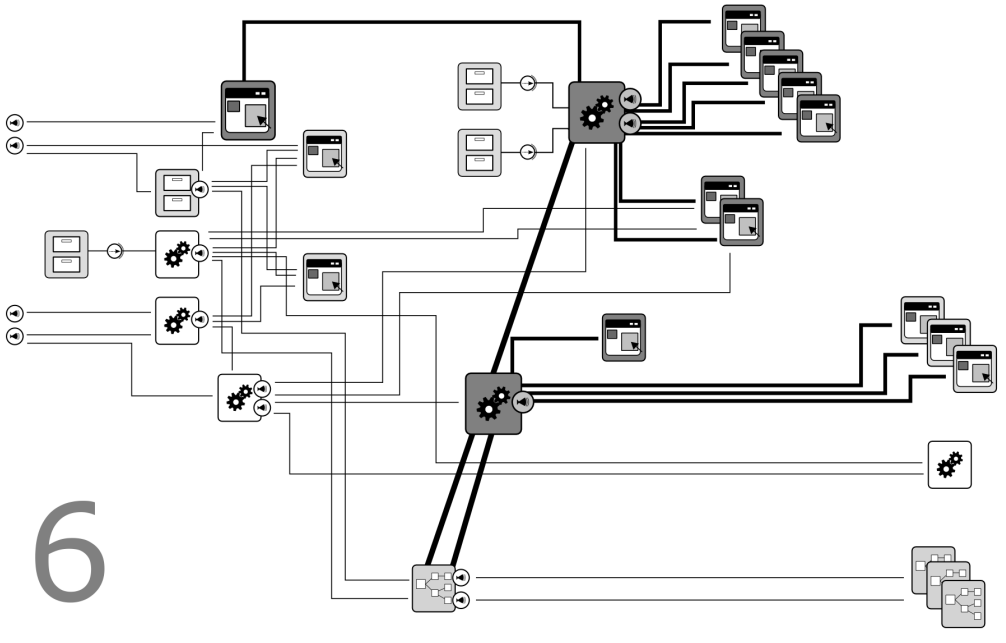
5. Impact to linked applications: As before, any applications which rely on accept order process trigger service (that relies on the customer management service) will now begin to experience degraded performance.

Figure 18.10. Fifth Impact



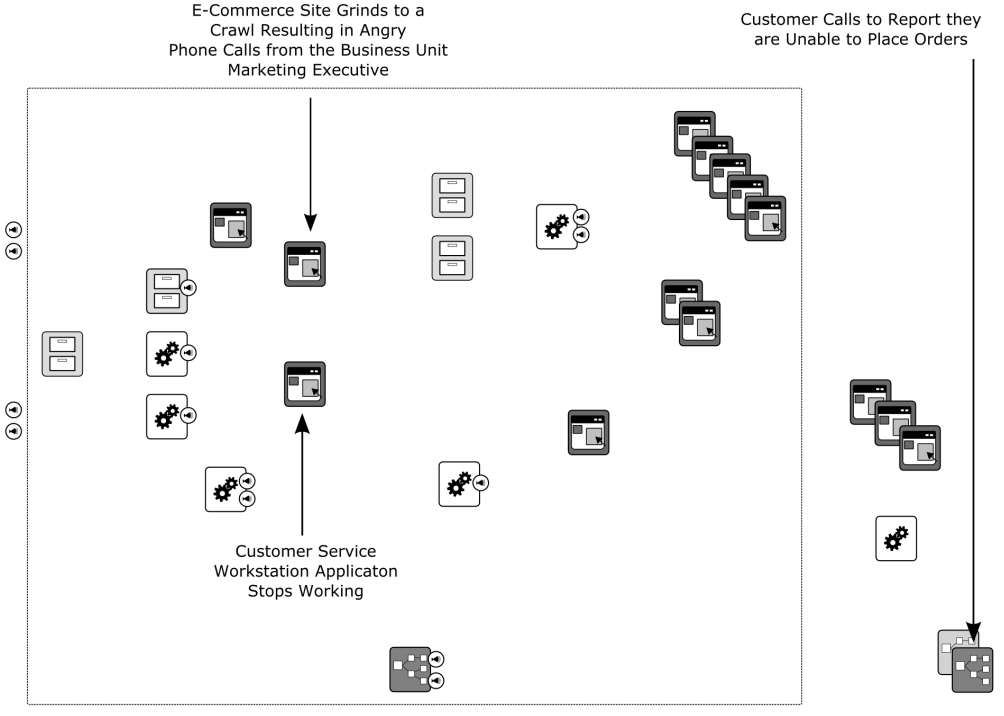
6. Continued ripple effect impact to service network and applications: Ripple effects continue as in previous frames until there are no more upstream systems acting as providers.

Figure 18.11. Sixth Impact



While Figure 18.11, “Sixth Impact” [178] is an ugly picture, the picture in Figure 18.12, “Failures appear to be random and unrelated” [179] is even uglier. The reality is there are no lines painted on the floor of the data center highlighting the interdependencies among the loosely coupled systems in a service network. There are no dials and gauges displaying Web Service performance. There are no alarm bells that sound when service performance moves outside normal bounds. There is no facility for tracing failures back through a tree of application interdependencies to get to the root of a service network problem.

Figure 18.12. Failures appear to be random and unrelated



Instead, the organization responsible for keeping the service network up and running is faced with what appears to be a bewildering set of coincidental system failures. And as Figure 18.11, “Sixth Impact” [178] shows, there are more to follow. These failures are expensive. In the example above, orders are lost, customers turn to competitors and valuable IT personnel are frantically troubleshooting problems, with limited insight, rather than innovating to improve the business. Even in a service network of a single service provider and one consumer, the inability to rapidly detect changing service conditions and actively deal with them can be extremely costly. As the service network grows, those costs explode - the impact of a service network change ripples through a larger network of business applications.

In order to effectively deal with this problem, there must be a facility to:

- Know there is a service network problem - that the network of services is deviating from its normal operating range;
- Identify the root cause of the problem - if it is a service network problem, which application, service and operation is at the head of the cascading chain of trouble;
- Insulate the downstream applications from the root failure - minimizing ripple effect costs and providing breathing room to address the underlying problem.

Planned Change to a Networked Service

In many cases, intentional IT actions may cause unexpected impact to the service network. These intentional actions can include introducing new applications; replacing and enhancing existing applications and Web Services; upgrading systems; bringing systems up and down for maintenance; relocating servers; and outsourcing applications.

Examples of intended change with unintended consequences include:

- **Change:** A pharmaceutical company has made the decision to replace a custom-built inventory management system with a packaged application. A collection of Web Services had been built in front of the old system to allow other applications access to the information and processes in the system. The new packaged application comes with pre-built Web Services, but while they are richer in functionality, the Web Service interfaces are different.
- **Result:** Service locations change, because the application moved and the Web Service interfaces change.
- I**mpact:** Systems that rely on the old version and location of the service (identified by a URI) will abruptly lose contact with that system, sending messages into a void. Even if the messages were to arrive at the new system, the service request message form would be alien to the application. Service requests fail. And ripple effects begin.

- **Change:** An aircraft manufacturer creates a new portal that will display real-time manufacturing process status. This portal is to leverage a Web service that returns the current status of a given aircraft assembly project. The Web service is enhanced to accept a more detailed query and to return a more detailed response.

- **Result:** Service contract changes.

- **Impact:** As above, systems bound to the old version of the service suddenly find their message formats are invalid. There were some systems that relied on the old version of the service that were not even known to the Web Service owner. A developer in another group had discovered the availability of the service over a water cooler conversation and had built a system that relied on the service, albeit infrequently. Discovery and quick utilization of a service that enables the quick creation of a business-enhancing application is a big benefit of the SOA approach. That system is a time bomb waiting to detonate the next time it tries to access the service through the old contract.

- **Change:** A financial services organization acquires a regional commercial bank. The newly acquired bank has a collection of systems that must leverage an existing loan yield calculation Web Service engine for consistency across the organization and to meet certain regulatory reporting requirements. The systems at the new bank are connected to the existing yield calculation Web Service.

- **Result:** Service performance degrades on the yield Web service and eventually the application is overloaded to the point of failure.

- **Impact:** Similar to the scenario covered in the section on unexpected change, applications that rely on the yield calculation engine begin to unexpectedly fail across the bank. Many of these applications rely on the service indirectly through other Web Services that build their capabilities on the back of the yield Web service. Diagnosing the root cause is a nightmare.

In order to effectively deal with this problem, there must be a facility to:

- Understand the makeup of the service network - the true dependencies between systems, even those that utilize a service infrequently.
- Predict the impact of a change - will this new application overload the Web Services to which it will connect? If I move an application providing services, which applications must be updated to interact with the new service?
- Insulate the downstream applications from negative impact while still allowing the change. It is not a solution, for example, to gate the enhancement of a service or the replacement of an application on the re-binding of dozens of applications to a new service location. (While this would not be an issue if organizations were to adopt and require the use of service registries like UDDI, the reality is that universal adherence to this policy, even in a single enterprise, will never be a practical assumption.) Neither is it a solution to re-bind the applications to a new service contract where the message schemas change. (This is not addressed by a registry solution alone.)

Planned Simultaneous Change to Many Networked Services

In some cases, change must be applied to the entire network of Web Services (or some large subset thereof.) Examples include:

- **Change:** A travel reservation network provider has decided to begin billing for the use of its Web Services - both internally, to facilitate cost accounting and externally, to generate additional revenue. There are many systems that provide services across the network and billing information must be collected at each one.
- **Result:** Every system that provides a Web Service for which billing will be initiated must be enhanced to report service utilization to a billing system.
- **Impact:** Some systems, like packaged applications, can't be easily modified. Custom application modification is expensive and time consuming. And even if possible, the incredibly costly duplication of effort across all these systems is

wasteful and inefficient.

- **Change:** A commercial bank has added a new set of affiliates to its partner network. These affiliates will integrate some of their loan processing systems with those of the commercial bank. The bank makes this possible through a collection of Web Services that are provided by a number of different systems - some packaged applications, some custom systems and some processes that are managed by an EAI platform. In order to secure these services, access control is enforced based on credentials passed in the Web Service requests.
- **Result:** There are many systems that must be updated in order to ensure partner access is granted.
- **Impact:** The variety of systems providing Web Services means multiple security frameworks must be understood. Adding partner access to a packaged application is different than adding access to a custom application, which is different than adding access to the EAI-hosted process triggers. This is a costly exercise - both directly and in terms of the risk that is introduced as each system is touched and configured in its own unique way.

- **Change:** A central services IT group in a large farming equipment manufacturer has built a collection of Web Services they are encouraging other internal IT units to leverage. In agreeing to leverage these services, one business unit has asked for a guarantee on service quality - response time, throughput and uptime.
- **Result:** The central IT group has a number of systems providing services. They must be updated to track performance statistics and to raise alerts when service levels dip below expectations (or, better, when they approach unacceptable levels.)
- **Impact:** As above, multiple heterogeneous systems must be modified, some of which are difficult to adjust at best.

While adding new applications to the network or modifying the behavior of individual application modules is made much easier through the SOA approach, it can be more difficult to make changes where global policy is involved. This is a direct result of the nature of an enterprise service network: There are more systems

to touch and those systems tend to be from a mixed set of vendors. Some are packaged and some custom built. And some services are provided by applications not even under the control of the organization that wants to apply policy to their usage. In addition to the direct costs of mass change, the risk of breaking a system in the process of enhancing it can lead to the types of cascading network impact costs discussed in the previous two sections.

In order to effectively deal with this problem, there must be a facility to:

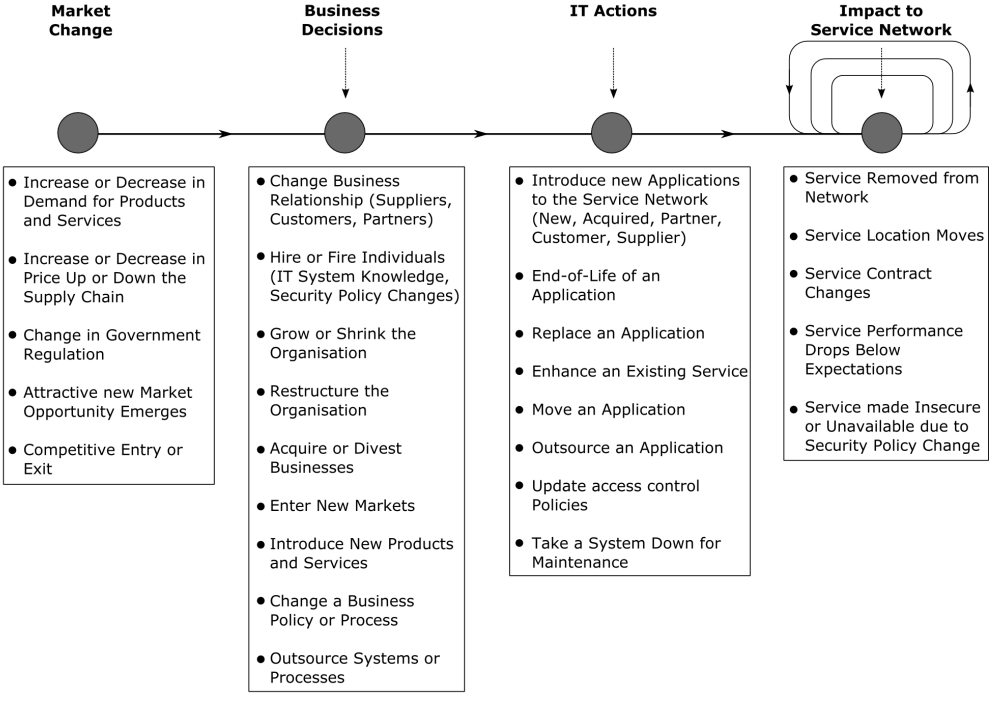
- Understand the true makeup of the service network - to identify where new behavior is required;
- Formulate the appropriate behavior - like permitting or denying access to services based on the sender's identity or sending billing information based on service utilization;
- Distribute policy, rules and behaviors into the network of Web Services in a consistent and automated way.

The Ultimate Impact of Change = Skyrocketing Costs

Encapsulating the discussion to this point, Figure 18.13, “Sources of change in an enterprise service network” [185] provides a comprehensive model for considering the types of change that can impact an enterprise service network, starting with changing market conditions leading to the impact to the service network and the ripple effects it can bring.

As we have seen, change can be planned and may start all the way over on the left hand side - a decrease in demand for a product results in a business decision to lower prices and promote a product which leads to IT implementing a new application (assembled using existing Web Services) which can lead to a drop in service performance resulting in ripple effects and cost. In other cases, change is sudden and unexpected and can come in from the top - an outsourced Web Service providing market quotes for securities fails. This is an immediate and unexpected impact to the service network coming in from the top on the right hand side. Ripple effects follow and costs accumulate.

Figure 18.13. Sources of change in an enterprise service network



Whereas the service-oriented approach to application architecture was meant to accelerate the responsiveness of the IT organization, the impact of changes to the service network and the ripple effects that ensue can quickly lead to precisely the opposite result. Change is easy and affordable, but the impact of change (unexpected or planned) can be unmanageable and expensive as shown in Figure 18.14, “Running and evolving an unmanaged enterprise service network” [186]. The ease and cost-efficiency of change in an unmanaged service network is an illusion, at best.

Figure 18.14. Running and evolving an unmanaged enterprise service network

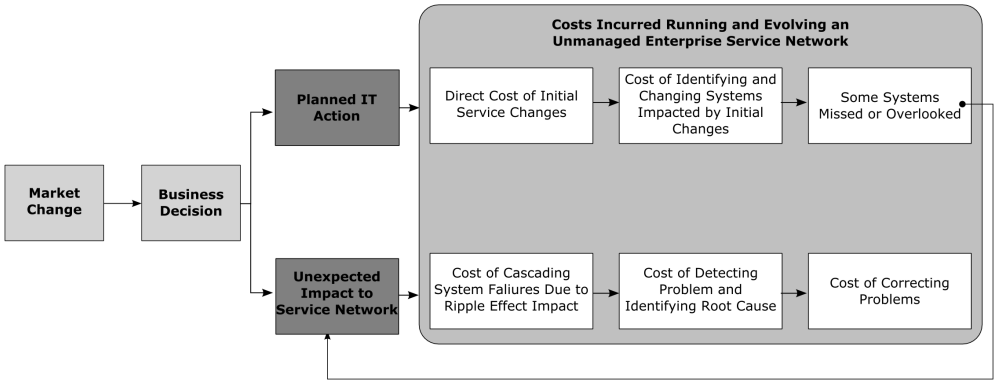
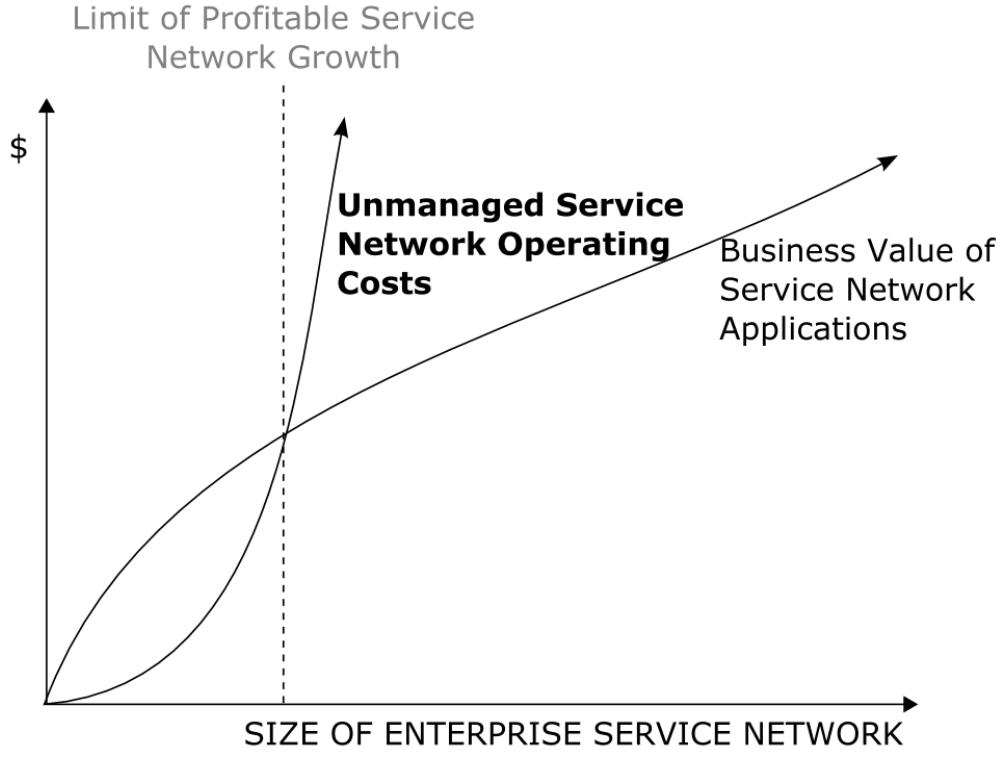


Figure 18.15, “Skyrocketing costs and complexity of an unmanaged service network” [187] illustrates the net economic result of these skyrocketing costs - an artificial limit on the ability to profitably grow the enterprise service network.

Figure 18.15. Skyrocketing costs and complexity of an unmanaged service network



Rather than freeing IT to be more responsive to changing business needs, the SOA approach can quickly lead to a reduction in responsiveness and increased costs.

In order to truly reap the benefits of the service-oriented approach to application architecture, organizations must have a way to manage the impact of change in their enterprise service network.

Web Services Management Platform Requirements

A Web Services management platform is critical to the continued profitable growth of a service network. To do the job a Web Services management platform must address the three types of change in an enterprise service network that dominate rising costs: unexpected change to a service in the network, planned change to a service in the network and planned simultaneous change to many services in the network.

The prior analysis yielded several key capabilities required to achieve this objective:

- Know when there is a service network problem - that the network of services is deviating from its normal operating range;
- Identify the root cause of a problem - if it is a service network problem, then which application, service and operation is at the head of the cascading chain of trouble;
- Insulate downstream applications and services from service changes and failures, or from an intentional change that would cause trouble if it were not isolated;
- Understand and visualize the makeup of the extended service network - identifying the complete set of the direct and indirect dependencies between systems;
- Predict the impact of a change - will this new application overload the Web Services to which it will connect? If I move an application providing services, which other applications and services will be impacted;
- Formulate a response to a problem or a plan for non-disruptive intended change to a service or services - a set of rules that if followed in the service network, will eliminate or prevent failure conditions;
- Distribute policy, rules and behaviors into the network of Web Services in a consistent and automated way.

In addition to these capabilities, a solution must scale and perform in a complex

enterprise IT environment, requiring the following characteristics:

- **Active** - The solution must actively participate in shaping the behavior of the service network - redirecting traffic, changing in-flight documents and enforcing policy. Merely observing and reporting on the operation of the service network lengthens the time and expense of reacting to trouble and does little to expedite cost-efficient intended change;

- **High Performance** - Solutions that can actively manage a service network must be able to efficiently manipulate in-flight XML documents. In environments where message volumes are high or in which responsiveness is desired, this processing must be extremely efficient. Algorithms for gathering, consolidating and transmitting statistical information must also be efficient. Efficiency results in low latency, high throughput and fast service response times. Horizontal and vertical solution scalability must be demonstrable;

- **High Availability** - A solution must not introduce a single point of failure into the service network. Like the service network itself, a solution must be distributed by design. In addition, each individual component of the overall solution must be designed for maximum uptime and rapid failure recovery;

- **Vendor Neutral** - Every enterprise-computing environment is heterogeneous. The adoption of a service-oriented application approach heightens that heterogeneity. Application components can be mixed and matched. Packaged applications mix with custom applications. Services are often extended for use across the enterprise and across enterprise boundaries where the application underlying a service may not even be known. A solution cannot limit the types of systems that can be actively managed in an enterprise service network. A good solution will enhance an organization's flexibility in that regard;

- **Non-Intrusive** - A solution must not require that systems participating in the service network be aware of its presence. Any solution that requires service providers or consumers to knowingly participate in the solution effectively creates a proprietary network of services. Requiring coding to a special toolkit or framework in creating consumers and providers substantially limits the value and scope of the solution. For services delivered from or to other organizations, changing those systems may not even be possible. Management platform components must slip unnoticed into the service network;

- **Deployment Flexibility** - Solution components must be deployable on the widest variety of platforms and in the location most appropriate for the situation.

In some cases, deploying on a service provider or consumer is appropriate and desirable. In other cases, deploying in the network between service providers and consumers is appropriate and can provide enhanced functionality. In some cases (for example, where managed service providers or consumers are not under the control of the entity seeking manageability) deploying between systems is the only practical option.

Architecting a Solution

Figure 18.16, “Service brokers, Web Service providers and a centrally managed policy” [191] below, provides an architectural overview of a solution that meets the requirements highlighted above, while Figure 18.17, “Alternate deployment option placing Active Agent on Web Service provider” [192] shows an alternate deployment scenario.

Figure 18.16. Service brokers, Web Service providers and a centrally managed policy

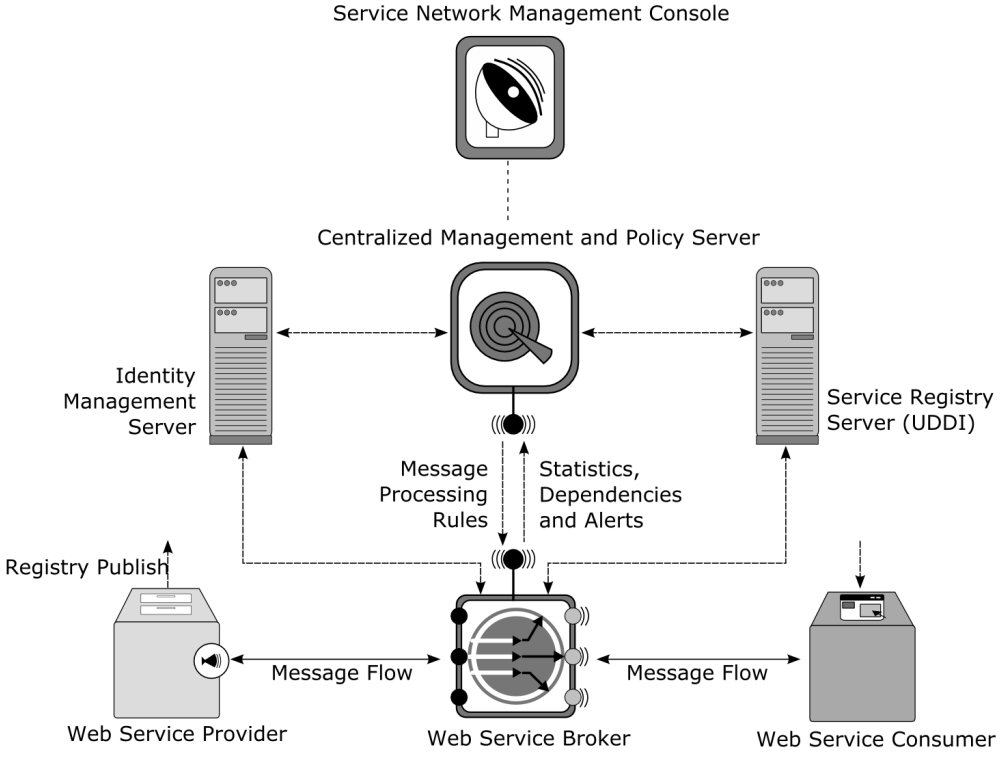
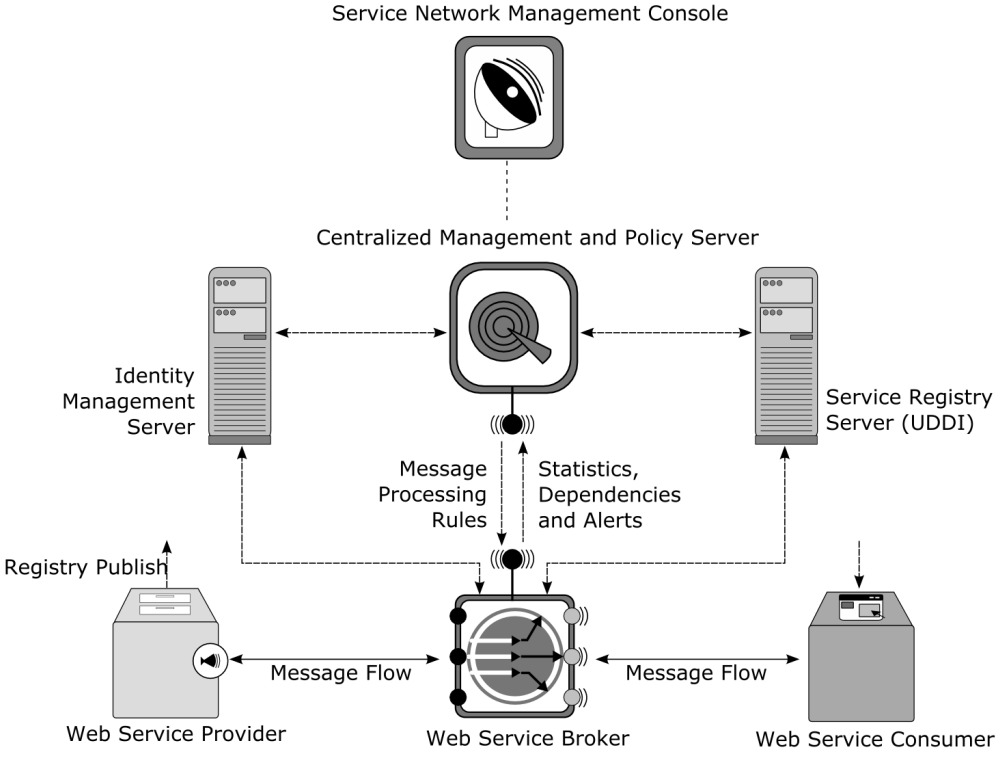


Figure 18.17. Alternate deployment option placing Active Agent on Web Service provider



This architecture combines the power of centralized visibility and policy management with distributed monitoring and policy enforcement. In this approach, active components live in the enterprise service network, logically positioned between the providers and consumers of Web services. These are called "in-network" components. A centralized management and policy server, working in concert with service registries and identity management solutions communicates with these in-network components both receiving and sending information. Information received from in-network components includes service performance statistics, alerts and service interdependency information. Policy, in the form of rule sets, is sent to the in-network components. These rules define in-network component behavior - when to raise alerts, how to process transiting message traffic, where to route messages, how to enforce security policy.

The following table summarizes the functionality required of the Web Services management platform:

Central management and policy server	
Function	**Detail**
Interact with in-network components	- Gather statistics - Receive alerts and notifications - Gather service dependency information - Transmit policy sets to in-network components
Analyze, display and report	- Correlate information from multiple in-network components to build network-wide view of service interdependencies and performance - Display service network configuration, interdependencies and performance in support of policy formation - Create reports and audits summarizing network policy, service utilization and network-wide performance
Detect and respond to unexpected service network impact and support non-disruptive intentional change	- Alert when service network performance deviates from preset expectations or from operational norms - Support root cause analysis via dependency and service execution path drill-down, mapping and visualization - Support the creation of rule sets that, when distributed to in-network components for enforcement, will heal current service network failures, facilitate automated healing when future problems occur or prevent unintentional impact to the service network when intentional change is made to a service - Coordinate with other service network infrastructure components including service registries, security policy servers and grid capacity managers to facilitate proactive and reactive change management
In-network components	

Central management and policy server	
Function	**Detail**
Function	**Detail**
Interact with central policy and management server	- Receive policy sets (rules for acting on Web Services messages) - Gather and send statistics - Gather and send dependency information
Apply rule set-defined policy. Trigger rule actions based on context- and content-sensitive evaluation of arriving messages	- Create access points for a managed service with the ability to change the protocol, security model and message schema - Raise alerts - warnings and alarms - Enforce access control policy - Change message content - Change message destination - Execute user code
Facilitate local policy management and reporting	- Create local rule sets - Log, alert and report on local service activity

Service Deployment Action

Figure 18.18, “Service brokers and active agents deployed in support of initial project” [195] through Figure 18.20, “Successive project deployments” [197] show the deployment of in-network components across successive Web Services projects in the enterprise.

In Figure 18.18, “Service brokers and active agents deployed in support of initial project” [195] the initial inventory project is now depicted with Web Services under management. A service broker is used to manage the services provided by the packaged CRM solution and those created for inventory management. The customer service workstation application consumes these services by delivering messages to a URI representing a message receiver on the front end of the broker. An Active Agent deployed directly on the application server is managing the order management service. In this case, the CSR workstation application sends service requests (messages) directly to the application server rather than through a broker. The use of an agent provides certain advantages such as increased performance and more transparency, but offers less functionality than a broker and introduces an invasive footprint on the server. For the tax and shipping services being provided by an outside organization a broker is the only feasible in-network component choice,

since there is no access to the systems providing the services. Even if access were possible, it is unlikely permission would be granted to modify them by installing new software.

Once under management, reuse of services to create new applications and application components means reuse of the management capabilities now built in to the service network. Where new services are created, they are brought under management. For example, in Figure 18.19, “New application components link to existing services” [196], a service broker deployed in the "DMZ" is installed to manage the new services providing integration with customer procurement systems.

Figure 18.18. Service brokers and active agents deployed in support of initial project

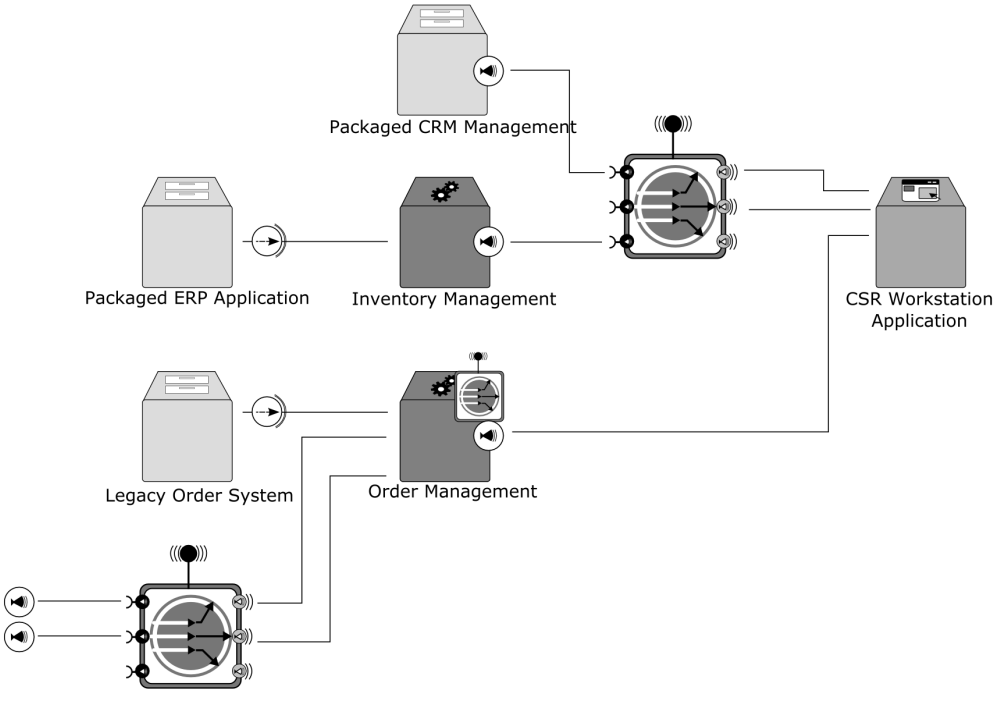
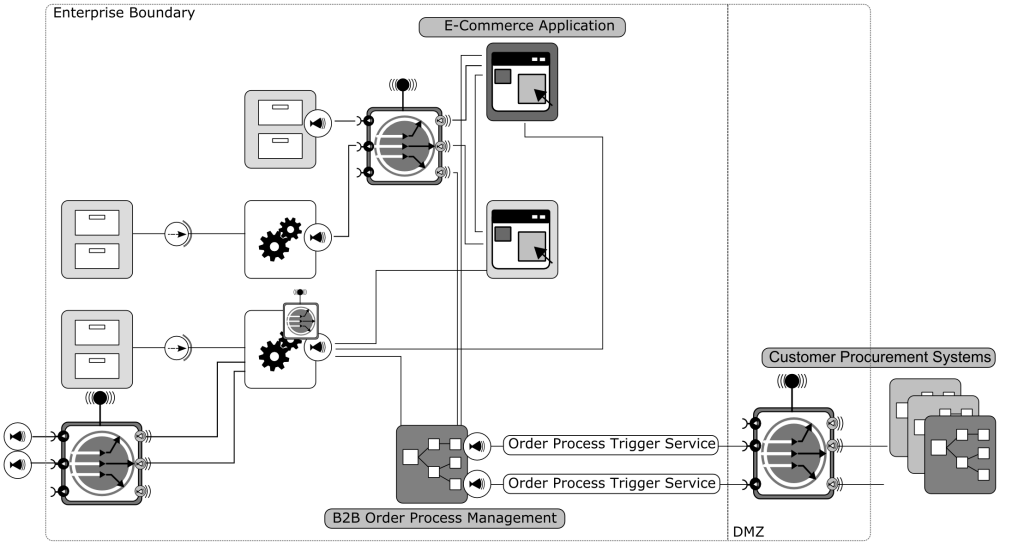
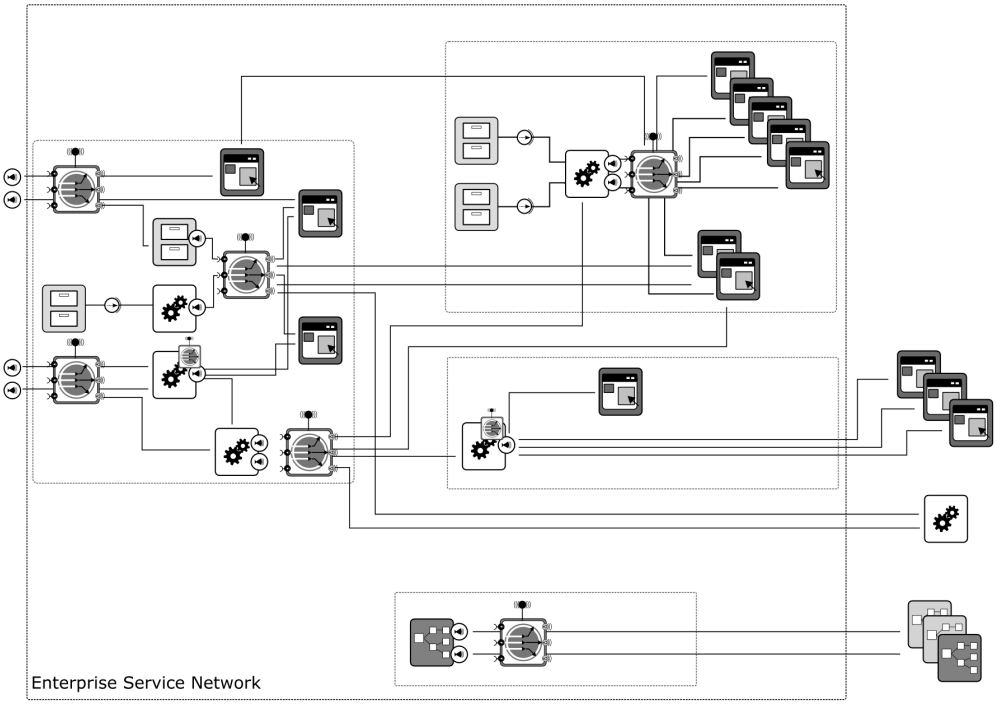


Figure 18.19. New application components link to existing services



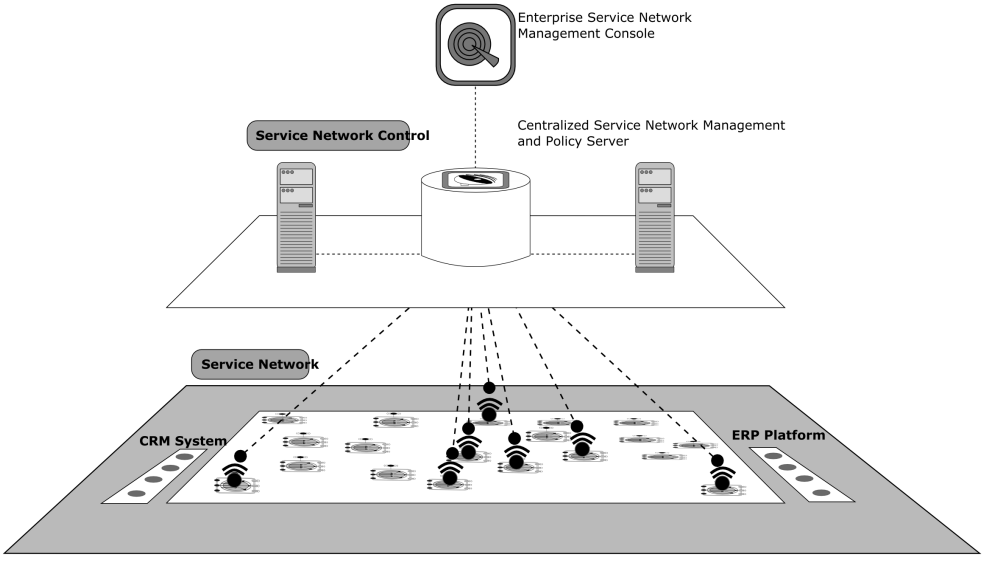
This process continues for each new Web Service project. Figure 18.20, “Successive project deployments” [197] shows the result of successive deployments as the service network grows. The result is a fabric of control woven into the enterprise service network.

Figure 18.20. Successive project deployments



The central policy and management server taps in to this in-network fabric of control as previously described and as shown in Figure 18.21, “The effective Web Services Management Platform” [198].

Figure 18.21. The effective Web Services Management Platform

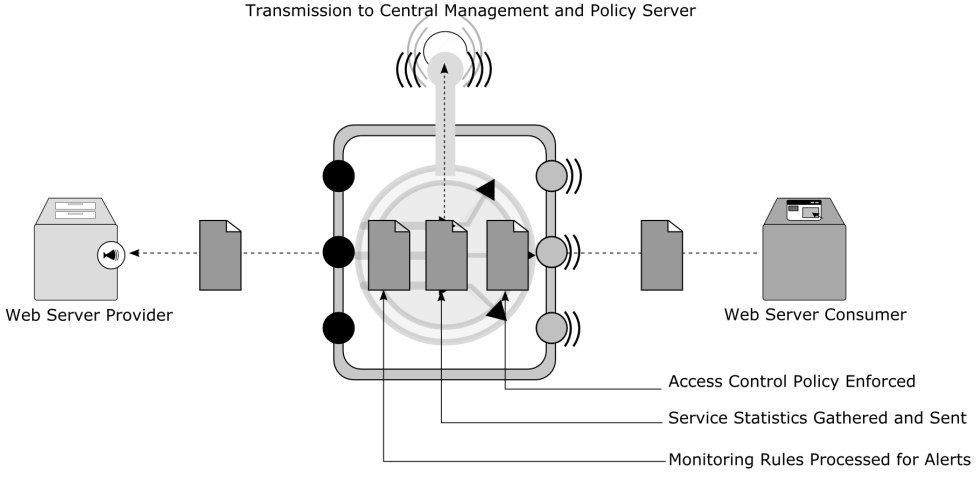


This solution provides a complete answer to the challenges faced in dealing with change in an enterprise service network - both unexpected and planned.

The Solution in Action

Figure 18.22, “Service broker intermediates provider and consumer interactions” [199] shows an example of brokered interaction between a service consumer and service provider.

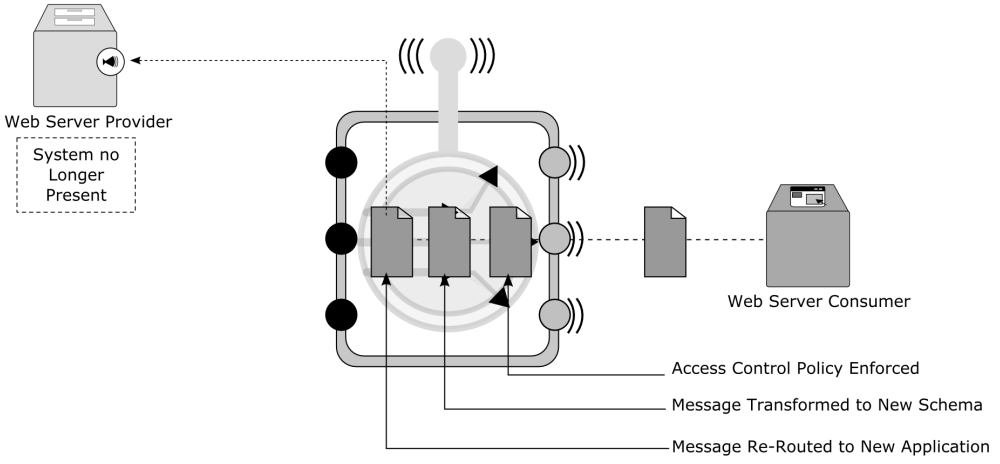
Figure 18.22. Service broker intermediates provider and consumer interactions



In this example, the service consumer transmits a message to a service broker URI (embedded in the service WSDL) representing the access point for the managed service. When the message arrives, the broker processes the message, applying the rules applicable to messages arriving on this access point. Here the set of actions taken includes: applying access control rules; gathering and sending statistics to the central management and policy server; and evaluating whether an alert should be triggered based on inspection of message content, processing context or performance statistics. For example, there may be a service level agreement in place guaranteeing service throughput, response time and uptime. Among many other things, the monitoring rules could compare current and cumulative service execution statistics and service availability, raising an alert if service performance is outside of SLA bounds. The message, once processed, is forwarded to the service provider, potentially over a different protocol and with a different security model and

credentials. This example showed a fairly "straight through" message processing configuration. The example in the figure below shows the ability of the broker to support the non-disruptive replacement of the application providing the service.

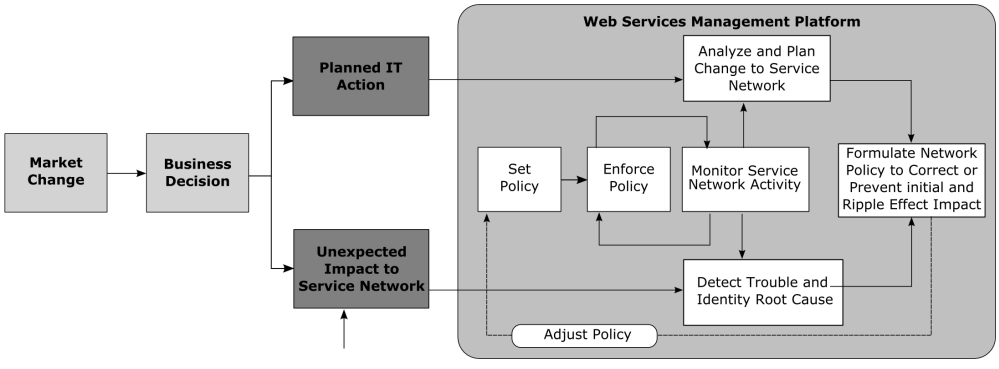
Figure 18.23. Non-disruptive application or service replacement



In this example, the scene has changed. The original service provider has been replaced with a new application providing a similar service but with a changed message schema. To facilitate non-disruptive change, the processing rules for messages arriving on this access point have been changed. In addition to the old processing rules (some removed from the picture for brevity) new rules transform the inbound message from the old schema to the new format recognized by the new application. The message is then routed to the new application.

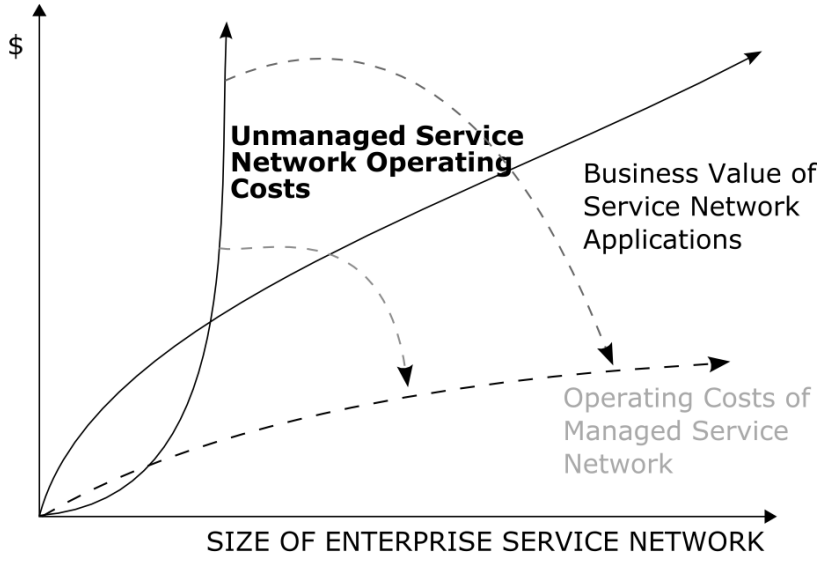
Even if there are consumers statically bound to the old access point and even if they are extremely infrequent users of the service, they will continue to work as expected. To facilitate identifying and updating these consumers of the old service version, alerts can be configured and reports run detailing service utilization. Without requiring a bit of change on the part of any service consumer (some of which may not even be under the control of the organization delivering the service) the service has been moved and the service contract changed without disrupting service network operation - no failed services, no downed applications and no ensuing costly ripple effects.

Figure 18.24. Web Services management platform automates reactions



The figure above shows the impact of change in a service network with a Web Services management platform in place. In contrast to the situation in Figure 18.14, “Running and evolving an unmanaged enterprise service network” [186], the managed enterprise service network automates the ability to respond to and embrace the constant change (planned or unexpected) that characterizes the enterprise service network - without the skyrocketing costs previously accrued. The net result is a flattening of the cost curve representing the ongoing operating costs of an enterprise service network.

Figure 18.25. Web Services Management Platform lowers the cost curve



Those enterprises that deploy a Web Services management platform in support of their adoption of a service-oriented enterprise software architecture will get what they expect - movement toward operation as a real-time enterprise and an IT organization able to rapidly respond to and even enable changes in the operation of the business.

SOA Implemented

We have come a long way in our understanding of SOA as we have discussed the various concepts, technologies and influences that make up an effective SOA. In planning a migration to an enterprise ready SOA we have used this understanding to discuss the implementation of a SOA, noting such things as planning, tracing the impact of various changes and dealing with specific situations.

.In all of this discussion we have not dealt much with what the SYSPRO system

offers you. In the next chapters and sections of this book we will focus a lot more on the role of SYSPRO as a core SOA platform and as an extender of the capabilities of your enterprise.

This page intentionally left blank

Part V. Applications

Having examined Service Oriented Architecture and some systems of implementation and management, we now look briefly at SYSPRO applications in relation to SOA. You will find a product description, as well as discussion on the extended enterprise, intra-enterprise relationships and inter-enterprise relationships.

By the end of this part of the book, you will understand more about SYSPRO and what we offer, as well as how SYSPRO has worked to extend the areas of functionality, system boundaries, application development, integration, electronic commerce and collaboration.

Chapter 19. Introduction

Any theory remains just a theory until it is applied to a particular situation. We have learned a lot of theory about SOA, what it is and how to go about implementing it. Using this information as our basis, we will now examine some applications of the theory. SYSPROs e.net Solutions in conjunction with the SYSPRO core system provides a platform to show the theory in action.

As you read these chapters, please feel free to refer back to the preceding sections to clarify your own understanding on those details with which you may not be familiar.

This page intentionally left blank

Chapter 20. SYSPRO e.net solutions and the Extended Enterprise

As you have read in the early chapters of this book, SYSPRO has been in business for many years and has built a versatile product to meet the demands of the enterprise market. The evolving of the product into SYSPRO e.net solutions is continuing, allowing greater flexibility and agility for you, the end user.

Product Description

The functional footprint of SYSPRO e.net solutions encompasses primary "bolt-on" functions, such as Customer Relationship Management (CRM,) Supply Chain Management (SCM) and domain-unique functions such as "Utility Customer Billing." Yet it also expands to include applications unique to specific domains or "groups of industries" and specific industries, such as "Consumer Packaged Goods."

The business modules available from within SYSPRO e.net solutions include such functions as:

- Accounts Payable
- Accounts Receivable
- General Ledger
- Cash Book
- Electronic Fund Transfer
- Assets Register
- Contact Management
- Inventory
- Purchase Orders
- Business Analytics
- Sales Analysis
- Sales Orders
- Bill of Materials
- Quotations
- Work in Progress
- Requirements Planning
- Lot Traceability

- Report Writer
- and others....

These modules represent the core ERP functionality of SYSPRO e.net solutions. Around this core, or extending the core, are the business objects - components that can be assembled together to create customized applications . Each business object abstracts the functionality or business logic of multiple SYSPRO programs into a single executable piece of code, a 'service.' This code serves as a simplified interface to the core system. The business objects can be used to create specific customization for the core system and to create other applications using the SYSPRO platform. There are currently just over 200 business objects, compared to the 1 200 programs of which they are an abstract.

Extending the Enterprise

As we have shown throughout this book, the move toward a Service Oriented Architecture is providing businesses with a platform for greater functionality and better integration. This is the goal of SYSPRO e.net solutions and the incorporated extensions of the original ERP system.

Extending Functionality

We have seen that accelerating technological changes and the continued changes of the global economy are making it necessary for businesses to think and perform strategically in order to remain competitive and profitable. In order to respond to these and other changes, companies are being forced to expand their focus beyond their enterprise and gain greater insight into their customers and their supply chains. These changes are extending ERP system functionality and driving the move to higher collaboration between businesses enabled by the newer technologies.

By extending its ERP functionality, SYSPRO software creates an extended enterprise system. The extended enterprise includes the core ERP functionality and builds on it to include Customer Relationship Management (CRM,) Advanced Planning and Scheduling (APS,) Business Analytics (BA) and e-Commerce. This software helps manufacturers and distributors gain a 360 degree, real-time view of operations. It also promotes cost savings, faster inventory turns and reduced lead times while helping maximize profits and enhance the customer experience.

Extending Boundaries

The extended enterprise does more than just include extended functionality. It also extends ranges and types of communications between people and systems. This extends the boundaries of communication both within an enterprise and outside of it.

The SYSPRO document flow manager and SYSPRO EDI (Electronic Data Interchange) extend the boundaries of communication by making it easier to take information from one sector and provide it to another in the format that is needed.

Custom Application Development

The SYSPRO modules listed above have many customizable options. That customization, in light of the modular and component approach of the system, allows a range of custom application development options. You are able to match the application to your specific business needs.

The advantage of this approach is that you only need to install and maintain what actually applies to your business, while retaining the option of adding further functionality at a later stage. As your business grows and becomes part of a larger collaborative system, you will be able to make use of more and more of the services and business logic integrated through out the SYSPRO system.

Enterprise Application Integration

SYSPRO software optimizes your control of your enterprise application by providing integration between the areas of the extended enterprise through the various modules and business objects. As we looked at the current market trends and particularly at Service Oriented Architectures, Demand Driven Supply Networks, Chapter 12, *Demand Driven Supply Networks* [117] and Collaborative Supply Chain Networks Chapter 13, *Collaborative Planning, Forecasting and Replenishment* [123], we found that the ability to integrate applications will give increased functionality to the enterprise. In addition to the integration between modules within the SYSPRO software arena there is also the core integration with other operating environments, including Microsoft® Windows, Microsoft Office, Microsoft SQL Server, UNIX and LINUX.

By utilizing a multi-platform software product, you gain scalability and flexibility. You are able to select only those functions needed to increase your operational control and efficiency. The SYSPRO Document Flow Manager and SYSPRO EDI

(Electronic Data Interchange) facilitate transaction and communication between different systems and increase application integration within the enterprise.

Electronic Commerce

One of the essential results of modern business trends and business practice is the rise of electronic commerce. This not only includes the use of the Internet to market and sell products, but also the use of the Internet to aid communicate between businesses, provide business services, and increase real-time operations. Electronic commerce not only includes the Internet, but also the networks within each business and between businesses.

SYSPRO's comprehensive distribution capabilities include Internet buy- and sell-side solutions. These provide you with access to information, order visibility, online self-services and greater communication possibilities. They also allow you to take advantage of Internet transmission of purchase orders, production schedules and electronic procurement using supplier catalogs.

Collaborative Commerce

The era of doing business for oneself by oneself has gone. Businesses that are part of collaborative networks will have a significant advantage over those that are not. That advantage will allow them to thrive and grow, while those not involved in collaborative commerce will one day wake up to find that they are too far behind to catch up.

Collaborative commerce is not just where the future of business enterprises lies, it the current backbone of enterprises that are adapting to the demands of the global economy and the business trends that modern technology has initiated.

Collaborative Private Exchanges

In some instances two separate businesses are so closely linked that they would benefit enormously from a collaborative private exchange. A manufacturing business that has outsourced its distribution to a single company will find great value in collaborating in this way. The same data is used by both businesses and enabling a direct private sharing of that data would speed real-time exchanges and greatly improve efficiency.

Chapter 21. Intra-Enterprise

Within every enterprise there are applications that function only within the internal network structure of the business, sometimes called the 'back-office' functions of the business. Applications within this classification will usually have higher security considerations and may only be available to a limited set of users.

The core function of the SYSPRO e.net Solution functions on the intra-enterprise level enabling the sharing of information and collaboration of business functions within the enterprise system.

From a SOA perspective, the intra-enterprise level of operation is limited. Internal services are useful within the business, but they need to be made available to a wider audience to fully gain the benefits associated with SOA. Intra-enterprise operations do, however, form the core data functions that enables collaboration and real-time leverage within a SOA.

While intra-enterprise functionality is vital to an enterprise, it is the inter-enterprise functionality of a SOA that extends the enterprise and creates the environment of collaboration and business networking advantages

This page intentionally left blank

Chapter 22. Inter-Enterprise

SYSPRO e.net solutions provides the enterprise user with multiple options for access to and sharing of key components on a wider inter-enterprise system. These are the modules that are enabled as services, allowing collaboration and information sharing with other businesses.

There are two areas of inter-enterprise usage:

- Access to outside or third party information;
- The sharing of information with outside parties.

It is up to the individual enterprise to control the level of open or closed access between systems, but some modules (like those used in a supply chain network) must have a basic level of inter-enterprise functionality enabled in order to work. The sharing of information through shared or integrated data repositories brings the full functionality of SOA to the enterprise.

For a more indepth view of the specific SYSPRO modules or business object capabilities please contact your local SYSPRO representative and spend some time interacting with them.

The final portion of this book deals with specific situations and companies that have installed SYSPRO and the results or changes that they have experienced. Please use your knowledge of SOA to evaluate for yourself the difference that SYSPRO has made to these enterprises.

This page intentionally left blank

Part VI. SYSPRO - In Action

Five real life installations of SYSPRO are presented to demonstrate the SYSPRO system as the core of a Service Oriented Architecture in these different enterprises.

By the end of this section, you will have observed how SYSPRO enables or creates a SOA . You will also see what challenges and benefits other businesses have experienced as a result of implementing SYSPRO as their core systems. The experience of real life companies will allow you to understand more of the options that are available for your enterprise. For instance, you will see increased flexibility and agility, better interoperability, data rationalization and real time processing, all benefits from implementing a SOA.

Chapter 23. Introduction

When evaluating any new product, it is important to have some information of how the product has worked for someone else, particularly someone in a similar position. The same is true of a SOA. For this reason we have provided you with five different business experiences, five situations in which SYSPRO e.net solutions has been installed and studied to find what differences were noted.

As you read these experiences, keep in mind what you have learned about SOA so that you can evaluate the evidence presented for yourself. Feel free to flip back in this book and re-read sections if you feel the need.

All five business experiences are based on SYSPRO System 6, particularly e.net solutions. Because SYSPRO's system is in modular format, not all the modules have been used in each enterprise system This is positive. Being able to choose just what your business needs is part of the SOA ideal.

This page intentionally left blank

Chapter 24. Dewhurst plc.

Founded in London in 1919, Dewhurst plc. has subsidiaries in the United Kingdom, Canada, USA and Australia. It manufactures push buttons and control systems for the lift, keypad and rail industries and has long been a pioneer of new manufacturing technology in this field. The company has a turnover of around 30 million pounds and employs over 300 people around the world.

The main business driver for Dewhurst's new ERP system was its strategy for continued development as a global manufacturing organization with satellite sales offices around the world. With an increasing percentage of the company spread across different countries, it wanted a system which would provide local sites with the tools to deliver real customer focus, enhance the accuracy of figures reported across the group, and improve visibility and understanding of the individual businesses.

This case study was originally researched and reported on by Ronan Martin-King.

The Situation

Dewhurst, as with many typical transnational companies, had an agility problem due to its use of disparate systems, and a lack of real time information about the business.

Various IT systems used throughout the group and at the main plant in Hounslow, were not fully integrated. Management and shop floor personnel alike were becoming increasingly frustrated with the system shortcomings. Site using separate software packages for manufacturing and accounting. The only way to keep up with the changing requirements of the business was through the batch processing of data. The main frustrations were with the lack of reporting capabilities, insufficient visibility and integration and the inability of the system to facilitate transactions in real time. However, disenchantment with the existing system meant Dewhurst didn't have to face the level of resistance and skepticism from staff that often goes with the introduction of new IT systems.

According to Jared Sinclair, financial director, "It wasn't as if we were replacing the IT system for the sake of it or forcing a change; people weren't happy and saw the new system as something that would make their lives easier."

SYSPRO e.net solution, as an extendable SOA platform, was able to give Dewhurst the agility and real-time information required, in addition to speeding up manufacturing and resource management.

Planning Stages

The company conducted a SWOT analysis of the existing system and used it as a specification for a new solution. They approached several suppliers, eventually choosing SYSPRO from K3 (one of SYSPRO's UK partners.) "Naturally, all the vendors were keen to claim their systems would meet all our needs, but we asked them to provide evidence by demonstrating the software against our specifications. We knew from the assessment we did of SYSPRO that the system would give us the visibility needed, and it would be much more user friendly than the others," said Sinclair.

Dewhurst installed the system at its LiftStore subsidiary in Flint, Wales, because of the site's immediate needs but then delayed the installation at Hounslow. Aware of the growing importance of lean manufacturing principles to the business, the company wanted to take a step back and get an objective view of its operation.

Sinclair: "We felt there was a danger of replicating our old processes and practices in the new system, ultimately resulting in the same frustrations. People are naturally constrained by their own experiences, and it's easy to reapply current practices without challenging them. We saw this as a golden opportunity to improve our processes and find what the business needed from the new IT system to support those processes. It's important to engage independent, expert help for that --somebody that can challenge current practices, encourage lean thinking and offer guidance."

Coincidentally, K3 had already been working with a specialist lean manufacturing consultancy firm (HOSCA Group) and was able to introduce them to Dewhurst. Using a detailed benchmarking and process mapping exercise, the consultants were able to provide Dewhurst with an objective and detached view of what could be achieved, as well as a blueprint and set of metrics against which the SYSPRO system should be implemented.

Sinclair: "The consultants encouraged our approach to thinking outside the box, always asking 'Is there a better way?' We knew the go-live might be delayed because of the extra discussion, but felt the time spent up front would benefit us later - we wanted to establish as lean a process as possible."

In fact the system did go live on time and on budget - and as a result of the pre-implementation work, there was minimal additional consultancy required afterward.

Paul Williams, business development director at K3: "Our standard methodology is to review current business processes, then blueprint a set of desired processes, applying lean manufacturing principles. Then we configure and demonstrate the software supporting those desired processes. That's particularly important in a group of multiple businesses where there's an objective to standardize best practice across the organization."

Sinclair is quick to note Dewhurst wasn't prepared to force the system on its subsidiary companies. Having assessed the capabilities of the system from a key skills, key employee and data transfer perspective, SYSPRO was the preferred solution. Dewhurst, however, would only implement it group-wide, if it made business sense.

"Each subsidiary had the opportunity to voice its concerns about whether the system, as specified, would meet its specific needs. We would then consider whether the subsidiary should have some other SYSPRO modules or a different system altogether." states Sinclair.

A near tragic incident served to highlight the benefits of having a group-wide system. Prior to the SYSPRO system going live, the accountant in the Canadian company was involved in a serious accident while on holiday.

Sinclair again: "It's not a very pleasant story, but it is a prime example. Had the Canadian company been on SYSPRO at the time, we could have said to our finance manager in the UK, 'You know the system, please make sure it continues to run smoothly.' We could have assisted with the day-to-day processes to ensure business continuity until they found cover. As it happened, SYSPRO wasn't live, and we had no real in-depth knowledge of their systems or processes, so they had to deal with the situation on their own."

What a Difference!

Dewhurst believes the success of the SYSPRO system and lean manufacturing has largely come about as a result of a change in culture by involving all staff in both the system implementation and the introduction of lean practices. The lean work was initially focused in one particular area - the 'pressels' shop. It served as a proving ground by demonstrating to the shop floor staff that what they thought impossible

could actually be achieved. (A pressel is the face of the push button and the company makes thousands of variants.) Lean manufacturing then moved downstream to the sheet metal area.

On the basis that customers ordered complete push button assemblies rather than the individual pressels, Dewhurst changed the set up to be driven by customer button orders rather than MRP Pressel orders. As a result, a stock room previously full of manufactured pressels now lies empty. The principles and practices are now rolling out across the Hounslow site and throughout the rest of the company.

The real benefits of the new ERP system and lean program have been realized in dramatically improved levels of customer satisfaction. During the previous two years, the company had grappled with the problem of overdue orders, but now overdue orders have been eliminated completely - and it's not at the expense of lead times, which for pressels have also fallen by 78%. Dewhurst is now using its own internal resources rather than the external consultants to spread lean practices throughout the organization. The consultants trained two of the company's own employees as 'improvement champions,' and the head of operations, Alan Orr, (himself from a lean consultancy background) is leading the roll out of lean practices both within the Hounslow site and across the group. (He has just returned from Canada having introduced a new factory layout there.)

Orr says that most satisfying aspect of the program has been the cultural change in the business: "It's the soft skills that are important. It's easy for people to become defensive when you're introducing change like this, but we've seen a big shift in attitudes - that's key to project success."

Sinclair adds: "While we were all aware of the benefits of investing in people in this way, it isn't easy to quantify the returns in advance, and because there are consultants out there that don't deliver, it's easy to be skeptical. "Traditionally," he emphasizes, "manufacturers have seen investment in capital equipment as the means of improving lead times. In fact that's the analogy and justification we used - we asked ourselves if we could invest 100,000 pounds in a machine to deliver these sorts of reductions in lead times, would we? The answer: We would have bought two."

Actually, Dewhurst's investment in capital equipment has also fallen significantly (from 200K in 2004 at Hounslow to just 66,000 this year,) and Sinclair believes much of that is as a direct result of the focus on improving through people, reinforcing the justification for the project. However, Orr says the reduction in lead times means some investment in capital equipment is inevitable: "Because customers know our lead times are much shorter, they are ordering later, more often and in

smaller batches. Because the runs are much shorter, one of the biggest consumers of time is changeovers. Reducing set- up times through single minute exchange of die (SMEDD) techniques, only contributes so much, so we are buying some second-hand machines, which have less capacity but which we'll be able to leave set up; that will help us drive down lead times even further."

The SYSPRO system is now live across all but one of the group's five sites and Sinclair expects the roll out to be complete by the end of the year. Sinclair has been very pleased with the results and the system will also be extended to include a newly acquired Australian company.

As the group's financial director, Sinclair has benefited directly from the system as has the rest of the management team. Management accounts used to be reported within ten to fifteen working days after month-end. With SYSPRO it is now possible to forecast profits on day one of the new month and report actual figures by day four. It is also possible to look into the system live at any time to see how the business is performing for the current month. "As it's all in real time, we can get an up-to-date view of the current situation and respond immediately," he says.

Initially, the data was set up on each subsidiary's individual system with group reports compiled manually. Now the company is using SYSPRO to help consolidate reporting across the group. "Local resilience was the priority; now we're planning the best way of obtaining the group data. We have decided on how to construct the general ledger and we're currently working through the options for sales orders, invoicing, letters and so on. We are determining what sort of information we want to pull down overnight and the bandwidth we need."

Sinclair anticipates the consolidation will allow him to cut the time it takes to report to the stock exchange after the half-year and year-end by at least 30%. He also believes it will bring big benefits in inventory management and group purchasing.

Dewhurst is also working on enhancing its electronic trading using SYSPRO's e.net technology and document Flow Manager to write the interface to customers' and suppliers' own systems. SYSPRO e.net solution (a Microsoft-based architecture) allows Dewhurst's IT department to write applications for SYSPRO and link to other best of breed applications without affecting the core SYSPRO environment or damaging the upgrade path. "Even if our customers don't have SYSPRO, the SYSPRO e.net technology will enable them to trade with us in an efficient manner. That will take the system to the next level!"

There has been a major side effect of the lean ERP program Dewhurst has completely reorganized the layout of the Hounslow site, reducing the physical space

it needs for manufacturing by half. According to Alan Orr, "The company has been able to make more efficient use of the available space and we have plenty room to expand if we need to."

SOA Summary

As we have gone through this case study, we have seen many of the benefits of SOA exhibited. Dewhurst also experienced gains through better in house supply chain management and better information access and reporting (Business Analytics.) Their overdue orders were eliminated, and lead times on orders fell by 78%. The company cut the time taken to report to the stock exchange by at least 30%. With SYSPRO e.net solutions all the data is in real time, so management is able to get an up-to-date view of the current situation and respond to it immediately.

Dewhurst's experience with SYSPRO e.net solutions shows a successful migration to a SOA platform and the improvements that follow.

Chapter 25. Lakeshirts, Inc

Lakeshirts, a full-service screen print, embroidery and dye works company in Minnesota that produces promotional clothing products for resorts, golf and corporate entities, was founded 20 years ago by Mark Fritz and Mike Hutchinson in a home basement. Today the company maintains a library of more than 5000 stock designs that can be customized with the names of resort or golf clubs and applied to a variety of garments, including tee shirts, sweat shirts and golf shirts as well as caps. Lakeshirts, which has produced products for such prestigious events as the New York City Marathon, sells exclusively through a network of sales representatives.

Producing shirts for special events requires that the project be completed at exactly at the right moment and that the order be 100% correct. There is no room for any variance on deliveries, order precision and quality. However, these stringent requirements were not easily met until recently.

The Situation

Lakeshirts sales representatives had been using a custom computerized order entry solution which proved cumbersome and inefficient. The sales reps had the ability to enter orders onto their laptops while on the road, but had to wait for access to a fax to send the orders to Lakeshirts. Moreover, there was no immediate feedback on whether the items ordered were even available in the combinations, colors and sizes specified.

Order processing at Lakeshirts was equally awkward. The incoming orders were used to create an import file for the company's order entry system. A variety of databases were used to transmit work instructions and schedule production of the customized designs on the garments. Invoicing and inventory control were accomplished by a separate accounting software package, leaving many gaps and opportunities for errors in the chain from order entry through production and shipment to invoicing. In fact, as many as three days could elapse in getting an order to the production floor, necessitating the constant checking of orders by customer service to assure validity. The work of maintaining two systems was also costly and a drain on Lakeshirts' time and resources. "The system was unwieldy and sometimes we had to wait days for an order confirmation," admits salesman Kirk Lundmark.

Solution

The challenge, therefore, was for Lakeshirts to implement a system that leveraged the right technology to facilitate order entry as well as maximize internal efficiency. A primary goal was to enable Lakeshirts' sales representatives to order on the Web with the ability to select from a complete sourcebook displaying all available designs and garments, enter the order, print a copy and review orders previously entered (i.e., setup a Web Service system.)

The first step was the selection of new software. A review process, working from a consultant produced "wish list," eventually narrowed the choice to SYSPRO enterprise software. RT Enterprises, the local SYSPRO reseller, accomplished the software installation during the company's slow season to minimize disruption.

While SYSPRO has made numerous efficiencies to the company's accounting and production scheduling, it is the "value-add" which RT Enterprises "brought to the table" that is proving to be the winning combination that is enabling reps, like Kirk Lundmark, to accelerate sales levels and Lakeshirts to speed order turnaround time.

After a redesign of the Lakeshirts web site to clearly delineate garment and design choices, RT Enterprises employed SYSPRO e.net solutions, one of the first Microsoft .NET component-based architectures, to add customized functionality to the SYSPRO software without changing the software source code. RT established a new 24/7 Web order entry process, an on-line rules-based product configurator to ensure order accuracy and new, highly efficient order processing procedures.

Action Results

For the past six years, Kirk Lundmark and his wife have been sales representatives for Lakeshirts. However, it wasn't until the new Lakeshirts SYSPRO processes (which enable the two to place orders over the Web at any hour, ensure order accuracy and provide their customers with instant information on order status) that each saw their bookings climb to the million dollar range.

So, what sales tools does an independent sales rep need to move from being "good" to being a "Million Dollar Producer"? Sometimes the answer isn't always obvious. For Kirk Lundmark, the new SYSPRO ERP software implementation was the additional tool that he needed to speed order entry and ensure order accuracy. Ultimately, the efficiencies and access to information provided by the SOA system became "stepping stones" that helped him and his partner each reach

the lofty goal of membership in the “Million Dollar Sales Club.”

The New System

Through the use of SYSPRO e.net solutions COM (Component Object Model) objects, orders placed by sales reps over the Web are automatically transacted directly in the SYSPRO database. The Web application passes information relating to the order to the SYSPRO COM object using the XML standard. The SYSPRO COM object, in turn, processes the transaction in real time. Orders go automatically to the scheduler and a job ticket is produced for each design with the various sizes and colors of garments. The job ticket is automatically transmitted to the production department. This solution replaces the old time consuming and error-prone order-entry process, speeding order turnaround, particularly during the company’s busiest seasons. Also, when orders are placed over the Web outside of the company’s normal business hours, they are immediately and automatically processed without order entry personnel. The result is that the majority of orders now arrive in production the same day they are received in the sales office, correctly documented with customer specific information from the SYSPRO Customer Master File.

The Microsoft .NET-based Product Configurator enables Lakeshirts sales reps, such as Lundmark, to configure products over the Web. The Configurator ascertains that all order criteria are captured. After selecting the design and type of garment, the sales rep is automatically presented with a list of the valid placements and the available colors and sizes of the garments. Because the configurator is rules-based, the rep cannot make incompatible choices. Component dependencies are based on options and selections are dependent on other selections. Moreover, additional charges, such as bagging and tagging, can be easily selected from a list with the correct pricing automatically added to the order.

The solution also provides comprehensive reporting for the sales reps via the Web on a 24/7 basis, again saving customer service time and expense in generating and sending reports.

Lundmark admits there was an initial learning curve to get the new process rolling, but now he wouldn’t think of going back to the old way. “I love the new system. It is so user friendly,” he says.“The new process has given us what we lacked before – the ability to ensure order accuracy and speed order turnaround.” He is quick to attribute it to the recent entry of both him and wife into the “Million Dollar Sales Club.”

Lundmark says the new system not only speeds the entry of new orders, but it is instrumental in speeding the re-ordering process. “When a customer calls in for a reorder, it’s a snap to find the original order and duplicate it. If there are revisions to the orders, they are very easy to make thanks to our ability to create ‘name drop’ locations with the initial orders.” Re-order turnaround time has dropped from two weeks to less than a week, he says. In addition, he appreciates the ability to print out order confirmations replete with the required designs right on the confirmation.

Lundmark also says the new processes are enabling him to build business with smaller customers. “They can log onto the Web and look at the various designs. They also can see the status of their orders,” he says.

SYSPRO's e.net solutions enabled Lakeshirts to extend their ERP system to include real-time data access for all sales reps and customers. By enabling the core functions of thier system as Web services they were able to improve turnaround and order proficiency.

Today, Lakeshirts exemplifies the fact that the application of service oriented system can produce the efficiencies that a small business requires to compete successfully. As for Lundmark, he says he has one piece of advice for companies: “Implement Web ordering ASAP. It’s been fantastic for us.”

Chapter 26. Union Carriage and Wagon

Union Carriage and Wagon is a South African company that manufactures and upgrades locomotives and coaches for the railway industry. Since 1957, UCW has produced over 13 000 new locomotives and coaches and forming a very important part of the 'wheels of South Africa.' The company provides innovative rail solutions both for the South African market and the export market, including Africa and Asia. It also does significant work in refurbishment and upgrades of existing rail coaches and locomotives. Another part of their business includes offering the following services: conceptual studies; estimating; procurement; project management; and commissioning of rolling stock.

At the time of writing this book, UCW had just been selected as the manufacturer of choice for the Gautrain project, the proposed public transport system linking Pretoria, Johannesburg and the Johannesburg International Airport. The Gautrain project is one of the biggest railway projects in South Africa and is vital to the reduction of traffic congestion in Gauteng. It is also a vital part of the infrastructure of the province that will be used for the Soccer World Cup in 2010.

UCWs' Need

Union Carriage and Wagon (UCW) had specific requirements needed met by a SOA extended ERP system. A very important part of UCW's business involved the use of escalation conditions based on a commercial evaluation (profit, cash flow, and exchange rate) of a specific tender. There were twelve particular requirements they listed at the beginning of the project:

- Estimate multiple contracts for a project;
- Estimate projected costs and profits 5 years into the future;
- Consider escalation conditions at base line;
- Commercial evaluation of profit and cash flow forward;
- Project Management of VO (Variation Orders,) DC (Design Changes) and RMR

(Raw Materials Requirements.)

UCW is a project management company. Variations in orders could be raised by the client or by UCW, in which case there existed a need for client approval. Production variances may also occur in regard to the raw materials issued for production of a specific coach.

- Production planning and scheduling

  Production Planning and Scheduling is vital. When planning is done, the load on the factory is not visible. UCW plans contract by contract into the future, thus multiple contracts are layered over on the factory floor. Being mainly a jobbing house, UCW has the scenario where the constraint exists all the time. Thus after planning the move is to scheduling mode to facilitate all the different orders in the factory.

- Electronic progress tracking of production orders

  This was already in place in the form the 'Controlware' System.

- Real time project progress tracking

  There was a need for real-time project progress tracking with progress reports relating these jobs back to the overall layout of the project schedule: The Project Roadmap.

- Drawing versions and releases configuration management

  This is of the utmost importance. Originally, there had been manual delivery of the drawings to the factory. This would now be done electronically.

- Scope of Work configuration (Internal and External)

  A coach will come in for upgrading. This begins with a stripping inspection list. Some of the work is outsourced. The undercarriage of a train in called a bogey. Some bogeys, for example, might be sent to an external supplier who would use the Internet to pick up the list to check what needs to be repaired. This list translates into the order list for a supplier.

- Sales cleared calculation

  Having long-term contracts means that certain milestone payments do not

necessarily coincide with month end. Thus, during the contract negotiations with the client, specific milestones are set up to define the basis for sales to be cleared.

- Income statement and balance sheet forecasts

All of these needs had to be addressed by whatever system UCW chose to install. SYSPRO e.net solutions was not the only system being investigated, but was the system chosen due to the software's flexibility and ability to integrate systems already in use (one of the properties of a good SOA.)

Existing Conditions

UCW has been in business for many years, and already had systems in place that catered to many daily business needs. These in place systems would have to be incorporated into the final Service Oriented Architecture:

- In-house developed AS400 system

  There was however a support risk as AS400 skills became scarcer by the day, leaving the system with no capacity for future growth. Interestingly, technology was not the driver for change, but the fact that their business needs had surpassed the system's capabilities.

- Controlware Solution batch integration to AS400

  Given that UCW already had real-time production tracking on the production floor, implementation of SYSPRO was easier. Nonetheless, this still required a huge culture change.

- AS400 emulation running on scanners

  This handled all the transactions that batched through to the AS400 system. Operational portions were already tailored to the needs. It was now necessary to get the financials and the scheduling and project management systems in place.

- The rest of the tools were bought in isolation:
  - Solid edge;

- Payroll;
- Excel spreadsheet. Used for estimation and commercial evaluation, with financial reporting delivered manually to the head office at Murray and Roberts.

UCW realized that no standard, off the shelf package would cater to the company's specific needs. They were specific and knowledgeable about what they required and how they wanted the system to operate. Given the open architecture of SYSPRO e.net solutions, UCW saw it as able to tailor to the system to full business needs, rather than the other way round.

It was important to first assess what systems were in place so that the structure of the SOA incorporating these systems and SYSPRO e.net solutions could be planned efficiently. Once this assessment had taken place, the planning phase could continue.

Planning

Any project needs to take stock of what is available and plan how to use or upgrade it, as well as plan what additional functions need to be added. The whole system (what is available and what is to be added) needs to be analyzed, and an overarching plan or map needs to be made. The plan should detail what will change, what will be added, and when these changes and additions will happen. An implementation schedule needs to be created and used to check progress along the way.

The team from SYSPRO, in conjunction with the staff at UCW, created an implementation and costing plan that spanned 18 months of work. The plan included system evaluations and integration programming and technical specification analysis, development, testing and execution of the new parts of the system and the integrated legacy systems, migration from old systems that were being replaced by "SYSPRO e.net solutions" and specific 'go live' dates followed by support and project reporting. The team planned on having most of the integration and new systems live and fully functional within the first 8 months of the migration project.

SYSPRO's rapid implementation schedule was shorter than the other system architectures that UCW had been considering. Another big influence on the company's decision making process was that an Advanced Planning and Scheduling module was already integrated into SYSPRO.

After the team from SYSPRO had analyzed UWC,s business processes, it realized it would not be able to employ the internal SYSPRO Projects and Contracts module as

a solution for the system, so it began to develop a suite of tip modules, all integrating with SYSPRO e.net solutions. This is a breakdown of each module and how it integrates with the SYSPRO core:

1. Estimating

   Each business process starts off with an estimate, the project manager providing a structure for the estimator to estimate a specific contract. The estimator then adds all the estimating elements, starting at the baseline at that specific point in time. The baseline is what it would cost to build one loco today. Since UCW works on a 2 to 3 year contract basis, a projection is done into the future determining what the cost of the materials would be and whether a project will be profitable or not.

2. Commercial Evaluation

   New escalation conditions on materials and labor come in at the commercial evaluation stage. Once UCW has tendered for a project and received the tender from the client, the commercial evaluation would be run once again with the new escalation conditions, now contracted with the client. This determines what the cash flow and expected profit of a specific project would be.

3. Project Management

   After acceptance from the project manager, planning is available for a specific contract or range of contracts. This stage is the interface into SYSPRO. Which delivers a roadmap, from the project manager into the planning system, or the structure. The users then go ahead and flesh out the specific hierarchy in the planning module. Through job creation in SYSPRO e.net solutions, a master to sub-job hierarchy is proposed. This all happens for every single contract to be produced.

The SYSPRO system takes care of the whole execution portion of the project system. The SYSPRO core system integrates with the project's Estimating and Project Management modules. Initial new project data becomes the input for the estimator from which to form an estimate. This is then used later with a timeline as the schedule for planning. The planning department does not specify the planning dates. These are dictated by 'Microsoft Project' schedules as start and end dates in the SYSPRO system.

UCW chose the SYSPRO architecture because of its competitive costing proposal, SYSPRO's e.net solutions greater flexibility, an open architecture that provided easy integration, the provision of a solution, not just a tool, the rapid implementation schedule, the Advanced Planning and Scheduling module and the flexible reporting available throughout the system. UCW could assess all of this because of the comprehensive evaluation and planning phases of the project.

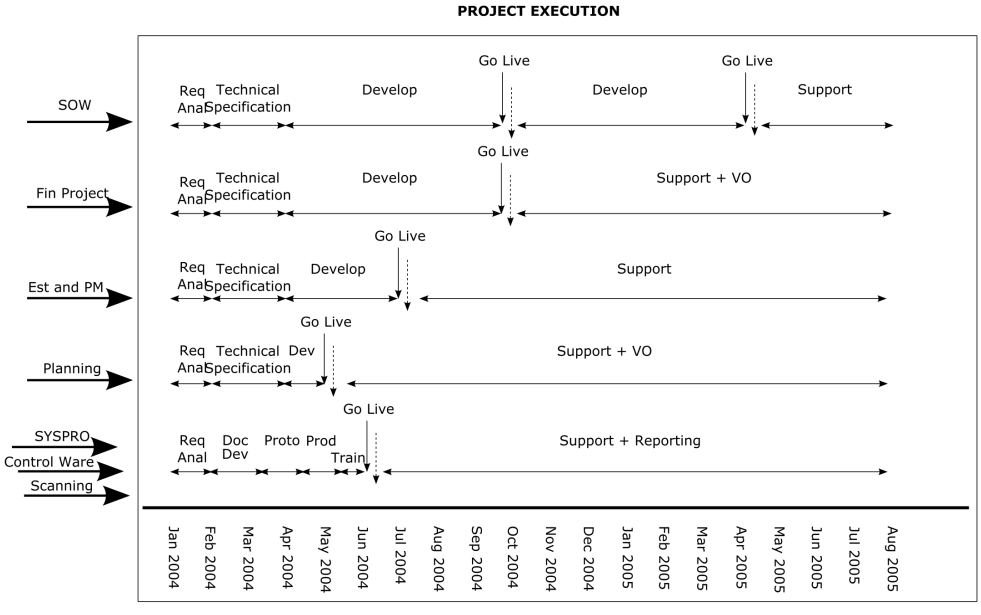
Once Union Carriage and Wagon had made its decision to install the SYSPRO system and the team had completed the full planning phase, installation and execution of the new integrated SOA proceeded.

Execution

Requirement analysis on this project began in January 2004 and lasted roughly a month, so by February the execution team was able to start defining the technical specifications to be used by the SOA and begin the process of documenting them. This was followed by the development process which started around the middle of March. The Planning Module was finished by the end of April and went live in May. The other modules continued in development and went live as planned.

The following figure shows the execution phases:

Figure 26.1. UCW Project Timeline



While examining the execution phase of the project, it would also be good to explain two aspects of the new system that directly relate to SOA.

The first is the application flow that happens within the integrated system. SYSPRO provides the link between different project jobs as well as the scheduling of jobs. It accomplishes this through two processes:

- Contract Scheduler: moves complete stages and sub-stages and all relevant sub-jobs to a different date.
- Advanced Planning and Scheduling: can be run to determine when jobs start and when the end dates of jobs can be realistically achieved. It determines the final capacity of the different work centers.

Files are exchanged with the existing Controlware System. Approximately 3 weeks of work are passed to Controlware in accordance with the scheduled start and end dates (using Advanced Planning and Scheduling and the Contract Scheduler.) As UCW handles between 2000 and 6000 operations per day, the volume of transactions passed between Controlware and SYSPRO is quite large. A specific job is sent to Controlware. The user/employee logs onto the job and as soon as all the operations on the job are completed, the data is sent back to SYSPRO e.net solutions as a labor transaction or process. This is further integrated with mobile applications, SYSPRO Stores Solution. The AS400 emulation had been run on the existing scanners. Now, this has been upgraded onto the windows architecture using SYSPRO e.net solutions. Capabilities in terms of store issues, such as binning and stock count for relevant contracts, were also added as part of the process.

The second aspect that directly relates to our discussion of SOA is the integration of the other applications that were already in use. The drawing package used by UCW was also integrated into the planning system. Once the design team has completed a drawing or design with the relevant sub-component determined, ir is sent to the planning manager. Then, it is routed to the relevant planner for planning purposes. The plan provides or joins the material application to SYSPRO and, in turn, Controlware prints out a job card.

The Sigmanest program was integrated with the SYSPRO system. This program is needed to optimize the use of remnant stock and existing plate. SYSPRO delivers the schedule of components to Sigmanest to be manufactured in a certain time window. A 3D drawing is downloaded to Solid Edge and flattened to 2 dimensions for plate cutting purposes. Sigmanest decides the relevant stock to be used, and Solid Edge provides the dimensions for the plate clipping process.

Web-based applications were also developed for Scope of Work (SOW) and PO commitments. The financial project management module gets throughout from the process and as milestones are reached, the client can be invoiced. This helps UCW to clear costs and sales every month and brings greater accuracy when costing into the future.

The estimating and project management portions, as well as the planning solution, were developed to finish ahead of the SYSPRO implementation. The reason for this was that the planning method in the AS400 system was vastly different to the way UCW is planning now. Thus, no data conversion could be employed but planners had to recapture all the planning for all the different contracts before it was possible to go live with SYSPRO. For a period of 6-8 weeks it was necessary to keep both planning systems live. In terms of the rest of the implementation, it was a matter of shutting down the old applications and starting up with SYSPRO.

One final comment on the execution phase. The SOW (Scope of Work) was never communicated as part of the original specification. It was a rebate system running somewhere in isolation, operated by 2 people. The team soon realized this would not work in the business process and so redeveloped the solution, going live with it in May 2005. It has been running ever since.

The flexibility and ease of integration shown by this project should reaffirm what we have said about Service Oriented Architectures.

Results

Once the new system was up and running, there were plenty of results that indicate a successful migration to a SOA. The full benefits of this installation will, however, not be measurable for some time. The full benefits of SOA are in the long term; therefore, UCW is well placed to enjoy them.

These are some of the impressive results of the system transformation of UCW:

- Elimination of 'software silos' -- Every piece of software that UCW uses now has a link to or is integrated with SYSPRO. SYSPRO eliminated all manual delivery printouts and transitions and converted them to digital information that is now communicated through the system.
- Real time project tracking and costing (spend curve and % completion) -- This is a real-time tool that the project manager can use on a day to day basis to monitor whether his project is progressing as planned. At any one time there are 5 to 8 project managers, each with their own project to run out of the same factory. They are permanently in competition with one another to liquidate times on different contracts.
- Real time cost and duration projection -- Reports communicate progress on projects.
- Continuous measurement between estimated, planned and actual costing and timeframe -- As soon as planning is complete to a specific sub-stage, the information is communicated to the project manager. The project managers are able to assess what the estimate was versus how the project is progressing. An early warning system is in place in order to check the actual costs/materials used against what was projected. If the values don't correlate, then the project manager receives notification and is able to investigate.

- Real-time job tracking and scheduling.
- Automated cost, sales projections and clearances.
- Precise Management - approval work flow process -- Sends it to an adjudicator, sent for client's approval and is costed accordingly.
- Automated Financial Forecasting -- Delivers all the financial forecast figures into SYSPRO. The old Excel system is no longer used. Statistical calculations then produce a Project Forecast straight from SYSPRO.
- 'The event is the transaction.' -- For every event in the system at a specific, relevant point, a transaction is incurred and posted throughout the system.

Whenever a company upgrades a system or installs a new system there is potential for new functionality within the system. At UCW the reason for installing the SYSPRO system was the need for more integrated functions and real-time reporting. A service oriented system like SYSPRO brings this flexibility and business advantage to those companies that are seeking to improve their competitive position.

Chapter 27. Bendalls Engineers

The Company

Bendalls Engineering, a division of CARR'S Engineering, was founded in 1894 and designs and manufactures specialist steel fabrications and assembles electro-mechanical equipment for a wide range industries. For example, its experience of the petrochemical industry spans many continents. Project destinations (for products from Pressure Vessels to Gas Scrubbers) run from Azerbaijan and China to the United States of America. While its clients include most of the major global players in the industry. Equally, the company has supplied the nuclear industry for over 45 years and is now an approved supplier to British Nuclear Group. Most recently, it was appointed as Lead Supplier to the BNFL Sellafield Site for the supply of machined and steel fabricated spares.

In meeting the requirements of these customers, the first clear distinction between how Bendalls operates and that of a typical manufacturing operation, is that design and engineering input is almost always the starting point for every order. As a result, its internal management processes and systems have to operate within an environment where there is very little in the way of standard products or even subassemblies, no Bills of Material (BOM), no forecasting of demand, no standard costs, no buying in materials in advance and so on.

Similarly, for mass producers, the production focus is on managing and improving processes, to control and improve costs, lead-time and quality. While Bendalls, as with all engineer-to-order (ETO) companies, needs to focus on every aspect of each job, in order to keep costs within budget, to meet set delivery dates and to ensure quality - for products that need to remain safe and reliable in the most hostile of environments - is never compromised. In fact, key operational differences run throughout the businesses even down to dispatch and invoicing, where detailed documentation packages have to be put together for each product and approved by the customer.

The company contacted McGuffie Brunton, a SYSPRO UK Corporate Partner and after evaluating the needs of their business in relation to what SYSPRO could offer they selected and implemented SYSPRO. SYSPRO was able to provide an integrated data flow business wide - with sales, engineering, purchasing, production and accounts each having access to a single data set through their own workstations. In addition, the system's flexibility delivers the greater discipline, practice

improvements and enhanced project management the company needs to compete more effectively. Real time data and access to that data from multiple points within the business is one of the facets of a SOA that Bendalls Engineering's system demonstrates.

The System

One of the main SOA features that Bendalls has benefited from is the flexibility inherent within a good SOA. The SYSPRO system has the flexibility and customization needed to meet the business needs of the company.

Unlike the earlier business system that failed to take account of the operational differences of an engineer-to-order business, the key to the successful introduction and ongoing development of SYSPRO at Bendalls is that it has proven flexible enough to 'logically' and effectively encompass the way the ETO business has to operate - from quote to invoice - and provide the management support the company needs.

As soon as an order is received, SYSPRO creates a job number and job file, without the need for a BOM. From this point, all the important information and documents relating to the job are made accessible through this file.

For instance, the company has developed its own estimating system for pressure vessels and these quotes are now easily fed directly into the ERP system and held within the appropriate job file for ongoing reference. Also, once an order is won, the company,s first task is to produce production drawings, within the in-house CAD system. However, while these drawings are still being produced, engineering can now provide access to an incomplete parts list of long lead-time materials and components through the job file. This enables the buyers to immediately start to work off this list, raising orders against the job number. Once the production drawings are complete and approved by the customer, these along with the final part list are then also made directly accessible to everyone through hyperlinks within the job file.

Further, as soon as bought in items are received, checked and passed by inspection, the test certificates are cross-referenced within the file to the published parts list. Similarly, final product inspection and certification details are recorded within the job file. This helps to ensure complete traceability of materials, components and work throughout the production process.

The Results

Neil Murray Finance Director for Bendalls Engineering, says: "Since the introduction of SYSPRO, jobs are just so much better 'hung together' in that everyone has easy access to the information they need, when they need it."

Crucially, tracking individual job costs and progress on a daily basis is also en easy task. The system's WIP module allocates costs to individual jobs, with labor entered manually from daily timesheets and material costs set against the jobs as soon as items leave inspection. From this date, the company produces a weekly modified report that shows the total cost build up of each job to date, along with the cost of committed materials. The report also includes the original estimate and this enables ongoing reconciliation of actual cost against estimate.

"Previously cost information was just not accessible in a format that was readily usable. Essentially, single job reports were just not possible and it was a case of accessing all labor costs and material costs and manually cross referencing them. A very time consuming and error-prone approach," explains Murray.

"This modified WIP report has become a major part of our production control system and typically forms the basis of production meetings. We can monitor individual job costs and progress. As soon as a project starts to go off track, we know about it and can take corrective action. "By using the report's data to assess the progress of all the ongoing jobs, production control can readily determine present and future workload and see which fixed resources may become stretched, or conversely identify where there might be low utilization."

The integrated nature of SYSPRO offers a wealth of general benefits, which the engineering department has become increasingly keen to exploit. For example, accessing the WIP module allows engineers to look at current jobs, monitor stores inventory and track supplier lead times. The system has also brought new disciplines, such as the inputting of information at the right time, to key activities.

“In the past the excuse was always that people were 'too busy to do it properly.' Yet, it is now obvious that when tasks are done properly and the set disciplines followed, it actually saves time and effort,” states Murray.

Finally, the preparation of accounts and invoicing has become more professional with all essential information accessible through a single centralized data source. The preparation, validation and submission of a complete documentation package at the end of the project are much faster and efficient.

As Murray concludes “SYSPRO has been proven in meeting our specific needs. Everyone has immediate status visibility on every job, from order to final invoice and a raft of detailed general information and key analysis data is available within seconds. At the same time, we have seen improvements in the way we operate in every department. Overall, we are not only managing individual jobs more effectively and efficiently, but the system has made managing the whole operation far simpler and easier.”

Chapter 28. Cedarlane Laboratories Limited

The Company

Founded in 1957 and incorporated in 1975 by three researchers from the University of Toronto and the Ontario Cancer Institute, Cedarlane Laboratories Limited has matured into a world leader in the research and development of reagents - materials used to detect, measure and prepare other substances for research, particularly in the area of immunology.

Located in Hornby, Ontario, Cedarlane distributes products of more than 300 companies worldwide to over 20,000 life science researchers who trust Cedarlane's quality products and service excellence. In 1997 Cedarlane chose SYSPRO software to assist in running their operation.

In 2000, Cedarlane decided to diversify by establishing a daughter company, Cedarlane Shipping Supplies Inc., to specialize in shipping supplies for temperature - sensitive goods. Because of the highly sensitive nature of their products and the company’s commitment to the highest quality standards possible, Cedarlane Shipping Supplies was a logical evolution. “Our success is measured through satisfied stakeholders enjoying the benefit of a stable and viable organization which strives for continuous improvement. Cedarlane Shipping Supplies is a result of this mind-set,” says Susan Gater, Operations Manager, Cedarlane Laboratories.

SYSPRO Scalability

SYSPRO’s manufacturing, distribution and multi-company functionality allowed Cedarlane to efficiently seize the opportunities they saw in distribution and quickly formulate strategies for growth in the new business without the additional operational expense of acquiring new software. The modular nature of the software provides scalability and flexibility allowing Cedarlane to select only those functions needed to increase operational control and efficiency. “SYSPRO enables us to run both our manufacturing and distribution business on the same system; it has given us a solid foundation for successful supply chain management, from purchasing through manufacturing and distribution,” says Susan.

By 2003, Cedarlane's distribution arm had a supplier catalogue database, which had grown to over 1,000,000 items, and once again, Cedarlane looked to SYSPRO with a new requirement. The challenge was to make use of the existing, continuously growing, supplier catalogue database to automatically update their SYSPRO inventory system at the time an order was placed. Manual processes existed to update the inventory files, which resulted in delayed customer service - specifically, order fulfillment and inventory accuracy. Because the supplier catalogue database contained all available products, some of which Cedarlane's clients might never be in the market for, the challenge was keeping unnecessary product items out of the SYSPRO database until the items were sold. Out of the 1,000,000 possible product database, Cedarlane estimated that only 20% of these would actually be in demand by their customers; thus, they did not want to populate their inventory with unnecessary items.

SYSPRO SOA Solution

The solution involved the development of a custom sales order entry screen so that when the customer service personnel select a stock code, the supplier catalogue database is read and compared to the SYSPRO Inventory Master file. If the item does not exist, the system automatically updates all relevant SYSPRO files with the details for the item including, supplier contract pricing, supplier/stock code cross reference, pricing and warehouse details. Should the stock code exist in SYSPRO then the information is compared to the latest information from the supplier catalogue database. Once all lines are added, the entry is saved and a sales order is automatically generated, which can be maintained in SYSPRO and the necessary documents may be printed. “New customer service personnel are fully operational and entering orders within a day, whereas before it would take months to train an individual,” comments Susan.

SYSPRO’s ongoing product development is focused on leveraging key technologies such as Microsoft .NET® and COM (Component Object Model) to allow for interoperability between SYSPRO and other business critical applications. The SYSPRO e.net solutions framework provides a structured way of directly accessing the business functionality within SYSPRO software while maintaining the software’s built-in business rules and security.

SOA Results

Cedarlane gained the following SOA business benefits from the integration between SYSPRO and the supplier catalogue database:

- Quick and easy selection of items from the extensive supplier catalogue database by customer service personnel
- Elimination of duplicate database maintenance
- Improved accuracy of inventory and pricing information resulting in improved customer service levels
- Self-maintained, expandable system to accommodate future growth and expansion
- Seamless integration with its existing SYSPRO system and compatibility with future SYSPRO releases
- Significant reduction in staff training time

These results are not just once off benefits, but benefits that will continue as the system is used and grows with the company. Over a longer period other benefits of SOA will also be measurable, so Cederlane is well on it's way to achieving the competitive advantage predicted by SOA experts within the markets that they compete in.

“We are a dynamic organization whose product offerings are reflective of the research community’s changing needs and SYSPRO has been a valuable resource by providing us with the tools to empower our strategic decision making capabilities,” concludes Susan.

Currently Cedarlane is in the planning stages to duplicate the SYSPRO e.net interface for the purchasing of materials. This is part of their global strategy as they actively pursue new markets.

This page intentionally left blank

Bibliography

Books

Dirk. Krafzig, Karl Banke, and Dirk Slama. Copyright © 2005 Pearson Education, Inc.. 0-13-146575-9. Prentice Hall Professional Technical References. *Enterprise SOA - Service Oriented Architecture Best Practices*.

[newlom] Eric. Newcomer and Greg Lomow. Copyright © 2005 Pearson Education, Inc.. 0-321-18086-0. Addison-Wesley. *Understanding SOA with Web Services*.

Web Articles

Technology Evaluation Centers. Technology Evaluation Centers, Inc.. http://www.technologyevaluation.com. “Enterprise Applications - The Genesis and Future, Revisited”. P.J. Jakovljevic. Copyright © 2004 Technology Evaluation Centers, Inc.. http://www.technologyevaluation.com/Research/ResearchHighlights /Erp/2004/03/research_notes/TU_ER_PJ_03_31_04_1.asp.

[gartner_soa_smbs] *Gartner*. Gartner, Inc.. http://www.gartner.com/. “How Service-Oriented Architecture Will Affect SMBs”. Charles Abrams, Robert P. Anderson, and David Mitchell Smith. Copyright © 2005 Gartner, Inc.. http://www.gartner.com/DisplayDocument?ref=g_search&id=476373.

[gartner_five_hottest] *Gartner*. Gartner, Inc.. http://www.gartner.com/. “Gartner's Positions on the Five Hottest IT Topics and Trends in 2005”. David W. Cearley , Jackie Fenn , and Daryl C. Plummer. Copyright © 2005 Gartner, Inc.. http://www.gartner.com/DisplayDocument?doc_cd=125868.

[gartner_benefits_challenges] *Gartner*. Gartner, Inc.. http://www.gartner.com/. “Benefits and Challenges of SOA in Business Terms”. Michael Barnes , Daniel Sholler , and Paolo Malinverno. Copyright © 2005 Gartner, Inc.. http://www.gartner.com/DisplayDocument?doc_cd=130078.

[gartner_findings] *Gartner*. Gartner, Inc.. http://www.gartner.com/. “Findings From the 'All Software' Research Meeting: The Challenge of Service Optimization in SOA”. Simon Hayward. Copyright © 2005 Gartner, Inc..

http://www.gartner.com/DisplayDocument?ref=g_search&id=485865.

ebizQ. ebizQ, Inc.. http://www.ebizq.net. “SOA Fundamentals”. Beth Gold-Bernstein. Copyright © 2005 ebizQ, Inc.. http://www.ebizq.net/hot_topics/soa/features/6245.html.

[bloomberg] *Application Development Trends Magazine*. ADT Mag.. http://www.admag.com/. “Principles of SOA”. Jason Bloomberg. Copyright © 2003 101 Communications. http://www.adtmag.com/article.asp?id=7380.

SOA WebServices Journal. SOA WebServices Journal. http://webservices.sys-con.com/. “SOA: Learning from the Past to Disrupt the Future”. Kerry Champion. Copyright © 2004 SYS-CON Publications, Inc. . http://webservices.sys-con.com/read/45788.htm.

[foody] *SOA WebServices Journal*. SOA WebServices Journal. http://webservices.sys-con.com/. “SOA Command and Control - Taking SOA to the Next Level”. Dan Foody. Copyright © 2005 SYS-CON Publications, Inc. . http://webservices.sys-con.com/read/48744.htm.

ebizQ. ebizQ, Inc.. http://www.ebizq.net. “Making Future Shock Less Shocking”. Gian Trotta. Copyright © 2003 ebizQ, Inc.. http://www.ebizq.net/topics/soa/features/3389.html?&pp=1.

[philjam] *Enterprise Architect*. FTP Online. http://www.ftponline.com/ea/magazine/. “Managing the Impact of Change in an Enterprise Service Environment”. James Phillips. Copyright © 2005 Fawcette Technical Publications. http://www.ftponline.com/ea/magazine/summer/online/jphillips_05_30_02/.

SOA WebServices Journal. SOA WebServices Journal. http://webservices.sys-con.com/. “Best Practices in Integrating Data Models for SOA”. Jim Gabriel. Copyright © 2005 SYS-CON Media.. http://webservices.sys-con.com/read/48031.htm.

SOA WebServices Journal. SOA WebServices Journal. http://webservices.sys-con.com/. “SOA Command and Control”. Dan Foody. Copyright © 2005 SYS-CON Media.. http://webservices.sys-con.com/read/45788.htm.

Loosely Coupled. Procullux Media Ltd.. http://www.looselycoupled.com/. “Security rules in SOA managementl”. Keith Rodgers. Copyright © 2005 Procullux Media Ltd.. http://www.looselycoupled.com/stories/2004/secure-ws-infr1124.html#content.

Developer.Com. Jupitermedia Corporation. http://www.developer.com/. “The Benefits of a Service-Oriented Architecture”. Michael Stevens. Copyright © 2005 Jupitermedia Corporation. http://www.developer.com/design/article.php/1041191.

Developer.Com. Jupitermedia Corporation. http://www.developer.com/. “Understanding Service-Oriented Architecture”. Michael Stevens. Copyright © 2005 Jupitermedia Corporation. http://www.developer.com/design/article.php/2207371.

MSDN. Microsoft Corporation. http://msdn.microsoft.com//. “Understanding Service-Oriented Architecture”. David Sprott and Lawrence Wilkes . Copyright © 2004 Microsoft Corporation. . http://msdn.microsoft.com/library/default.asp ?url=/library/en-us/dnmaj/html/aj1soa.asp.

[cpfrmodel] Copyright © 2004 Voluntary Interindustry Commerce Standards. Voluntary Interindustry Commerce Standards. *Collaborative Planning, Forecasting and Replenishment*. An Overview.

This page intentionally left blank

SYSPRO™

SIMPLIFYING YOUR SUCCESS

Service Oriented Architecture (SOA) is all about business driven use of information technologies. It's about business being able to reduce risk and cost in a world where alignment and optimization of strategic sources and globally distributed operations is the key to profitable service delivery.

SYSPRO, internationally recognized as the worlds largest independent developer and supplier of business applications, was an early adopter of the concepts and technologies required in order to enable and support business process management in SOA-enabled environments. Today SYSPRO's pioneering spirit, expertise and experience uniquely position the company to assist businesses in understanding the business and IT benefits of SOA.

SYSPRO on SOA is a guide to Service Oriented Architecture. The book approaches what is a complex and confusing technical subject from the perspective of business and aims to explain SOA in the clearest and simplest. If you are a non-technical business manager, then SYSPRO on SOA makes an excellent companion that will help you decipher what your technology experts are speaking about.

Like SOA itself, SYSPRO on SOA aims to mask complexity. It's just one of the ways SYSPRO is "Simplifying your Success."

Expert Comment:
"The book is quite educational, even for non-techies."
P.J. Jakovljevic

Produced by:
SYSPRO Press

www.syspro.com