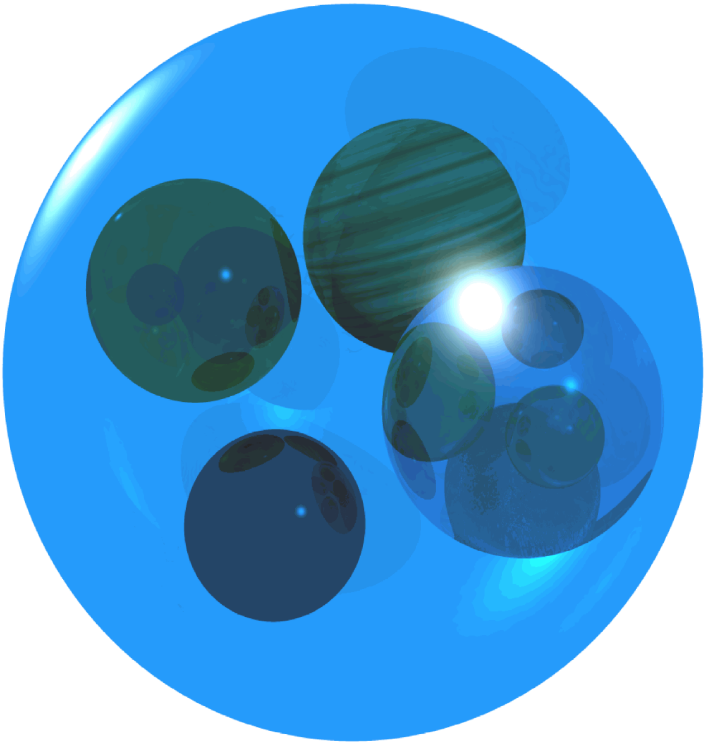
Using SOA to Deliver IT to the Business

Enterprise SOA Adoption Strategies



Steve Jones

InfoQ *Enterprise Software Development Series*

FREE ONLINE EDITION

(non-printable free online version)

If you like the book, please support
the author and InfoQ by
purchasing the printed book:
http://www.lulu.com/content/408923
(only $27.95)

Brought to you
Courtesy of
Steve Jones &

This book is distributed for free on InfoQ.com, if you have received this book from any other source then please support the author and the publisher by registering on InfoQ.com.

Visit the homepage for this book at:
http://infoq.com/books/enterprise-soa

Enterprise SOA Adoption Strategies

Using SOA to deliver IT
to the business

Written By:
STEVE JONES

C4Media, Publisher of InfoQ.com.

This book is part of the InfoQ Enterprise Software Development series of books.

For information or ordering of this or other InfoQ books, please contact books@c4media.com.

Production Credits:
Managing Editor: Floyd Marinescu
Cover art: Gene Steffanson
Composition: Laura Brown

Library of Congress Cataloguing-in-Publication Data:

ISBN: 978-1-84728-398-6

Printed in the United States of America

Acknowledgements

I would like to thank Steve Meyfroidt and Mike Morris in particular for help with this book they both have put in a lot of work over the years in developing service oriented thinking and delivering actual service oriented projects, without their help and experience this would be based solely on my jaundiced view of the world. Thanks to Floyd Marinescu for offering to publish and the support in getting it over the line. And thanks to my various employers over the year who have let me do service orientation when it didn't have the trendy name it does today, in particular Capgemini and netdecisions. I'd also like to thank the folks at various product vendors who have put up with my rants about SOA. Big thanks to Charles Beckham and Edwin Khodabakchian for listening and delivering, and of course for golf and BBQ respectively.

I'd like to thank the folks at OASIS who accepted the methodology contribution that forms a large part of this work and in particular to Miko Matsumura the chair of the SOA Adoption Blueprints group who convinced me to send it out. This book was edited using Open Office and saved using the OASIS standard OpenDocument format. I'd also like to give a shout out to the OASIS SOA Reference Model group more than ably led by the excellent Duane Nichol, without a doubt this has been one of the most interesting standards efforts I've been involved in and is the most coherent definition of SOA out there.

Finally and most importantly apologies to my superbly wonderful wife Heather for being locked out of a room in our house while I got this finished. I promise I'll stop claiming I

have to write the book when the kids start playing up. And everyone please visit her blog at http://hboutique.blogspot.com as its brilliant.

Contents

1

Introduction

Service Architecture, SOA, Event Driven Architecture, EDA and many other architectural buzzwords abound in technology at the moment. Although the standard view is that Services are a *technology* solution to add agility, past experience teaches us that technology solutions rarely deliver agility except when they are focused on the business visions. Services have been hijacked by technology vendors trying to sell integration and development tools, which most normally focus on "Business Process", "Orchestration" or "Web Services". This technology driven approach fundamentally misses the point of Services. The objective of a service is to represent *what* the business does and place a boundary which all parties, but predominately the business, can agree on. It is this representation of the business on which is the creation of a Service Architecture must be focused, technology is very much a secondary element and many of the services will **never** even be realised using technology. The risk of an IT related approach it becomes focused on technology rather than Method, this is a fate that SOA must avoid.

The key to ***Service*** Oriented Architecture?

"It's the Services Stupid"

Purpose

The purpose of this book is to create a "service discovery" approach that leads to the business service architecture of an enterprise or project. With all the hype surrounding SOA there has been a lamentable dearth of modelling approaches for the

actual "Service" and "Architecture" pieces. Organisations have been happier to concentrate on process or implementation rather than the construction of a well formed service architecture.

So the aim here is to provide both the notation and approach for the creation of that architecture and to provide a notation that is not currently available in other architectural notations. For enterprise modelling this notation should be considered as being at the *conceptual* level, while for projects it is more at the *contextual* level. This notation and approach is particularly aimed at collaborative discovery of business service architectures, which would then drive the complete architecture. In simple terms, this notation is not aimed at defining the detail of the services, its about helping you clearly identify what the services should be. The final definition of services and their implementation is an extremely non-trivial process that must involve formal architecture, technical architecture and finally solution design and implementation. There are many methodologies and approaches in these areas, the purpose here is to provide the context that ensures they are service oriented.

While the majority of this book concentrates on the architecture, the later elements are brief overviews as to how this impacts other parts of the life-cycle and the extra control it can enable. The methodology itself as it relates the definition of service architecture has been contributed[1] to OASIS and as such is available for anyone to use, build tools, or implement with no royalties or claim.

Abstract

Service Oriented Architecture is a powerful term that is regularly abused to refer to development technologies rather than an architectural approach, in the same way as Object Oriented Design was abused to refer to programming languages rather than a fundamentally different approach to design. This book lays down a methodology for ***Service*** Oriented ***Architecture*** which deals only with Service as it applies to Architecture, and with Architecture where it is about exposing the Services

framework. Its purpose is to explain how a Service Architecture is created, how this in turn drives Service Orientation in broader Enterprise and Solution architectures, and how this impacts all aspects of IT delivery from Business Process to standard code development. The Service Architecture methodology described here is fundamentally a process of top-down discovery of what a business or business problem is about, rather than the "how" of implementation. Architecture is about context, frameworks, blueprints and standards, not about the individual aspects of delivery. Other methodologies already describe how to deliver software projects, this methodology helps provide the architecture to ensure that the delivery is Service Oriented.

This book is predominately focused on the first element of either a business process, enterprise architecture, solution architecture or project engagement. The objective here is the first few days, and at most weeks, of a task to either map an enterprise's services, re-engineer its processes or to deliver an IT project for that organisation. As such this approach covers only

- Why Services need to be defined
- The importance of a common language
- How to discover what are the primary business services
- How to identify shared and supporting services
- How to define the interactions between services at a high level
- How to categorise services to help with management
- How Service Architecture enable changes in programme and project management
- How to use processes with services
- How to use services with traditional requirements approaches
- How to use services to change the way projects and programmes are managed

What won't be covered:

- Full Enterprise or Solution Architecture
- The technical requirements of services

- The functional requirements of services
- The implementation of services
- Management of service programmes

The reason for this is that these other elements all augment the basic service model, it is therefore critical that that service model is done **before** all of these other elements are attempted. Otherwise the architecture cannot be said to be service oriented.

2

Overview

Service has become the accepted phrase to describe how systems should be exposed and coordinated . Building on the technology roadmap discussed in a previous paper[2] this book concentrates on how SOA impacts the approach to project delivery, and how a Services approach drives both enterprise architecture and business process efforts. Using a Service Architecture approach it becomes simpler to answer one of the most challenging questions in the delivery of IT - "what do I deliver where?". Where most methodologies, including RUP[3] and XP[4], are based around "single room" delivery approaches and concentrate on the artefacts within the project the key to modern delivery is that distribution of the project team is now the norm, not the exception. Technology vendors concentrate on the technology implementation challenges if SOA. Industry efforts such as the OASIS SOA[5] groups deal with defining the terms and conditions of SOA, others deal with the business view or a technology view. The objective of this book is to provide a simple approach to the "Service" question in Service architecture and to provide a mechanism for planning, managing and delivering projects using SOA techniques.

This book will cover how to create a strong business oriented service architecture, how that architecture will effect the management of IT delivery projects, how business process and requirements gathering should be done in the context of a service architecture and finally what an actual service project should look like. The focus will be on the creation of the service architecture itself and its impacts on the other areas, it will not aim to define a fully detailed project, process or requirements management approach, there are enough of those already, but to

explain how these are changed when view from a service perspective.

Defining SOA

There are a huge number of definitions as to what a service is, and what Service Oriented Architecture means. Many of these, in fact most, come from vendors and are not unsurprisingly linked to the products they have to sell. Recently however there has been a successful attempt inside OASIS to define a Reference Model for SOA[6] which has resulted, at the time of writing, in a public draft[7]. As with any area of IT there is a huge desire to re-invent the wheel, rather than pushing forwards the debate onto the next challenge we devote all of our energy debating what the colour should be. The OASIS SOA Reference Model will undoubtedly evolve in the coming years, and be enriched by Reference Architectures and implementation experience but that isn't a reason to try and create your own.

The Reference Model does not tell you how to do SOA; it just helps you to determine what "good" looks like and to define the various terms that are used in SOA. It is strongly advised to take the reference model as your basis for your project or enterprise definition of SOA, it has the advantage of being vendor, technology and business neutral and from a recognised standards body. And if you think there are omissions then you can feed these back into the OASIS group, or even join it yourself.

In this book the SOA Reference Model is used as the definition of terms, the following are three key elements

Table 1 OASIS SOA Reference Model Selected Terms

Term	*Definition*
Service	*"The means by which the needs of a consumer are brought together with the capabilities of a provider."*
Capability	*"A real-world effect that a service provider is able to provide to a service consumer."*
Service Oriented Architecture	*"Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations."*

A key factor in the SOA reference model is that it is not limited to just technology, a common failing in most other views of SOA, but can be applied to all facets of an organisation, including IT, infrastructure and the pure business elements themselves. This is critical when looking to model a service architecture, if an approach or definition is limited to technology then it will fail as it will only cover those elements that are in technology today, and not those that drive those technologies and which may become automated in future. SOA has to be about all the services in the business, not just those we can "whack a WSDL on".

Ignoring Technology

There are some critical myths to remove before continuing in this book. The first is that SOA = Web Services. This is an often cited definition, though rarer now than during 2005, particularly from technology vendors. Web Services are an enabling technology for interconnection on various different technology platforms[8] from large scale systems right down to mobile devices[9], this sort of approach is an evolution of previous generations and represents nothing new. The current argument of SOAP v REST is equally pointless as these are just about the last mile between consumer and producer rather than being

about what the information and function means to both sides of that communication. Equally people have spoken about Event Driven Architecture being the "next" thing, again EDA is just an evolution of previous messaging approaches and there is nothing new in this thinking. RPC, event driven, messaging, DDI etc etc are all well understood technical areas, the change now is that technologies are becoming simpler in this area and more standardised so the connections are becoming simpler and more commoditised.

This book will not go into any detail on the technical arguments of SOAP, REST, EDA etc as there are more than enough definitions and works out there already. What should be noted however is the following

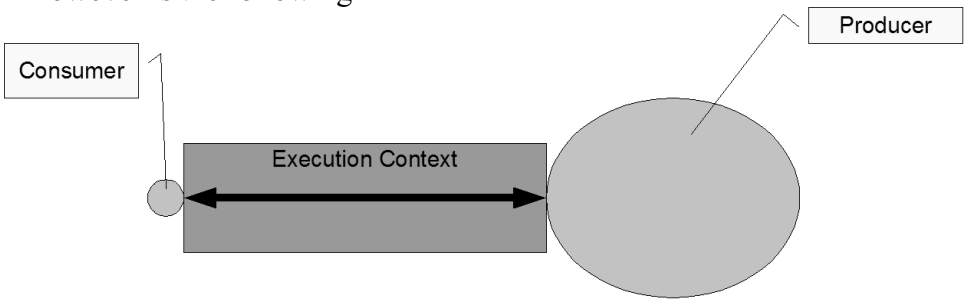


Figure 1: Execution Context for Services

The OASIS SOA Reference Model talks about the Execution context, this is the "thing" that enables the consumer and producer to communicate. It is at this level that technology is important, as the enabler between these two parties. The execution context has nothing to do with the value ascribed to that invocation nor to the reasons for that invocation happening. This book focuses on those reasons, values and drivers. Technology based SOA places a disproportionate level of importance on the execution context which obscures the basic fact: that without the consumer and producer there would be no need for that context, and the objective of the execution context should be to simplify that invocation as much as possible. The

best execution context, i.e. Technology, for SOA is one which no-one knows is there.

Technology SOA is a misnomer as it focuses on the item of least value in SOA, the execution context, and ascribes to that all of the benefits. SOA = Web Services is as silly idea as SOA = Unicode based RPC. With OO there was OOA and OOD and this technology focused approach is really about design and delivery rather than architecture, but Service Oriented Delivery of IT is unlikely to catch on in marketing circles. "Problems with your current systems? SOD IT can help", hence the abuse of the term SOA.

The Four Step Programme for SOA

A service architecture follows a broadly four step process, or more accurately delivers three stages of the process and provides the direction for the fourth.

- **What**: Defining the scope of services, this about determine what the services actually are.
- **Who**: Who are the external actors that drive the services or with which the services interact.
- **Why**: Identifying why one service talks to another, and why external actors interact with the services
- **How**: The detail about the processes that co-ordinate the services and also the detail on how a service itself will be implemented.

This four step process is about getting focusing on the higher order elements first, which provides the context for the later stages. The Service architecture

- Defines the **What**
- Identifies the **Who**
- Highlights the **Why**
- Doesn't do the **How**

This means that future phases and approaches, whether Business Process, Enterprise or Solution Architecture will refine and detail each of these elements, but it doesn't alter the relationship. What drives who drives why drives how, and never the other way around.

Figure 2 What, Who, Why, How

Its critical to understand however that while this book describes how a service architecture can be defined, it does *not* define how that service architecture can be delivered. It does not attempt to describe the full amount of information that is required for the successfully delivery of a services architecture solution or enterprise, this omission is deliberate. The purpose of this book is to explain the start of the process and get things moving from the beginning of the roadmap. SOA does not replace other elements of architecture and delivery rigour, it just gives a proper framework into which they can sit. It's critical to understand that SOA provides only a framework, and that a complete architecture *must* deal with the **how** and provide a structure for strategy, implementation and support that is

deliberately excluded from the Business Service Architecture. The amount of architecture required must be driven from the value and priority of the business, not from the desire to create vast architectural documents.

3

Why A Service Architecture Is Important

Before embarking on the creation of a Service Architecture it's important to understand what it aims to give you, and why you should be starting from this point rather than taking a more traditional approach of either mapping out business processes or jumping straight into the system requirements.

The first element that a service architecture aims to deliver is the "big picture" this will act as an overall guide to an enterprise or project and provide a simple view of how the organisation, or project, splits its capabilities into services. The big picture needs to be something that all parties agree on, and all parties use as a reference. For a project it might fit on a sheet of A4 or A3, for an enterprise it might be on A0. This is a key factor of the big picture, it needs to be a picture, too often in IT projects the value of images is under-rated.

This big picture is used to aid in understanding how change requests will be handled, new projects commissioned and business change delivered through IT adherence Make the big picture clear, use clear colours with a defined key, and most importantly ensure that it is kept up to date. The purpose of a good service architecture is not to focus people's minds on to the detail of the application, but to ensure they always have the context in which the detail is concerned. A service architecture concerns itself first with the "what", "who" then the "why" and only finally the implementation question of "how". The Big picture is the one that tells all stakeholders the "what", "who"

and the "why", it is down to the various areas to determine the most effective "how".

The big picture is the clearest representation of the service architecture, if the picture isn't clear the rest of the architecture will be similarly obscure.

Becoming Service Oriented

Once you have the big picture you then have the ability for Service Orientation to really define how you work, the key is that for a Service Architecture to be properly delivered then you must ensure that each stage of the lifecycle is also service oriented. This means that project and programme management, requirements gathering, development and ultimately support must all be done in the context of the service architecture. Otherwise there will be a great picture delivered in the same old way, as with building architecture it sometimes takes new approaches in construction to deliver new architectural visions.

This book will not cover in detail every element that needs to be done, apart from the definition of the service architecture, but aims to give a flavour of the impacts that service orientation makes, and what it enables you to do.

The aim of moving towards a service oriented enterprise, or of just IT or one large programme moving towards service orientation, is to clearly align solutions to the business. The ambition is to create systems with names that make sense to the business rather than the code names that currently proliferate, it is this ambition that requires organisations to not only draw the big pictures but to change each step in their IT delivery process, and in how IT engages with the business. Service Orientation is about changing the way the IT organisation thinks and acts, this requires much more than the use of a new technology and a big picture.

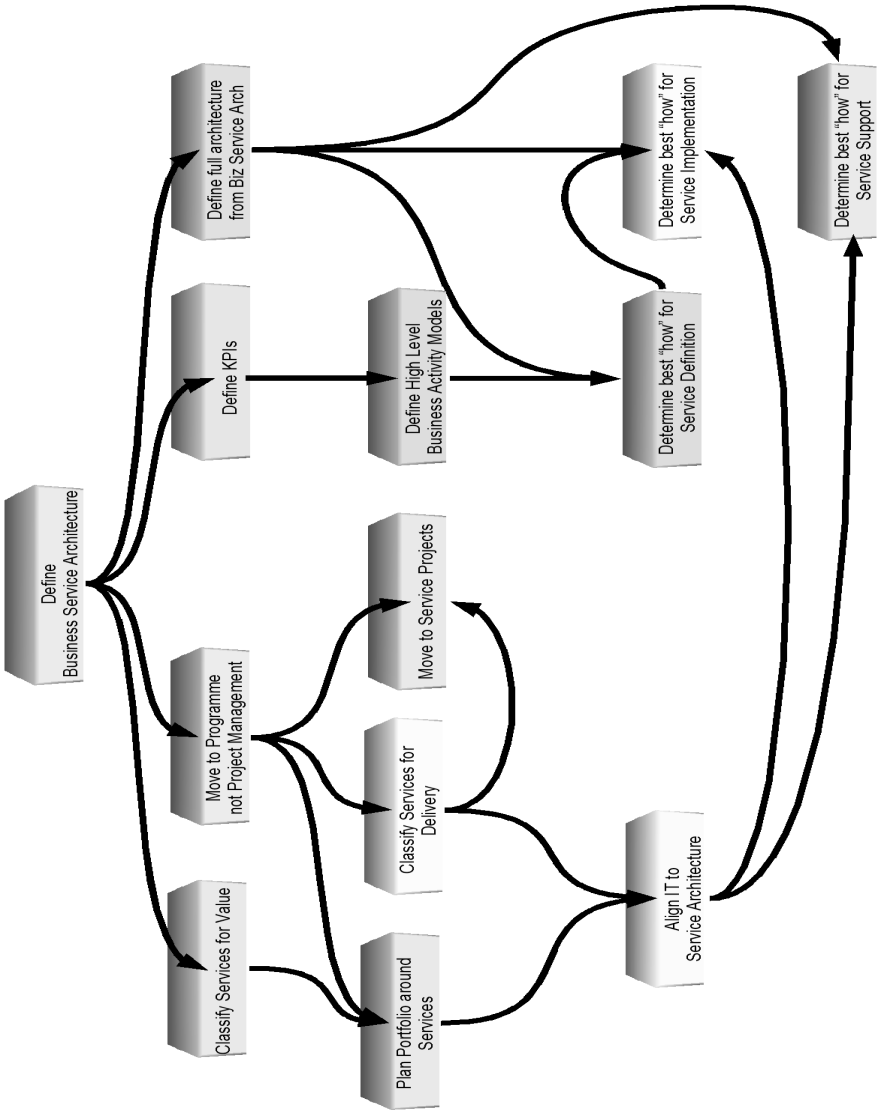


Figure 3: Service Oriented Impact

4

Start At The Top

The basis of this approach is to start at the top of the domain, whether this be at an enterprise level or for an individual area. The main reasons for doing this are

1. Organisations work "top-down"
2. Reduces clutter
3. Uses the organisational functions as its basis

When an organisation looks to implement change it does this via its structure, starting at the top. Having a structure with clear boundaries of delegation and control is essential to creating flexibility. A large organisation that had to phone everyone of its employees to tell them their new job description and manager on Monday morning would not be able to implement change very effectively. There is no reason to expect that services should be any different, if every service is viewed as an atomic, loosely coupled and autonomous entity then your are defining an approach where every service needs to be managed as an individual.

Using the organisational structure or functions of a business as the basis for services is not a new idea[10] but these have mainly been process driven rather than service driven efforts and have attempted to answer all questions rather than aiming for clarity around the key question of what services need to be made available and how. It needs to be noted that while organisational functions tend to be relatively stable, the actual structure and departments can often be flexible especially when a new senior executive is appointed. The objective is to use the organisational functions, the "what" of the enterprise, and not the temporal representation of those within an organisation chart. Modelling

the organisation structure is fraught with danger due to its tendency to change. The famous quote, wrongly, attributed to Roman writer Petronius Arbiter puts it succinctly: "*We trained hard, but it seemed that every time we were beginning to form up into teams, we would be reorganised. I was to learn later in life that we tend to meet any new situation by reorganising; and a wonderful method it can be for creating the illusion of progress while producing confusion, inefficiency, and demoralisation."* The Service architecture must therefore aim to detail the core capabilities and services of an organisation, rather than the latest org-chart that aims to manage them.

Moving Away From "How"

Service Architecture is driven by the domain, rather than the function and so represents a different mechanism for understanding how elements work together. The objective of this approach to service architecture is not to understand the skills elements, the capabilities, that have commonality but the functional groupings of an organisation, and hence the services that it provides both internally and externally.

Figure 4 shows a standard view of how Business and IT interact and how External organisations are engaged. This map could be more complex than shown when each additional function is considered in greater depth. Elements like Testing for instance are included in the "Support" group which also includes call centres and application maintenance. The key in this diagram is that it deals with the skills that are being used and not the domain under which they are used. This is typical of IT driven solutions which most often concentrate immediately on the "How" of implementation rather than first considering the reasons for that implementation.

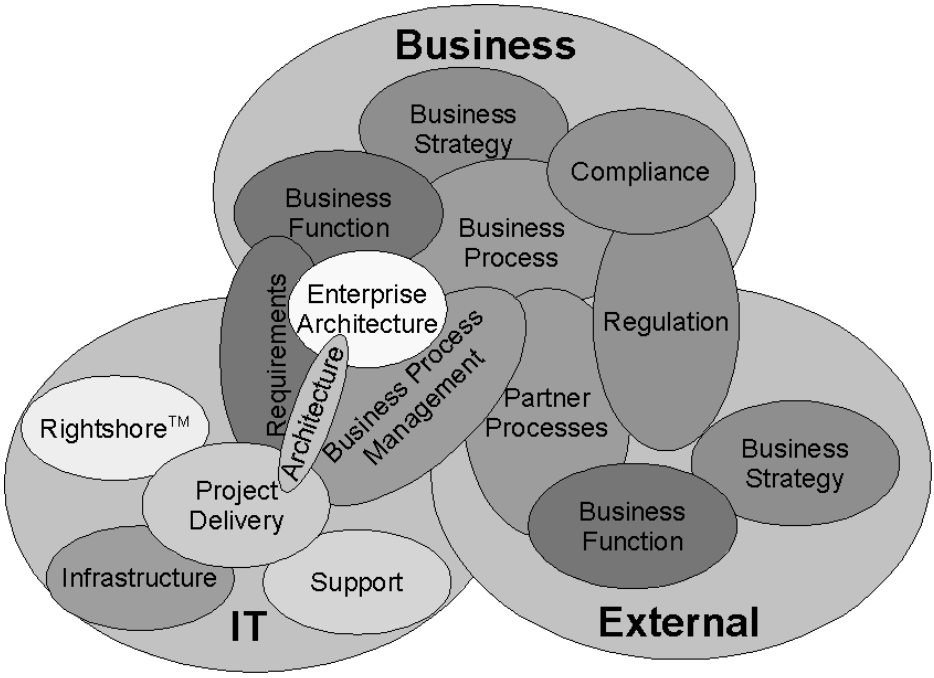


Figure 4 Standard View of IT and Business interactions

Why Not Start With Process?

A common approach historically to this sort of discovery work has been to use the high level business processes as the basis of discovery. This is explicitly *not* what this methodology attempts to do, this is for two distinct reasons:

1. Process based discovery tends to "drill-down" too early
2. Process based discovery tends to produce process "silos" of services and often fails to identify common ground between processes.

This service methodology is driven first by the broad "what" of the enterprise, and only second is processes considered as an orchestration of those "what's". It's a normal understanding that business is driven first by its end-to-end processes, yet most organisations are structured around their key functions, and the end-to-end process goes between those functions. This suggests

that organisations are, at a high level, first focused on the "what" and only secondly on the process, the "how".

Starting with process tends to lead to large "step through" workshops and interviews on that process, at this stage a service architecture is already imperilled as rather like the normal technical implementations, the "how" of the enterprise is considered too early. As has often been said, the first time that an organisation realises its processes are wrong is when IT documents them, starting with the process moves down this route and makes change harder.

It is somewhat of a myth to say that organisations are driven by their processes, in reality they are driven by a series of loosely coupled activities that in IT we mistake for hard-coded processes. This desire of IT to formally document is understandable, but it doesn't help when the objective of the business is to enable these processes to change. In simple terms modelling process is exactly the same as modelling COBOL, with all of the flexibility that COBOL systems have created. Just because there is a clearer pictorial representation of a process does not mean that it will not suffer from the same problems as previous process and procedural oriented technologies. COBOL is quite rightly not considered a highly productive language for development, and in 2006 we must ask ourselves whether Visual COBOL is the right way to model the enterprise, it is the view of this book that it is not.

5

Core Definitions and Approach

Terminology

The approach described in this book can be used at any layer of a market, an organisation or problem, the terminology and approach remains the same.

1. Level 0 - The start of a Service Architecture is the Level 0 picture. This Level 0 representation is the "70,000 ft" view of the domain under investigation. This could be to encompass a whole company, for instance to determine the 10 year roadmap for the organisation, or for a specific project.
2. Level 1 – Decomposition of the Level 0 model into finer grain services, this process can be continued to other levels (2,3,4 etc)
3. Support Services – Services which are not core to the general business or problem but which provide required functionality for the overall environment to function correctly. An example of this would be auditing for compliance.
4. Technical Services – Non business requirement functions that are needed for the IT system to be delivered. An example would be a hosting provider.
5. Contract – The functional and non-functional definition of a service, equates to a Service Level Agreement (SLA) builds on the work of Meyer[11].
6. Actor – A consumer (person, system or service) of facilities provided by a Service. Similar to a UML Actor.

Collaborative Working

Key to creating service architectures is the process that is gone through to get the information and understanding. It is all about creating a common dialogue between the various different groups and deciding upon a boundaries that work across the business.. This is not a task that can be under-taken via a series of interviews with individual stakeholders, nor is it something that can be gleaned from documents or websites within the company. The best way to get a decent services architecture is to engage in collaborative working, an area so well documented it has its own conference[12], what is recommended for either the project or enterprise service architecture is to start with an intensive session, between 1 and 3 days depending on the scale of the problem. This event needs to be well prepared with all of the required stakeholders in place, and information available on which to make decisions. Mostly what is needed however is the people who work inside, and outside the company who understand how the business, or project, functions. Creating the service architecture is about creating the broad sweeping pictures, not the detail within those images, as such ensure you don't get bogged down with a point of detail on a specific flow, as long as everyone agrees it goes from "Service A to Service B".

A properly facilitated event will create an invaluable resource that will deliver its benefit many times over. And by getting everyone working together you ensure that everyone is agreed on what the big picture is. This picture gives you the map for the enterprise or project and enables further analysis and detail to be created, while ensuring that these efforts are not impacting each other. If an organisation is serious about Service Architecture it will need to either partner with an organisation that runs these sorts of events, or create the infrastructure itself. There are a few key elements for any collaborative event:

1) Physical proximity – people have to be AT the event, its no good trying to do it over email
2) Permanent Attendance - people should be there for the whole event
3) No interruptions – mobile phones off
4) Dynamic working – whiteboards not word documents
5) Capture Everything and make it available – Digital photos and load onto a Wiki
6) No diversions – no side meetings, no dead-ends
7) Focus – Strict time-keeping and focus on achieving the goal of delivery

The objective of the event is to create the final product, namely the outline service architecture, not to be the basis for a series of meetings and word documents. The event must therefore be driven to make decisions.

Once the decision has been made for the first time the architecture becomes a living artefact that needs a regular review. Reviews should be initiated as part of normal business and technology change procedures, it should become enshrined in these procedures to first consider the impact on the service architecture. In the absence of such a review it is recommended that the follow minimum review periods be adopted.

- Level 0 this should be done once a year,
- Level 1+ once a quarter

A review both as part of a business or technical change, or via a standard review period, should not attempt to re-create the whole architecture but focus on those changes that may impact parts of the architecture, the objective in these reviews should be to understand the change, and to reflect those *changes* in the architecture.

These created artefacts must therefore be available not via word documents but via a collaborative environment which people can quickly access, and potentially augment. Without an environment in which the big picture is created, agreed upon and made visible a service architecture will become yet another piece of “shelfware” that means nothing to anyone. The approaches,

and terms, used in the Service Architecture must be driven into other areas, for instance the Enterprise and Business Architectures, to ensure there are solutions that can be completely traced.

Once you have done this collaborative working there is no reason to disappear and keep working in isolation. When looking at how you implement services do look at creating collaborative teams where the people with the requirements work closely with the design and development teams.

Managing Change

With a properly implemented Service Architecture change becomes easier to manage. Because the services map to the business the change tends to be limited within those bounds, and where requests go across bounds it is a clear demonstration as to why it can be more complicated. If you are finding that change is being requested in different ways to the service architecture this is a good indicator that the service architecture needs to be revisited. Organisations should look to monitor change, business processes and organisational barriers against the service architecture to identify where it needs to change. A key benefit of Service Architecture is that aims to define what the organisation wants to do, this means it will be continually changing, but within known bounds, based on the goals and aspirations of the business. The Service Architecture must not be seen as a static thing or it will solidify and become yet another monolithic IT solution.

Another advantage of service architectures, when properly delivered, is that services can be maintained and released at different heartbeats, rather than having large maintenance cycles into which everything can fit. This again requires an increase in best practice over what many organisations do and requires robust automated testing and quality control.

Coping With Major Change

If a review identifies fundamental changes in the way an enterprise operates, most often due to a large scale acquisition, merger, disposal or outsource, then a full architecture review should be undertaken and that could potentially result in significant changes to the architecture. This should be enshrined within these business change programmes, and indeed form the basis for how the IT organisation is to be guided and directed by the business. The intention of such a review within a major change programme is to ensure that programme's effective implementation; it should therefore run in parallel with such efforts. Comparing the established service architectures of organisations that are attempting an acquisition or merger is a good way of noting the compatibility of those organisations. While they may externally exhibit the same functions to the market a service architecture comparison can identify potentially serious discrepancies in how the organisations approach that common task. This matching of service architecture also represents a good way of identifying areas of commonality that can be turned into single shared services to deliver the expected cost savings. So while on paper both Walmart and Harrods both aim to "sell everything" they clearly have different objectives, drivers and business models, these would be seen in the different emphasis and drivers within their service architectures.

Language To Determine Services

When determining a service architecture a useful technique is to have people imagine they are looking at the enterprise, department or project from the outside. At this level what do they see? What a Service Architecture should be driving them towards is identifying the types of work that are being undertaken. "*What does it do?"* is the driving question at this level, the objective is to understand the form of the services rather than their details, so as the discussion delves down towards "well I ask for a paper clip and they give me a quote, then I submit the purchase request and they….." it is important

to ask for a term that groups all of those elements together, ask "So what would you call that type of function in your company?", or "and as a group what are they know as?". It is critical not to get bogged down in the process elements and to be thinking in the service architecture purely of the groupings. Once the primary groupings have been defined you can then work on the capabilities, again doing these at a high level, the intents, rather than the specific elements that are undertaken.

6

Creating A Service Architecture

In order to explain how a Service Architecture is created two different scenarios will be used. The first will be to describe the entire service architecture for an organisation, the second is for a specific project within that organisation. The organisation "Oblivion Widgets Inc" manufactures widgets, and the project is to enable vendor managed inventory (VMI) of its stock. The following approach assumes an optimal way of working, namely that you can get everyone together at a facilitated event, if you can't do that the deliverables will still be required they will just take longer to create and agree.

Template Of A Service Definition

The following is an example of the sort of information that needs to be captured during the service definition. As can be seen from this example its relatively high-level and concentrates on the drivers for requirements rather than the full detail. This is done deliberately to avoid going deep to early, and to ensure that the broad communication elements are clear to everyone. By identifying the different, often competing priorities, of the key stakeholders and actors it helps to identify the actually important elements, and the potential issues that could arise.

Logistics and Warehouse

Description

The Logistics and Warehouse Service is concerned with the allocation of products to vendors and the replenishment of warehouse stock to meet anticipated demand.

Business Owner(s)

The business owners are the people who will ultimately drive the strategy and decisions for the service. While these people may delegate authority for elements they will remain the final point for any issues to be resolved.

Table 2 Logistics and Warehouse Business Owners

Name	*Role*	*Description*	*Priority*
Brian	Head of Supply Chain	Responsible for all of the business and technical elements related to the supply chain infrastructure, including Warehouses, Logistics and stock forecasting.	Minimising of stock while meeting customer SLAs.

Actors

The Actors are those organisations, people and services which interact with this service, they provide it with its external drivers and priorities. There is no differentiation between actor types at this level as it doesn't matter to the service whether an external interaction is directly from a system or via an interaction of a person with this service.

Table 3 Logistics and Warehouse – Actors

Name	*Role*	*Description*	*Priority*
Customer	Organisation that buys, and potentially sells on, widgets	Typically medium to high scale retailing operations	Stock availability and low prices
Supplier	Manufacture or Wholesaler of component parts needed to make widgets	Typically Asia Pacific companies for large bulk elements and some small European companies for specialisation	High Demand, prompt payment
Logistics Company	Used when the standard supply chain cannot cope with local demand, or to supply to global markets	A combination of local haulage firms and international shipping and logistics companies	Demand, load size.

Primary Capabilities

The primary capabilities are the key facilities that the service offers, these are the tasks and their real-world effects. At this stage what is captured along with the task itself are the contractual elements of the task. These consist of three definitions

1. Pre-condition – These represent an obligation on the consumer, these elements must be true before the service is called
2. Post-condition – These represent an obligation on the producer, these elements must be true after the service has been called
3. Invariant – These represent an obligation on the producer, these elements must not have been changed by the producer.

Table 4 Logistics and Warehouse – Capabilities

Capability	*Description*	*Pre-condition*	*Post-condition*	*Invariant*
Ship Order	Request for an order to be shipped.	Goods on the order are available	Order is assembled ready for delivery, delivery is scheduled with the client.	Price of Goods
Add to Stock	Add new items into stock	Item is available and is a valid and known product	Items are added to the stock count	Stock of items un-related to the new items
Deliver	Request an external company, or internal logistics, to supply an order to a customer	Goods available for shipping	Goods removed from warehouse and placed on appropriate delivery route	Items on order
Supply	Receive supplies	Order had been placed with supplier, or supplier is managing own inventory	Inventory of supplied item is increased by amount supplied	Stock of items not related to this supply order

Stage 0 – Pre-work

Before engaging on a services architecture it is important to have a broad understanding of the area for which the task is being under-taken. If undertaking the task at an enterprise level this

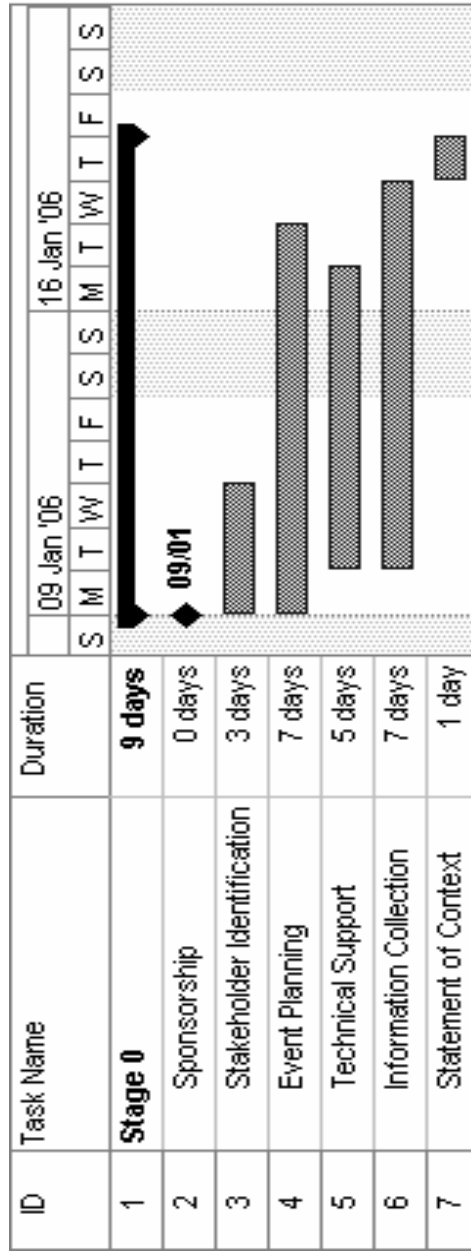
should include the external drivers on the company as well as an understanding of the sector it is in. If it is a specific project or programme then it is key to understand the primary drivers for the project. The result of this pre-work should be for the analyst, architect, consultant or manager to understand the external drivers for the architecture that is being created. A key element to start at this stage is the glossary of terms, so certain elements can be quickly referenced when required.
At this stage it is also important to identify the key stakeholders who are required to create the Level 0 and Level 1 service architecture, and plan the event to which they will be brought in order to create the architecture. This event must be properly facilitated with all information gathered at the event collected. During the Pre-work the initial collaborative tools for sharing the outputs of the work should also be created, this may change over time but a lightweight first attempt is a necessity to ensure the architecture becomes a living thing rather than a musty document.

An example project plan for this is shown in Figure 5, this is typical of the planning that needs to go into a services architecture event for a reasonably sized project. This would be sufficient preparation work for the sort of project that is expect to take between 9 and 12 months in duration and involve minimal external interactions within its service architecture.

Deliverable	Description	Enterprise Duration	Project Duration
Sponsorship	Without clear executive sponsorship for the effort a Service Architecture is doomed to fail.	N/A if this isn't achieved Stage 0 hasn't been reached.	N/A if this isn't achieved Stage 0 hasn't been reached.
Stakeholder Identification	The key stakeholders from the business, technology and ideally external interactions need to be identified and contacted for participation in the event.	3 days to 2 weeks depending on scale and external involvement.	Normally takes around 3 days to identify the people.
Event Planning	The Service Architecture event needs to be scoped and planned. This requires a certain amount of pre-planning to ensure any information required at the actual event will be available. This also involves managing the stakeholder attendance and flagging any issues with non-attendance.	2 weeks to 6 depending on the scale and number of participants.	Between 1 and 3 weeks depending on the scale of the project and the logistics involved with getting everyone together.
Technical Support	The collaborative tools and information availability to support the event	1 week to set up	1 week to set up
Information Collection	Gathering of the specific information to support the event	2-6 weeks depending on the depth and amount required	1 to 3 weeks depending on the depth and amount required.
Statement of Context	The mission statement for the event	2 days with event sponsor	1 day with event sponsor.

Table 5 Stage 0 Deliverables

Figure 5 Example Stage 0 project plan



At this stage the intention should be to keep the team small at this stage this normally means

- Project Manager for delivery
- Lead Architect
- Lead Business Analyst
- Administrative assistance (P/T)
- Technical Support (P/T)
- Event co-ordinator (P/T)

In an organisation set up to do these events on a regular basis it is normal for the last three roles are part of the physical event venue team.

For a large enterprise definition exercise the key driver in duration is the number of stakeholders who have to be corralled together, this is where clear executive sponsorship with the ability to pull people into line is critical. The team to do the work should not be much greater than that for a normal project, normally there is an addition of one analyst and a supporting architect if the level of work is particularly large. The key at this stage is that it is about comprehension and not about gathering of requirements. Operational Research groups are of most use during an enterprise definition stage 0 to obtain the required level of relevant information.

At this stage there is no service architecture.

The Event

The objective of the event is to get all of the stakeholders to agree on their position. This means it must be run as a collaborative event with all stakeholders encouraged to voice their opinions. A properly planned event, and the supporting infrastructure, should not be mistaken for a "workshop". The intention here is to produce a specific deliverable, paper is not required, whiteboards are. Tables should only be used for breakout sessions and the event should be scheduled to run 100% of the time during the day with moving refreshments

available all the time. The event venue infrastructure must have access to the internet, all elements must be captured and available via a collaborative site by the end of each day. Some basic rules must also apply:

1. No mobiles
2. No email
3. Timetable at the start of each day is fixed, end means end.

The Facilitator

At the event facilitation is key to the successful creation of the architecture. The facilitator requires the following skills:

- Strong Communication skills
- Strong listening skills
- A broad understanding of the business domain
- The ability to co-ordinate teams
- The ability to end discussions with an agreement

It should be noted that none of these are technical skills, and indeed it is the case that service architecture can be created without having an architect undertaking the facilitation role. A business consultant or business manager is as equally well equipped to undertake this process if they have the right level of understanding. It is most normal however that this be an Enterprise Architect who can bridge the gap between the Business and IT functions and work with the various different elements on implementation and strategy.

The final and most important skill in the facilitator is that *they have done it before*. This means that a facilitator should not be undertaking an enterprise wide service architecture discovery if they have not previously done the task either on large programmes or on organisations of similar or slightly smaller size and complexity. A facilitator for a project should have previously undertaken a similar role, or been an assistant in a large programme or enterprise wide mapping exercise.

When determining what is a service and what isn't, there is no substitute for experience.

Level 0

The key when considering architecture at Level 0 is that each of these services must be core and central to the actual *business* being considered. For this reason support services, which may have large departments, would not be considered as Level 0 services. At Level 0 the important element to consider is that each Level 0 service could potentially be considered as a service in its own right, often something that could be considered as business in its own right. For an organisation this often means that the model reflects areas that could be either outsourced, sold or partnered. And for a project it often represents the different business objectives that the system has. The other key element in deciding what the Level 0 services are is that *combining level 0 services into larger domains would not reduce the high-level clarity of the system.* As a rule of thumb the number of Level 0 services should be between 1 and 5, but certainly no more than 10.

Defining What

The focus for level zero is the same question as for all levels "What is it?" the ambition here is first to get the high-level picture. The first, and most important element, is to determine what the actual services are, without worrying about the interactions. At this stage the most valuable tool is a white board as there may be many candidates for Level 0 services depending on either the perspective or opinion. The target is a clear statement of *just* the Level 0 services.

The level 0 picture in
Figure **6** represents a powerful picture for an enterprise, it shows clearly *what* the organisation is about. For a project the type of Level 0 diagram shown in
Figure 7 can sometimes be compelling but more often the next level of refinement is need to really explain what the *project* as

opposed to the services it delivers is about. This is because a project has explicit drivers that provide context to the services, while an enterprise *is* the services themselves.

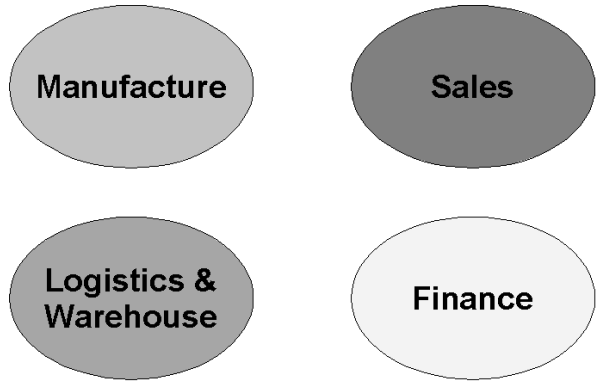


Figure 6 Initial Enterprise Level 0

For the Oblivion Widgets Inc Enterprise the first element is going to be the core Services themselves. This just describes the "what" of the enterprise, and absolutely no more. It is critical that a business agrees on this diagram before attempting anything else, because if the top can't be agreed upon there are some significant differences in how the organisation perceives itself. For Oblivion Widgets Inc that diagram is shown in
Figure **6** and it represents the four key areas that define what the business is about.

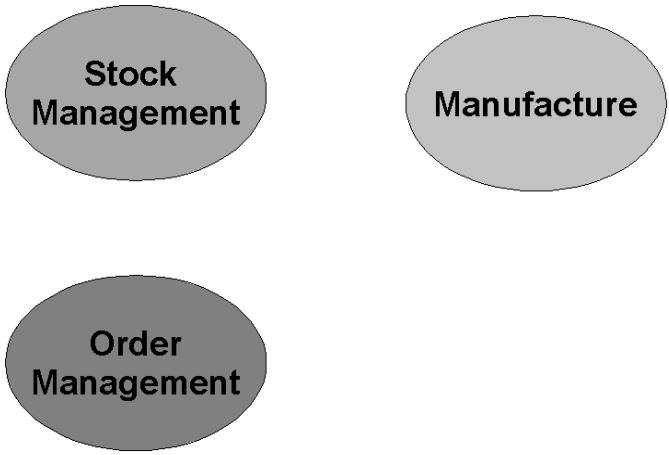


Figure 7 Initial Project Level 0

For a project the approach the need for agreement is no different. The question is "what is it about" and the answer the key domains that need to work together to solve the problem. Figure 7 shows this minimal set of services required for the vendor managed inventory project. It's important to note on a project basis that the driver is still what are the business focused on rather than project focused services that are required. The services created must fit within the context of the overall business and not just be specific to the project.

Specifying "who"

The next elements to be added to the diagram are the external actors which round out the domain under consideration and represent the core features within the Level 0 model. The key here is that the actors are *external* to the services, rather than being the internal objectives of those services.

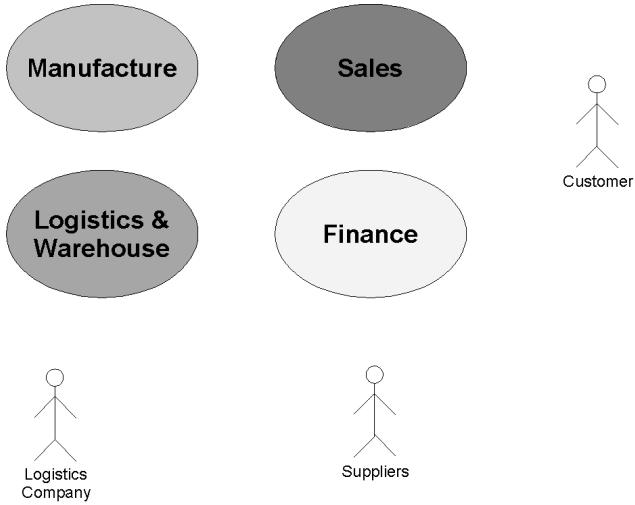


Figure 8 Enterprise Level 0 with Actors

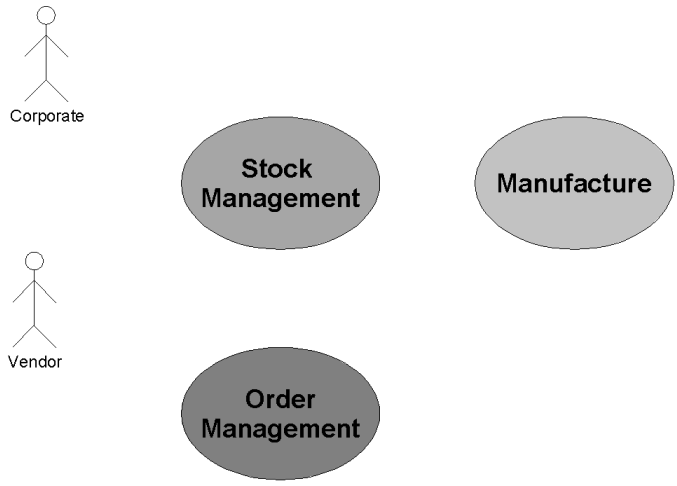


Figure 9 Project Level 0 with Actors

For an enterprise this (Figure 8) expands on the "what" of the enterprise to talk about the "with whom". This can be a powerful tool to help and enterprises determine the scope and importance of partners. If they are not important at Level 0 they should probably not be considered strategic. For Projects (

Figure **9**) the impact can be less as although it defines the totality of the core features for the project it doesn't at this stage define the full "why" of the project. It is therefore important to understand at what point the diagrams become truly useful. For our Enterprise it was reached at stage 1, for our project we have to normally have to complete the picture.

Adding the "why"

The next element in creating the Level 0 picture is to understand *why* various services and actors interact. This is the key question for the facilitator to pose at this stage, "does service X interact with service Y and if so why does it interact". The aim here is to start defining the value contract for the service that it offers to the world. This then delivers for the enterprise, Figure 10, a full definition of both what the organisation does, and why it does it, and for the project,
Figure 11, completes the picture as to why the project is being under-taken.

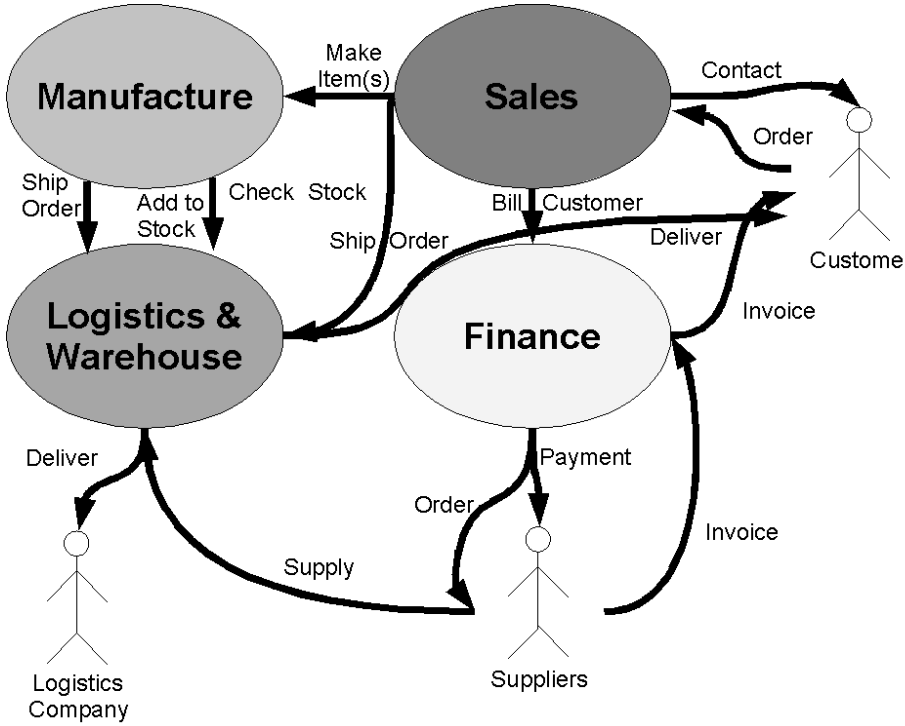


Figure 10 Enterprise Level 0

During this the facilitator must continue to consider which single diagram will best convey to all stakeholders most powerfully what the service architecture is about. For a project it is normally the element at this stage of refinement, but often for an enterprise the simple "what" diagram can often be the most effective due to its absolute simplicity.

Fleshing out the services

Once the full diagram has been defined its time to fill in the information required to describe the service. The sort of information required is described in the Template Of A Service Definition section. It is recommended at this stage that break-out groups are created to focus on each of the services; these should include people from that service domain, and people from the services that interact with that service. These groups should then

report back the definitions for a final confirmation by the whole group.

Level 0 Deliverables

At this stage the following elements should have been created

Table 6 Level 0 Event Deliverables

Deliverable	Description	Enterprise Duration	Project Duration
Service Diagram	Diagram of the services at this level	½ day	2 hours
Actors	Identification of the key external actors	2 hours	1 hour
Service/Actor Interactions	Identification of the interactions between services and actors	2 hours	1 hour
Service Definition	Definition of the services and their interactions in more detail	4 hours	2 hours
Report out	Confirmation of all findings	30 minutes	15 minutes

Enterprise Level 0 model

For the enterprise the approach to understanding the services in the Level 0 model is to understand **what** the key business areas are that make up that enterprise. With Oblivion Widgets these can be split into four key areas, Sales, Manufacture, Logistics & Warehouse and Finance, these represent the central elements of the business. The key actors at this stage are those that interact with the services externally, not the internal actors who provide that service. The question therefore is “what is it?”.

Figure 10 shows the level 0 model for Oblivion Widget with the key external actors. It also details the primary interactions that those actors have with the services. These are **not** individual functions, but descriptions of the capabilities and effects of the interaction. On this diagram there is no description of the process of the interaction, only the start, or end, of an

interaction. This picture is deliberately simple, and should be kept so. Its intention is to act as the reference point for all initial decisions within the organisation and as a reference point to which every stakeholder, business, IT, internal or external can agree. These boundaries also mark the areas within which change will be managed and constrained.

Project Level 0 model

Within a specific project exactly the same approach is taken, this time at a much lower level, if an organisation has undertaken an enterprise Service decomposition then its important for this to be used as the base for the project, not only does it represent a head-start on the process but also ensures that the required stakeholders are aware of the project and its impacts, and of course that the project doesn't start re-inventing the wheel and build its own "enterprise" service model. The first element is to understand what is required of the project as a simple statement. The objective in this example is to minimise the amount of stock carried while maintaining, or reducing, the amount of out-of-stock issues that customers have.

The project level 0 (Figure 11) is equally simple, and hopefully equally powerful, the intention is to understand the objective of the project from a simple diagram and the domains which the project covers. From a top level the objective of the project is therefore simple, the organisation (Corporate actor) wishes to minimise the amount of stock required, and the vendor is now responsible for managing their own stock levels. Flows which are not directly changed in the project scope, e.g. ship against order. Are excluded at this level.

In order to demonstrate the purpose and options at this stage it is recommended to create a number of business activity diagrams, these should be seen not as formal process definitions, but as sketches that represent either the different options or the different scenarios.

Figure 12 and Figure 13 show two such diagrams. These are very high level and are used to demonstrate how a polling (former) or event (later) approach to VMI would work

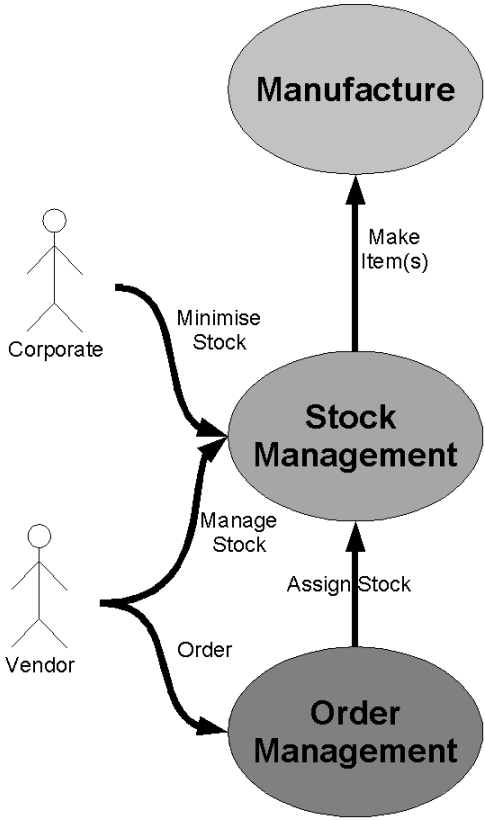


Figure 11 Project Level 0

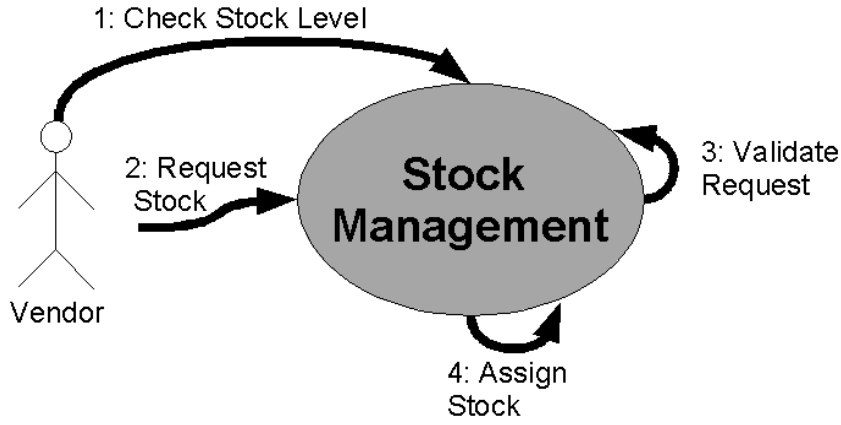


Figure 12 Standard Business Activity

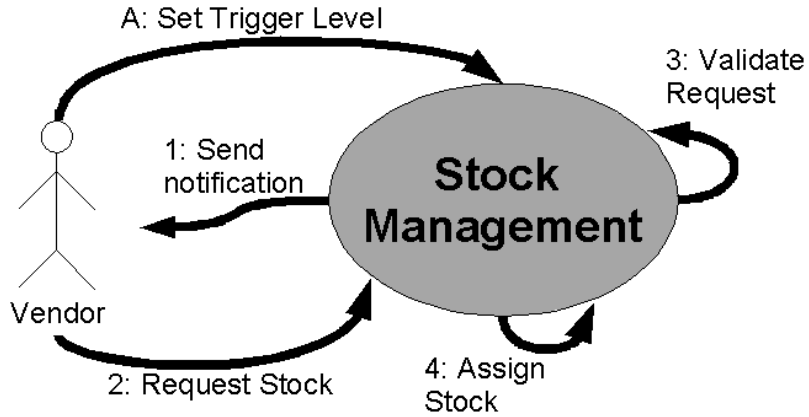


Figure 13 Asynchronous Business Activity

The objective here is simplicity, **not** to go into the detail. Level 0 should be kept deliberately abstract and be clearly understood by all stakeholders. If understanding of the domain is lost at this stage there is little hope that the project will succeed. It is normal that the activity diagrams used here are revisited in detail by either the business process or architecture streams of a project or programme. It is important to retain the *context* described at this level, even if the full content is much more detailed.

Drilling down and creating structure

A basic principle of this SOA methodology is that structure is a good thing, and that the context of a service can change the way it operates. As mentioned earlier having structure helps you to provide a clear mechanism for rippling change down and understanding how change will happen. The other element that structure helps with is comprehension. Having a "Procurement" service that lives in isolation to Finance makes little, indeed arguably no, sense as you would have to describe the whole finance function to understand why you can't just go out and buy a pen. Understanding that the reason that Finance has established Procurement is to minimise the amount of purchasing done and to standardise the way in which purchasing is done will be key to creating a valid procurement service.

Modelling Services without structure assumes completely autonomous and atomic services, these will therefore become fine grained and inherently useless in the longer term. This approach is often promoted by people wishing to take a process oriented approach, with the steps become ever smaller and often consisting of single activities.

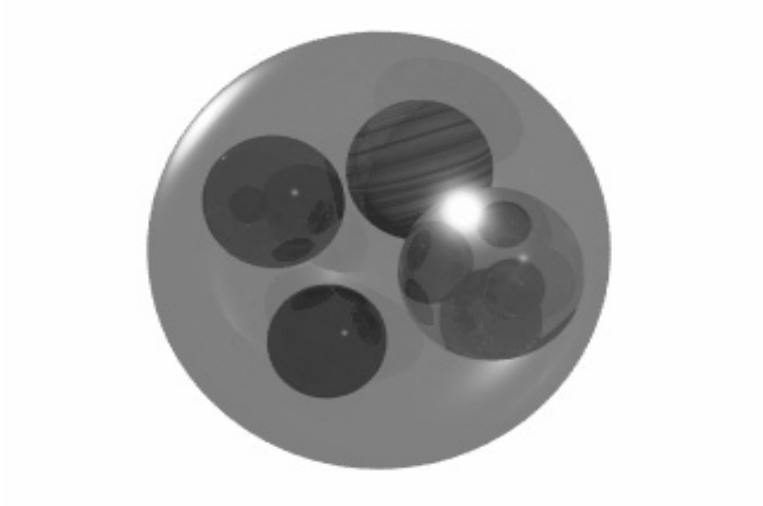


Figure 14 Services nested inside services

As a result are several elements that need to be borne in mind when considering decomposition. The first is that the enclosing service confers behaviour and management onto those at a lower level. This means that while services from two distinct domains can interact, they have to do so in a manner that enforces the surrounding, as well as service specific, contract. This inheritance of principles, contracts and other functional and non-functional attributes is central to a well constructed model. Thus rather than thinking of a circle on a circle, a better mental model is to think about nested spheres (Figure 14).

The second consideration is that the purpose of this classification is to enable simple navigation of the service architecture to ensure it can be understood and future requirements can be easily classified within the service models. The purpose here is to avoid the "death by availability" challenge that comes with too flat a structure. Without a defined structure and governance model it becomes impossible to manage the complexity of the enterprise (Figure 15) Its this problem that SOA tries to address so its important that any Service model is designed to aid in comprehension, not to stress the complexity of the environment.

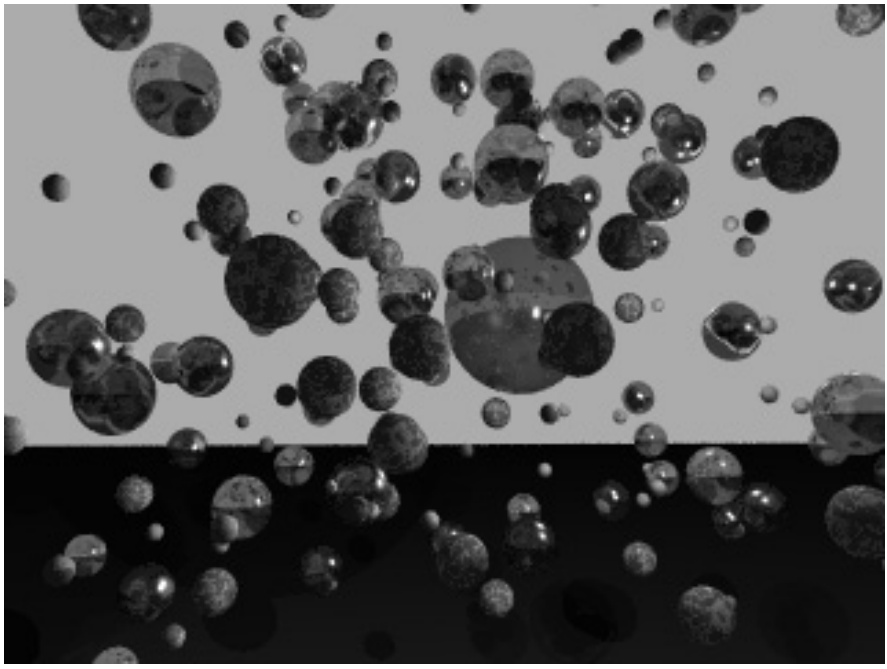


Figure 15 Too many Services, too little hierarchy

Creating The Next Level

Drilling down from the Level 0 into the lower level elements is just a series of repetitions of the steps described for creating Level 0 and creating the same deliverables, the difference is that it is applied to the lower level. It is advised that teams be split at this stage to help increase the rate at which services are defined. A team that is decomposing a service should have

- The primary stakeholder or business owner of that area (identified in the service definition)
- A representative from all groups that interact with that service
- A facilitator
- Representatives from the business and technical community of that domain

Teams should then report back their results to the whole group. The timetable should be expected to be similar to those outlined

for Level 0, which means an event scheduled over 3 days at an enterprise level should only expect to reach level 2 at best.

If there are impacts on the Level 0 model these are dealt with immediately, the main facilitator and the main sponsor act as the veto on changes to the Level 0 model and it is important to ensure that teams do not "cry wolf". One successful way of doing this is to apply a similar system to that used in the National Football League[13] in that the teams have a number of challenges. A team in this case is given a number of challenges, normally 2 or 3, and if their challenge is accepted they keep that number, if a challenge is unsuccessful they lose one of their challenge rights, if a team loses all of its rights it **cannot** challenge the Level 0 diagram. Another option is to use a variation on the "sin-bin" approach from Rugby, Field Hockey and Ice Hockey, here on an unsuccessful challenge the team is excluded from making a challenge for a given period of minutes, normally between 5 and 10.

Enterprise Level 1

Rather than decompose all of the Level 0 services, which would create a much bigger but more repetitive book, only one service has been chosen, namely manufacturing. The principles remain the same no matter what service is chosen.

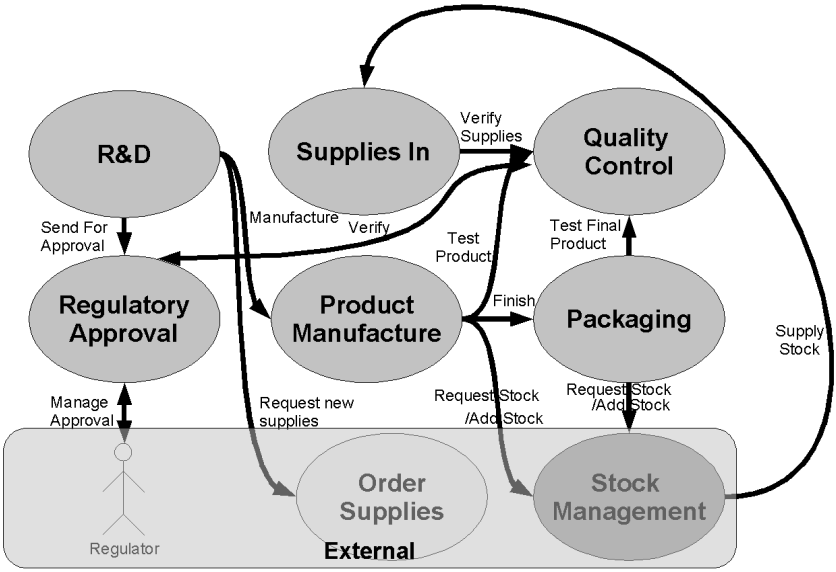


Figure 16 Enterprise Level 1 for manufacturing

The Level 1 for manufacturing splits down Oblivion into six distinct areas. Thanks to the centralisation of purchasing around stock management the only area that needs to make direct purchasing requests is R&D as they use newer materials than are currently being used. A key part of this level 1 diagram is that it identifies services from other areas with which it interacts. This is why some form of common tooling can help in creating the service model so changes in terms are instantly reflected across the various instances of that service.

An enterprise Level 1 sometimes matches the operating departments within a division, or at least a potential set of operating departments. It is risky to attempt a one-to-one mapping of services to departments as these can be subject to change, while the organisational functions they undertake will remain relatively static. The important difference of a service is that it represents "what" while the department often represents "how" . This is quite often the stage at which more discussion is required as the key is to understand the key services that the

division is offering, rather than just repeating that "Frank runs both Packaging and R&D", if the two elements have a clearly distinct purpose then they should be represented as separate services. Again logical groups should be represented as one service and further decomposed as required.

Project Level 1

A project level 1 is often the stage at which the requirements become more apparent for the system, where the Level 0 (Figure 11) details the intent of the project and sets it context, the Level 1 indicates the actual work to be undertaken and explains how it will work.

Figure 17 applies the same principles used to create Figure 16, only this time applied to the specific requirements of the project. Again the external interactions are reported, and the major interactions within the service model. Level 1 often refers to the major IT components of a delivered system, these can be as large as whole packaged solutions, for instance Logistics Management in this case, or bespoke areas of targeted development, for example forecasting. It is within Services at this level that the requirements are often initially captured. For small projects a Level 1 de-composition will be all that is required. Level 1 can also be more complex than level 0, but a maximum of 8 Level 1 services for each Level 0, with a normal amount being around 4, should be used as a guide.

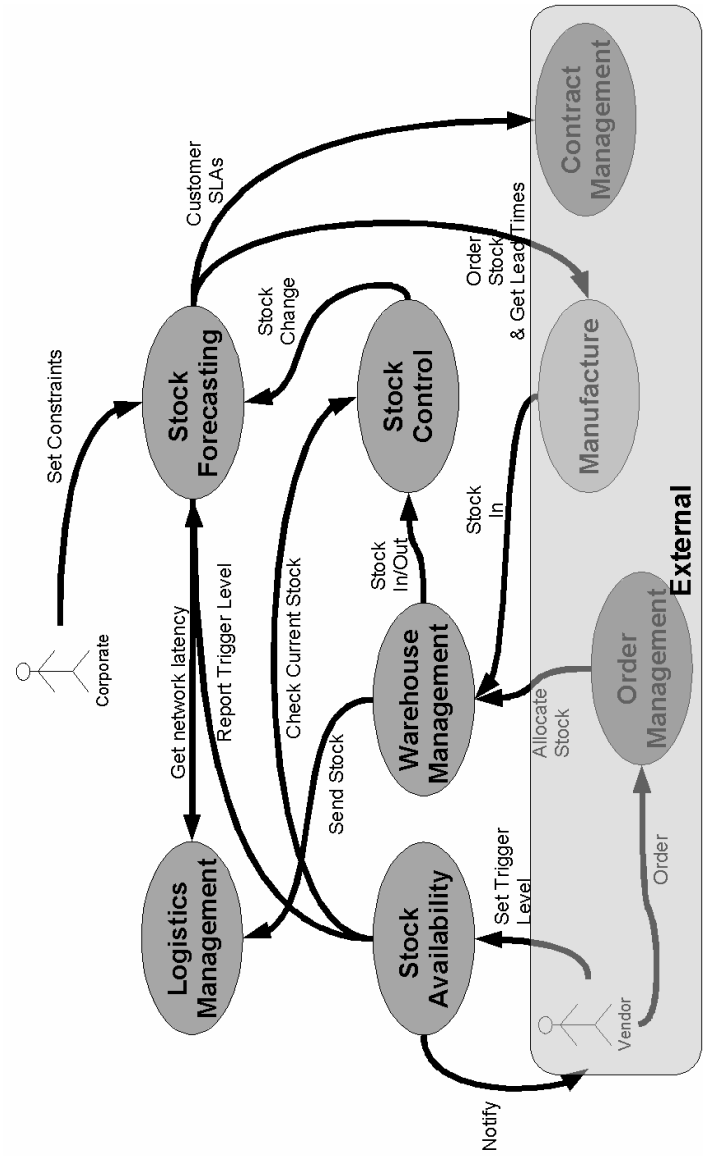


Figure 17 Project Level 1 for Stock Management

Further refinement

Further refinement can be driven by two different objectives. The first objective is to delve deeper and understand the problem

domain more. This approach involves recursively decomposing services until you reach a sufficient level of definition. This uses the same approach defined for Level 0 to Level 1 decomposition and can then produce as all the levels of decomposition that might be required. The other refinement is, for a given service, to focus on the different external representations that it may have. This is used for Services, and collections of Services, which have a number of external consumers which interact with a common set of functionality in differing ways. This should not be used where a service just has multiple actors who call differing functionality, for example a purchasing service where some people buy, some price, and others approve, these functions make up common areas defined by roles within one services.

It is quite normal to have different parts of the service architecture driven by different objectives, so for Oblivion Widgets for example there might be a detailed decomposition into Manufacturing with the aim of completely understanding those elements which have not yet been automated. Logistics might decide to focus on the external representations of its forecasting and inventory management services to understand clearly the different, and potentially conflicting, drivers and objectives that the services have.

Understanding when the Big Picture is done

A key part of facilitating the event is to prevent the group going into details and trying to create complex process maps to connect the services. This is very much a secondary task, as is the question of "Where does my SAP system fit in this" or "We use .NET". The deliverables defined in this section are *all* that is required. The purpose of this event is to create a simple diagram that everyone can agree with. The more detail that is created the less clarity there will be in that diagram. The facilitator needs to drive towards this big picture and to judge the level of decomposition that is required. This level may be different depending on the part of the "tree" that it is in.

Publishing the big picture

Once the big picture has been created it needs to be published so everyone can see it. Make sure there is access to a large scale printer (A1 and potentially bigger) to print out the diagram on a single sheet. Multiple A3 sheets can do the job but its cleaner on one piece of paper. Computer monitors tend not to have either the size or resolution to display a full diagram in one image. The challenge at this stage is one of representation and information. We have used ovals to represent Services throughout this document as when conceptualising a framework they can present a more abstract notion than a more regimented regular shape. Ovals however are poor when it comes to representing nesting and the level of services that could exist at this level.

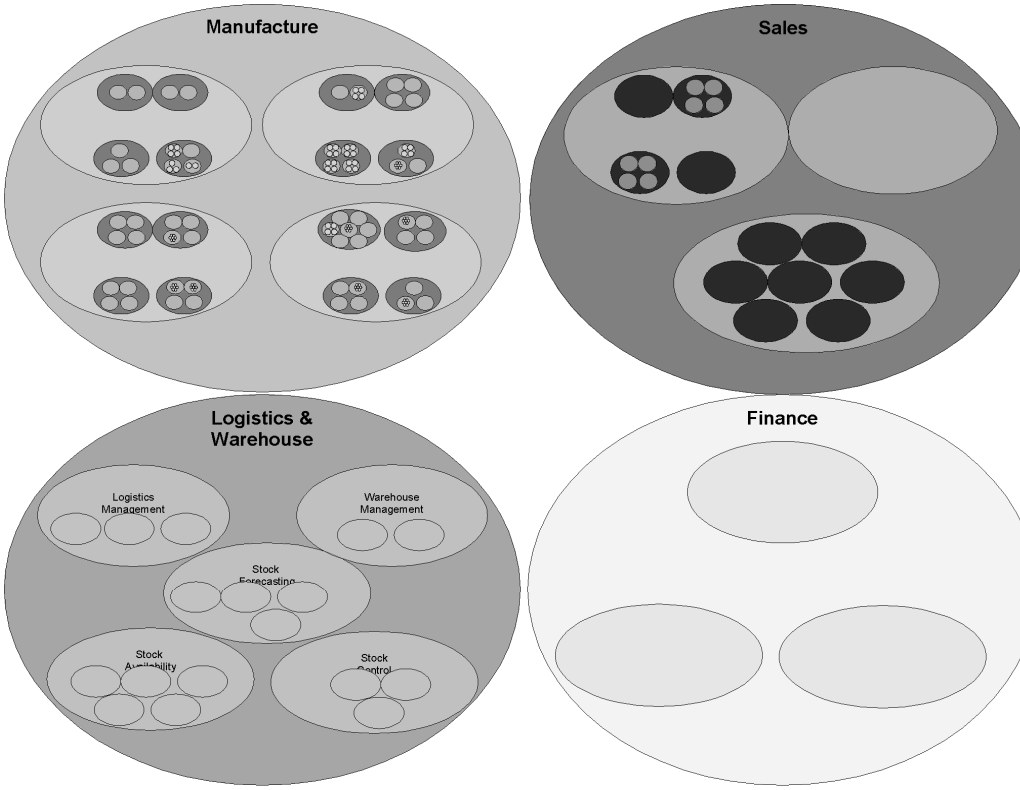


Figure 18 "Big Picture" in Ovals

Figure 18 shows the challenge. Simply put Ovals just don't tessellate. So putting text into the various boxes becomes harder and harder, but not impossible. If it is possible to represent the diagram in this way then that should be done, but for large enterprise and programme models it sometimes becomes necessary to move towards a simpler shape, the rectangle. This shift could also be done if there is a desire to represent the architecture not as something constant evolving and under review, but as a representation of what shall be. By using a more regimented shape it helps add formality to the same information.

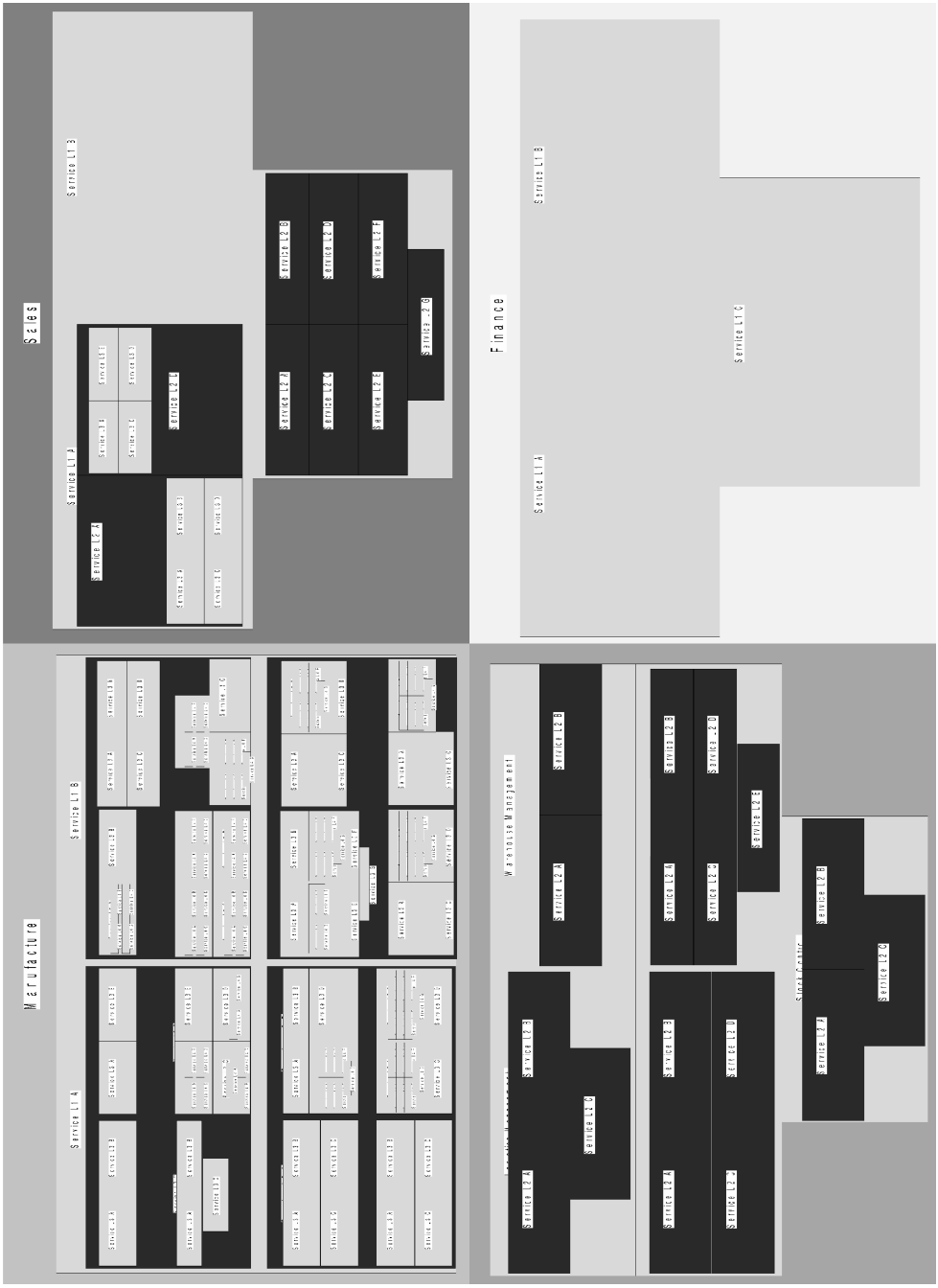

Sales
Finance
Manufacturing
Warehouse Management

Figure 19 has the same information but displayed in the different format. The key here is that people should be able to quickly walk up to a wall with the A1/A0 picture on and pin-point the area they need to consider. This is the picture that should be on every wall of every person that needs to know how things work together. Apart from the static images there should also be a website that guides people down the architecture. Standard organisational tools, such as the one in Microsoft Visio, can be used for this model to create a simple drill down and navigation mechanism, or a straight forward Wiki.

Ideally there should be a tool that supports this approach, but as yet none in the market appear to be mature enough to provide the functionality required for this first diagram, let alone the refinements that come next. So far the most effective tool found in this area has been the use of a Wiki to capture and link the various artefacts required, this is better than nothing but hardly ensures rigour and direct impact on later stages. This first diagram is the big picture of the project, programme or enterprise, that gives context for all decisions. However the support services and identification of shared services has still to be completed.

End of the Event and next steps

The event has one further task beyond the creation of the diagram and the detail on the services. This is to define the timescales and next steps. These next steps can differ depending on the context and scale of the undertaking. For an enterprise model the key next element could be to identify the high level business processes that work between the services, for a project it might be to identify the current technology that is delivering some of the services. The critical element however is to keep the momentum high. Table 7 shows some potential next steps, this list is not exhaustive, some of which could be actioned during the final session.

Title	Description	Project or Enterprise
Business Process Map	Co-ordinating the various services into high level business process maps	Enterprise
Execution Process Map	Detailing how services are to be orchestrated via technology to deliver business processes	Project
Support Services	Identification of the support services (all categories) that flesh out the architecture	Both
Service Specification	Detailed specification of the contract for a set of services.	Project
Project Identification	Using the Service Map, allied to business strategy, to understand what the most appropriate first project will be.	Enterprise
Communication Plan	How to propagate the findings beyond the event group	Both
Change Process	Definition of how revisions will be made to the Service Architecture	Both
Visibility	Technical support to the Communication plan, and the collaborative and interactive tools to navigate the service architecture	Both
Service Classification	Classification of the services by various means, including business value, risk, potential to change.	Both
Enterprise Architecture	Creation of the Enterprise Architecture that will deliver the Service Architecture, this often covers the IT Strategy and technology roadmaps of the organisation	Enterprise
Solution Architecture	How a specific project will be delivered and the fully fleshed out architecture for its delivery	Project

Table 7 Potential Next Steps

It is important that after the session that the full findings are made available as quickly as possible. At most 2 working days after the end of the session, this is done to ensure the information is close at hand as soon as people start talking about what they did.

The collaborative information site that has been created to support the service architecture needs to be kept up to date. And as the support services are added they need to be added to the overall "big-picture" creating a new image that represents the "whole picture". The big picture still needs to be kept as it was as it represents the business view of the system, but it is critical that particularly on the technology side there is a view of all the potential services. Quite clearly this can be a huge picture on a complex project or enterprise.

At this stage the service architecture has first been exposed, it is now possible to attack some of the other tasks from the perspective of these services.

Guidelines

The following are guidelines only for the number of services you should expect at each level, these are not hard and fast rules, but are elements that you should take into account when reviewing your service architecture.

1. Services – Each level should have between 1 and 8 services, ideal numbers are around 4
2. Capabilities – Anything less than 3 should be a concern, anything over 10 should be a worry.

"Get" is not a Service

A common mistake when defining services is to have "services" called "GetCustomer" "UpdateCustomer" etc, these are **not** services, they are just invocation points "The why" elements on the services. The service is "Customer", which has an

interaction point of "Get". Services are collections of processes, not individual processes themselves, hence the reason that "Get" is not a service. To help with this thinking consider about an old, manual organisation, they would use paper forms (with many copies) on the basis of which many different tasks were done. For example: a requisition was raised by writing the basic information on the form. The form was then sent to the order dept for processing, your manager and his assistant for filing, to finance for their records and to match up when invoices came in. These multiple copies of the forms are then acted upon in different ways by these departments. And so on…

This "hand over" between function is the ambition of a service architecture, while finance on this occasion did "receive requisition" the organisational function is finance, and its task was "receive requisition", the "what" is finance, the "who" is the supplier, the "why" is "match requisition to delivery", only when we consider the "how" do we worry about the actual physical handing over of the form.

Back to the definition of service from the OASIS SOA RM, "Get" is a capability which is accessed via a service.

7

Completing the Service Architecture

The previous section detailed how to create the business service architecture, which gives the view as to what the business services should be, what it doesn't tell you is what should be built. To deliver these services to the world will mean exposing them in different ways, creating technical underpinnings for these services and understanding where services can be re-used rather than re-developed. If the business service architecture has been done at only a high-level then this stage will probably be a augmentation of the service architecture.

It is important to remember at this stage that you are not looking to modify the core pictures created, but to create additional representations for new services and to create additional links between services that denote common or shared behaviours. Using collaborative technologies, even as simple as a Wiki, helps you to map out these elements from the context of the overall business service architecture rather than having to pollute the business service architecture and by cluttering it reduce its effectiveness.

Identifying Shared, Technical And Virtual Services

Once you have created the business service architecture a good next step is to start identifying how services will be exposed, where there is functionality that can be shared, and what are the underlying technical elements that need to be in place in order to deliver the full service architecture. When you are delivering a

project these are the elements that help you identify what should be built first.

Virtual Services

Virtual Services should be used where a collection of internal services is combined to provide an external view for a customer, thus creating a "virtual service" that is one which provides no direct business function but which offers a façade over those services. It should also be used where one service provides distinct operational objectives depending on how it is being invoked.

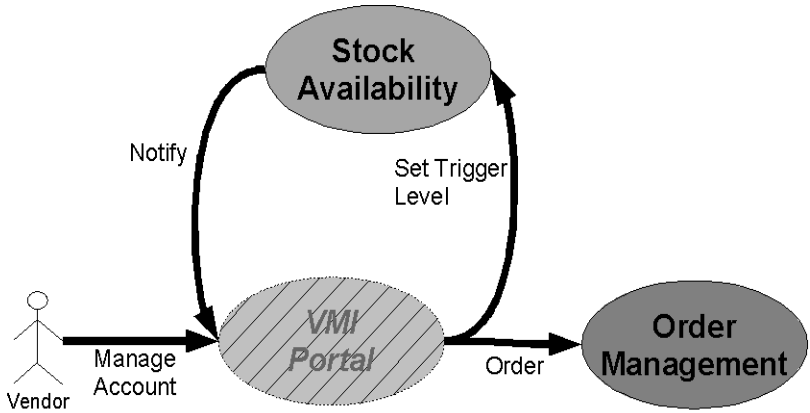


Figure 20 Virtual Service Example

Virtual Services are often the information or interaction points in the systems. Because they are virtual, having no direct business domain, does not mean they are trivial or are not owned by a clear domain within the business, and often can be the source of projects on their own within Enterprise models. Sales or customer portals are often represented as virtual services and owned by the Sales Level 0 service.

How they are differentiated from normally services, beyond being represented hatched on a diagram, is that they are not a service in which business logic will be implemented. Virtual

Services therefore provide a way to indicate where business logic can be co-ordinated and potentially simplified, but the implementation of the actual logic should only be done in full services.

Support Services

Support Services are often the technology capabilities that the business doesn't care exist, as long as they exist. These are split into two groups, those where have a clear encompassing service in the enterprise model, and those which fall within. Support Services have two distinct groups, technical and associated.

Technical Support Services

Technical Support Services can vary from elements such as hosting and printing, through to specific plant control elements or an RF-ID portal. As with other services it is possible for these to be relatively high level and be decomposed into further technical services. The key element with technical services is that they provide support to a business function, rather than being the specific business function themselves. For this reason it is important to differentiate between pure technical services, and those business elements that have been automated. It is also critical not to fall into the trap of creating services from existing technical elements just because they exist in a given form.

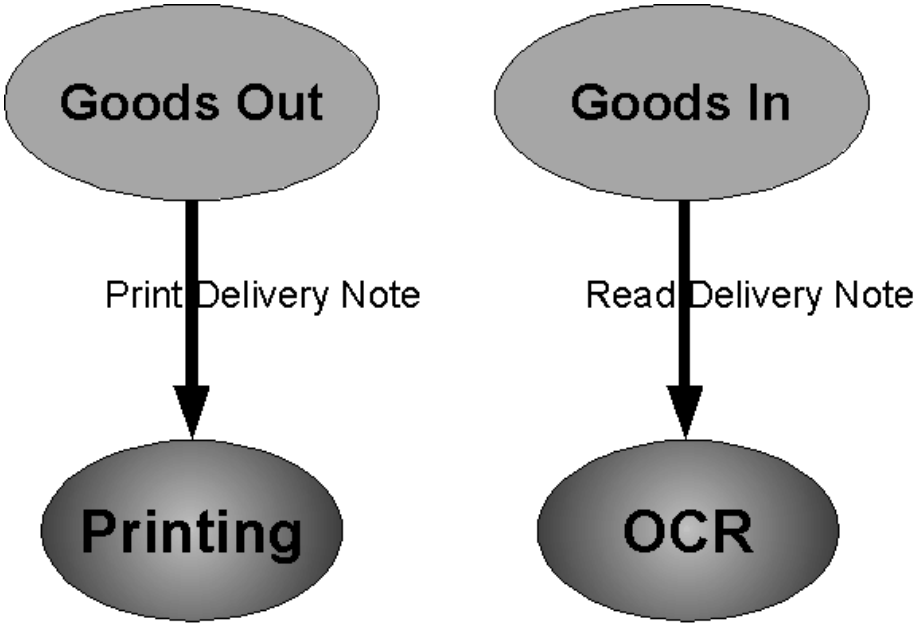


Figure 21 Technical Support Services

Technical support services are those elements that support the business, not business functions in their own right. Having an ERP system does not mean that "Enterprise Resource Planning" is a specific business domain in its own right. Nor does the existence of a CRM system mean that "Customer Relationship Management" becomes a support service automatically. The objective of a service architecture is not to determine the technology that will be used, but the domains of functionality that are required.

It is the job of application architecture and design to determine the most appropriate technologies and their implementations. Technical Support Services therefore are used only to indicate those elements which indicate a category of supporting function that is required by the main business elements. These are normally only defined at lower levels of granularity, and are normally shared between multiple domains, they are also consumed by Business Services, rather than being consumers of business services themselves.

Associated Support Services

Associated Support Services are those elements that are not required for the business operation of the system, but which are required for the business to operate. Elements such as Human Resources, Desktop Support and other internal only functions of business, and projects, need to be represented on the overall service map. It is important however to remember that these elements are **not** central to the operation of a business and should not therefore be exposed as Level 0 services.

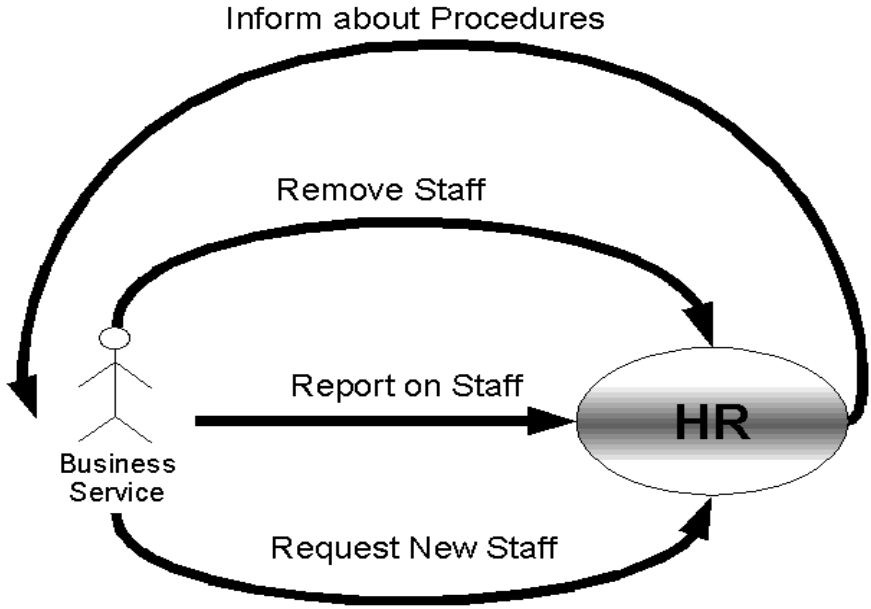


Figure 22 Associated Support Services

Associated Support Services should always be represented on diagrams in a specific way in order that everyone is clear as to what the primary goals of the project or enterprise are. Associated support services may often be important to the operation of the enterprise or project, but their existence is only to ensure that the business services are delivered, without the business services there is no reason to have an associated support

service. Associated Support services are often also that provide a supporting function to multiple business services.

The relationships between associated support functions are often irrelevant at the business level and just need to be described only on the Associated Support service.

Shared Services

As a service architecture drives into more detail it is common to identify certain services that are common between multiple business areas. These services maybe technical or support services, or may on certain occasions be clear business services which are defined to work in two places. In all of these cases it is important to recognise that some elements of the service might be shared, or that the whole service is common between multiple areas. Representing shared services requires an increased level of control and visibility, it is recommended that shared services are grouped into two groups:

- Technical and Support services split by
    - Shared between multiple business services at all levels and across Level 0 service boundaries
    - Shared between multiple business services with a specific level of a hierarchy (e.g. within the Level 1 diagram for a Level 0)
    - Those with common or similar bases, but differing drivers or implementations.
- Business Services split by
    - Totally shared services with defined business reason for being shared
    - "Apparently" shared service, these are services that appear to have the same characteristics but are deliberately separated
    - "Common Base" these are business services that shared a common base of context but have been specialised for a particular business purpose

The reason for making these classifications is that it enables IT teams to understand which services **can** be re-used across domains and which *even though they appear common* should either be hosted or implemented separately due to differing business demands. It also helps to drive both business and IT investment decisions about where the most benefit could be derived, and to help with the prioritisation of work. It is important to consider that Services with differing objectives and measures can be implemented as a shared service by building the service to meet the most stringent requirements. This assessment should be made on cost-benefit grounds and requires a realistic view of possible re-use or more ideally common usage. This can either be done at this stage, or more normally identified as a potential at this stage and then clarified during a future iteration. The advantage of a clear Service Architecture is that it gives clarity and context for that decision.

Figure 23 Cross Domain Technical Service and Domain Shared

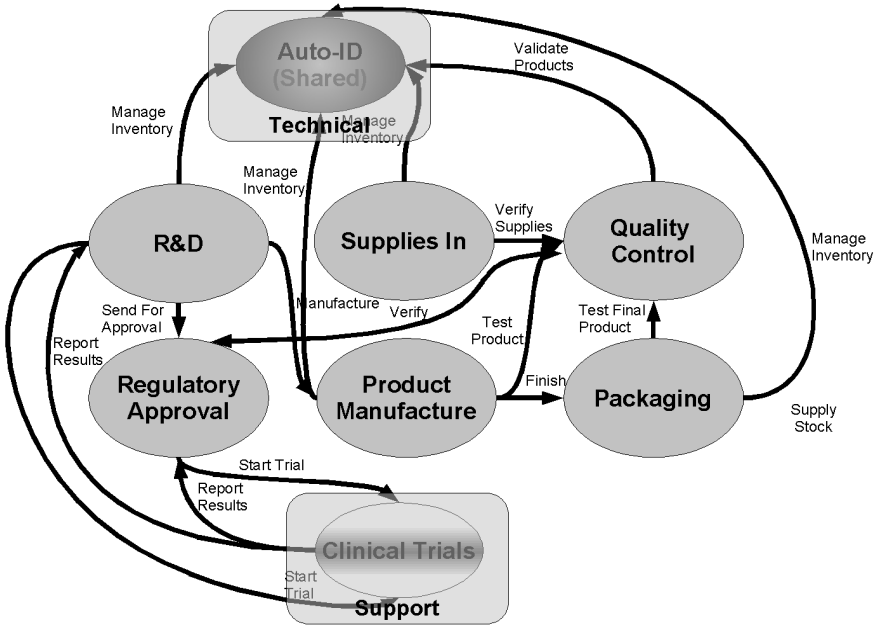


Figure 23 which is Figure 16 without the external services, shows an example of both a cross domain shared service and a domain specific share support service. The examples chosen here are for Auto-ID (barcodes, RF-ID, EPC etc) technical service, which need to be available across the organisation, and of clinical trials support services which while specific to this domain, are to be shared between the R&D and Regulatory efforts. Diagrams such as this should be used as decorations to the key service models to demonstrate additional elements, rather than trying to put all these elements on the main business diagrams

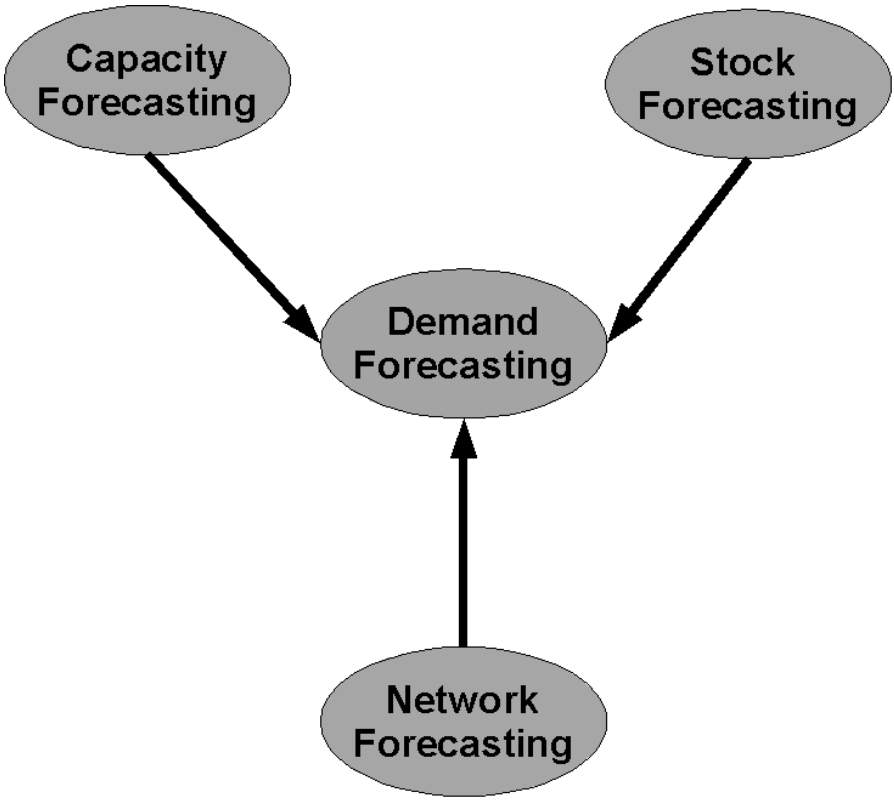


Figure 24 Common Base for multiple Services

A central theme of these diagrams is that they direct the strategy of these support services. If an organisation wished to trial multiple different Auto-ID solutions it would be reasonable to specify these services as not shared, but as having a common base. A future strategy that adopts one of these approaches would then be focused on unifying these disparate services. Figure 24 gives an example of another secondary diagram that can be used to represent common bases for services, as with a multiple pilot approach for Auto-ID all of the various types of forecasting are at their heart forms of demand forecast. This does not mean that these services are the same, or even that they share any business drivers or goals, but that the conceptual frameworks for the services share a similar base.

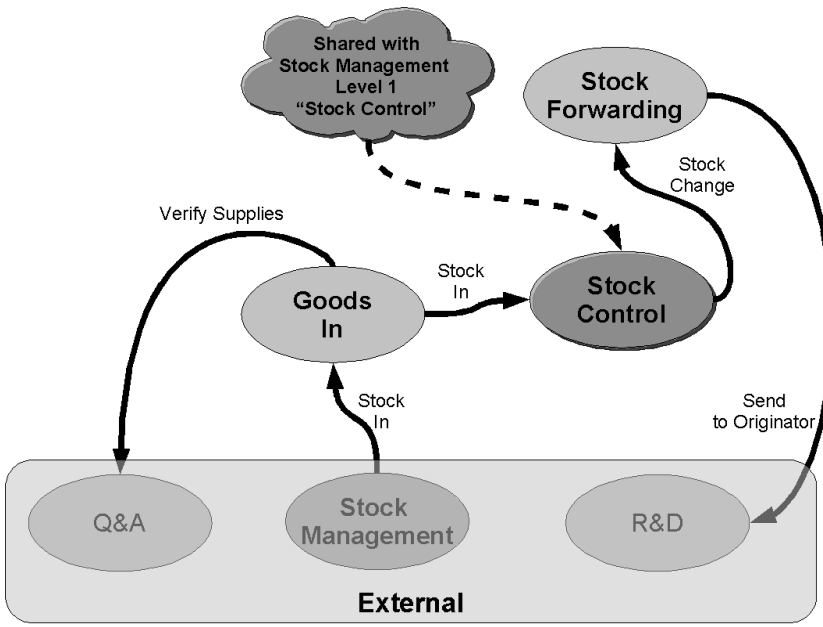


Figure 25 Shared Business Service

It is important to identify these common bases as they can, on occasion, lead to the identification of potential shared services and also identify areas where business services can co-operate on common ideas and approaches. This is particularly important when trying to foster cross-functional links in organisations as it can be used as a basis for common understanding. Figure 25 ows an example of a shared service, in this case explicitly referenced, this is an example of where exactly the same functions are duplicated between two parts of an organisation, but the processes, priorities and directions of the service remain the same.

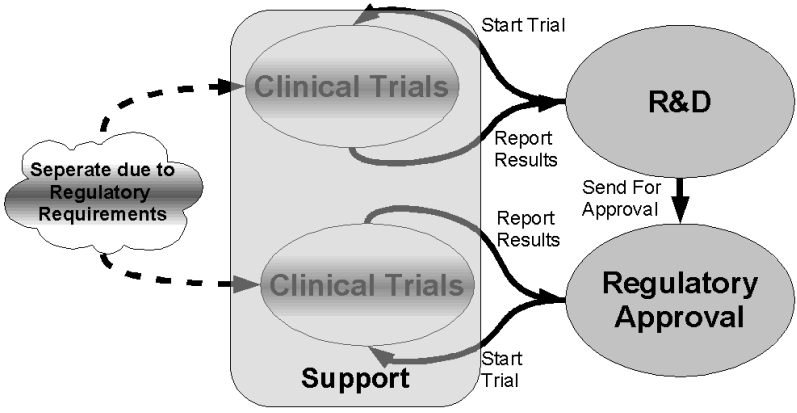


Figure 26 "Apparently" Shared Services

These shared services can often be physically manifest by separate departments, but which a shared IT and audit capability. It is important to note at this level however that they are, from a *business* perspective performing the same function and that any real-world separation is a matter of logistics or convenience not of differing business objectives. Figure 26 details a case where services, though apparently shared, are in fact distinct services with distinct objectives.

In the case of this is due to regulatory pressures on the market in question which ensure a clear separation between clinical trials

during the course of a companies R&D and those conducted for regulatory approval. "Apparently" shared services are often driven by regulatory drivers, either due to differing legal rules across regions or due to specific legislation around a business domain. When decomposing these apparently shared services it is normal to identify elements that are shared services, and others which must remain separate. The difference between these three forms of sharing, Common Base, Shared and Apparently Shared is more than just a semantic one. Common Base services are functionality different but have either a common "ethos" or core approach, they however are not the same either in form of their objectives or their direct function. Shared services are alike in both their objective and their function and can be view as a **single** service shared across an organisation. Apparently Shared services are those which share both objective and function, but are separated due to indirect concerns, in other words they are kept separate by factors not driven by the business. By identifying these groups, normally as a third or forth iteration after the basic business service model is done, businesses and IT can identify opportunities for consolidation and co-operation, but done within the correct legal and operating constraints.

8

Building The Complete Architecture

A properly defined SOA is the start to any architecture work. What effectively the business service model has defined is the context, concept and goals of the whole architecture. It has also described the various different domains and the different business drivers that they have. As the full architecture is created the focus is then, within a project, on understanding where these services "live" in the infrastructure of the business and how they are to be actually delivered to users. Within an Enterprise Architecture it is the process of understanding the business, IT and vendor strategies and determining the best technical and architectural standards and practices for the organisation to migrate to a full SOA. Architecture can also add all of the non-functional elements to the services, their security constraint, performance requirements, redundancy and reliability and generally use the basic service architecture as the starting point for a fully fledged project or enterprise architecture. This is Service Architecture therefore represents the skeleton onto which the complete architecture, whether Enterprise or Solution, adds flesh, sinew and muscle.

It is quite normal that the definition of technical, and some support, services is left until the project architecture phase of the project, and it is certain that both enterprise and project architecture phases will modify some of the service understandings, in particular around shared services.

When attempting to move an organisation from "Big to Small"[14] the services architecture helps in two ways, firstly by identifying the small elements, and via the hierarchy helping to understand

what the big picture is as well. The key when evolving the architecture is to pick a good practice guide and toolkit that helps you shape and publish the architecture as it is fleshed out.

9

Classifying Services

Defining services is the first step, and as stated earlier this can then lead in to many of the more traditional IT approaches, with Services providing the framework for that other work. Once services have been defined there is the opportunity to start understanding how services can be categorised, what different views can be placed on the service architecture and to understand in greater detail the right approach for a given service. There are many different types of classification that could be applied, but two of the key ones are described here, namely categorising based on business value, and categorising based on technology and delivery approach. Combined these present a powerful approach to portfolio management which is linked straight back to the overall business objectives.

Value Classification

Value classification[15] enables the business to split the services up between those that drive value, to those which are considered cost centres. This helps both sides understand how funding decisions will be made and to focus minds on the services that genuinely matter.

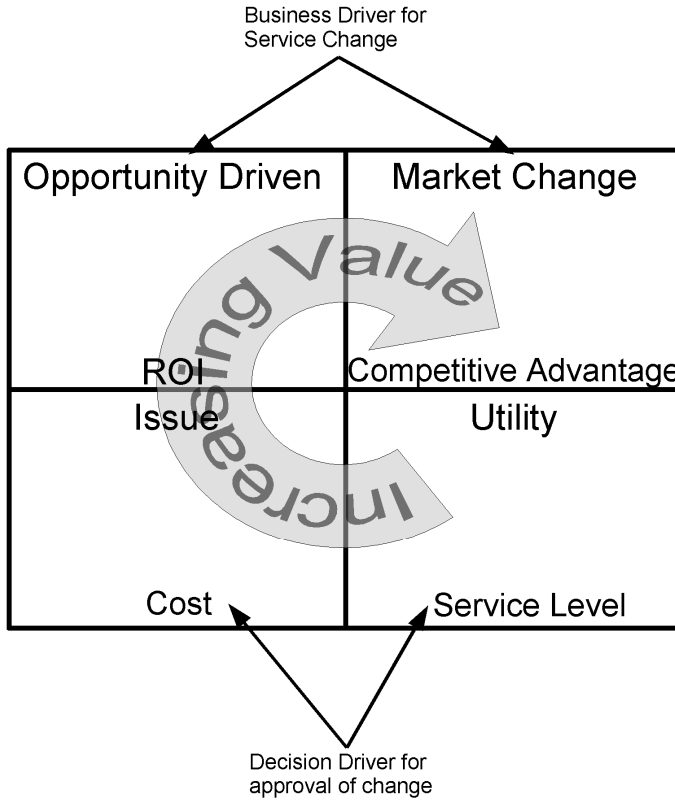


Figure 27 Simple Service Value Classification Matrix

Figure 27 shows a very simple classification matrix for services based on their business mindset or importance within the business. Broadly this splits services into 4 categories identified by what elements are used to drive change in those areas, and the way decisions are made to implement those changes. This enables both the business and IT departments to understand, and communicate, what will be the drivers and "valid" projects in a given area. There is little point in the IT department proposing a multi-million pound project to re build a legacy system in a new technology, when for the business change is driven by specific issues, and the system is adjudged to meet the 80/20 rule.

By having this categorisation, and recording where on the matrix a service lives it gives clarity on how the business is thinking, and on what changes are liable to be appropriate.

As an example consider 4 services

- A Network Support service would be in the bottom right, this is a heavily commoditised space where prices have been driven down and the key separation is the level of service that is offered to the business.
- Warehouse Management may be issue driven by a "if it ain't broke don't fix it" mentality, in this case it would be in the bottom left and minimising the cost of a change would be critical.
- Marketing is in the top left, driven by identifying opportunities and measured by the returns on those opportunities.
- Stock Forecasting is in the upper right, its intention is to react to market changes, and to deliver clear advantage to the company over its competition.

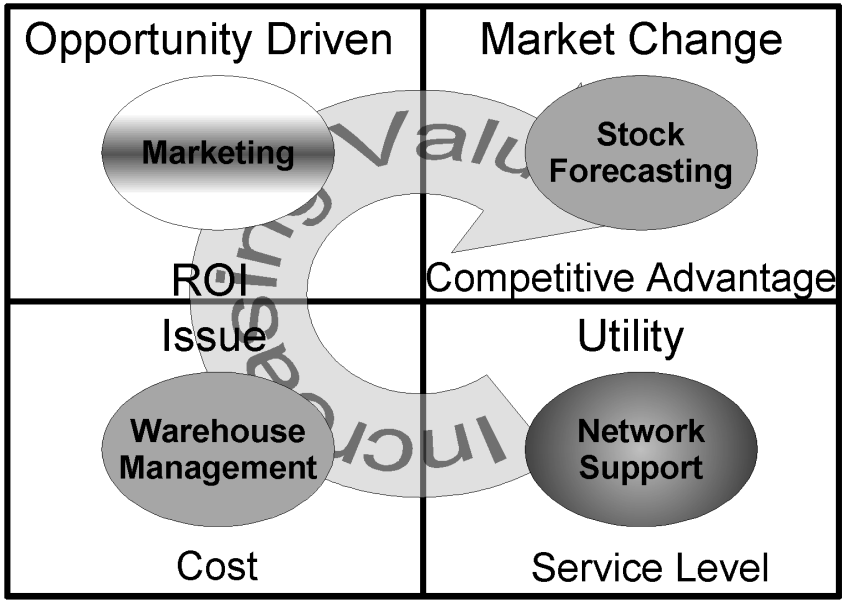


Figure 28 Example Service Value Matrix

This also shows how a single service may decompose into multiple services with different values. The Logistics and Warehouse Level 0 service might itself be considered to be Opportunity Driven, while a service at Level 1, and below, in that domain could fit in any one of the sectors. It is important to consider each service individually, as well as within the context of its parent when determining its value. The value of the service is related to how it is driven within its own domain, not how it is driven in the context of the business overall. This is done to ensure effective granularity that is appropriate to that service.

This approach can then be applied to the overall Service map to create a "heatmap" for services identifying the most business critical (red) down to the least (blue). At a glance its now possible to see where investments are going to be made and where commodity should be the order of the day.

Having this sort of visibility of the business value is essential to IT understanding how the architecture needs to be finally implemented and managed, viewing all services as equal leads to over investment in some areas and under investment in others and to implementation choices being made without all the available information. As an example it would be extremely rare for a global organisation to suggest building its own network infrastructure to connect its offices, this just wouldn't be a sensible investment as while its important to have these communications they are considered a commodity element. Businesses that do not let IT know the pieces of their organisation that should be viewed as valuable or commodity are doomed to have Rolls Royce solutions for the commodity and Trabants for the valuable.

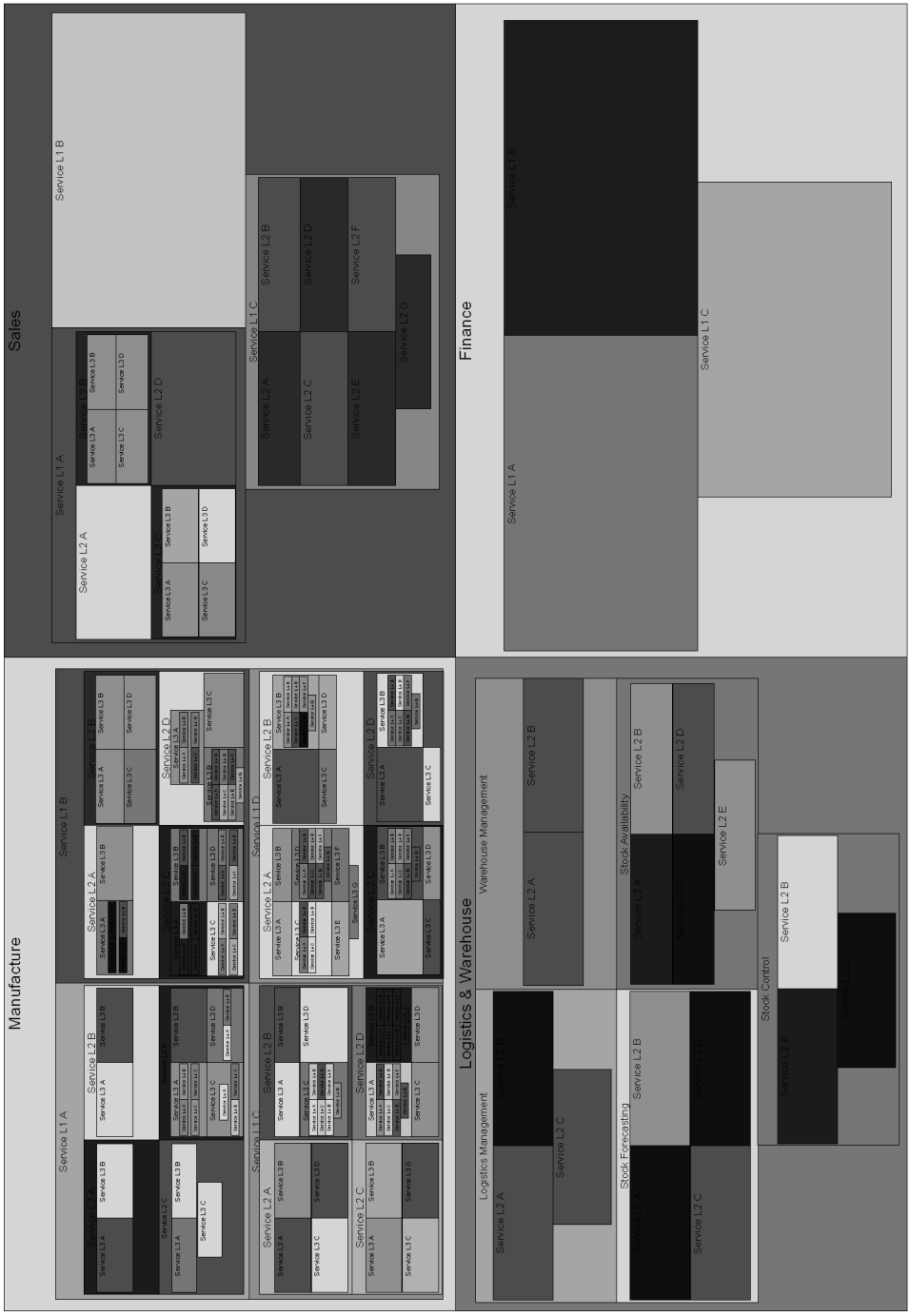


Figure 29 Service Architecture heatmap

Classification for Delivery

The approach described here can also be used at a business level to determine where services can be delivered from, but is more usually used on a project to determine which services will be delivered where and to identify the high risk services that should be tackled early, and tracked carefully. This matrix is used at the start of the project to aid in the estimation process and to provide clear contractual separation between companies if required. It can be done at many levels, from broadly identifying the level 0 services that different companies will deliver to decisions on what individual will deliver a specific low-level service.

At this stage it will probably help to consider a set of Level 1 services within a project which is intended to be delivered in a distributed manner between the UK and lower cost offshore centres, such as in India. These services should map to discreet deliverable elements that everyone can understand. At this stage the classification depends on the type of project being undertaken and the number of different delivery options.

There are many different ways in which the services in a project can be classified; Figure 30 gives a simple matrix that helps to understand how a service could be delivered. It separates the requirements, design and delivery efforts of the project and seeks to understand the drivers and constraints on each of these elements.

The first axis (z) is the requirements axis. At the bottom of the scale are elements which are so well understood that a requirements document can be written anywhere and then just sent to the client for sign-off and at the top of scale are those ones that require real investigation and iteration with the business.

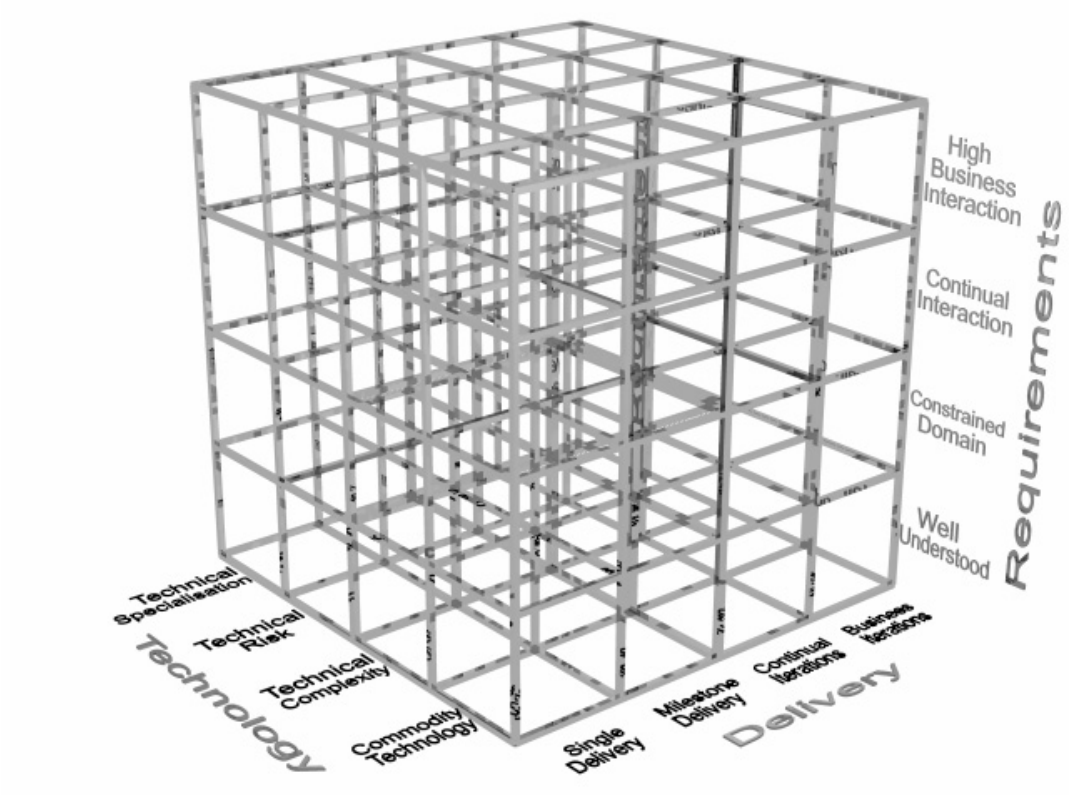


Figure 30 Delivery Categorisation

The second axis (x) is the Delivery axis. This is about the type of delivery model required for that service. Can it be delivered as one large element, are there sub-milestones or does the service require rapid iterations. These elements affect where the delivery can be done, and the environment that needs to be setup to support the delivery of the service. It also identifies areas where change is more expected, by identifying them as services that must be delivered in a more iterative fashion.

The final axis (y) is the technology axis. This axis determines the sort of technical challenges that will be need to be overcome in delivery. This helps to decide how different services can be delivered given the resources and teams available. It also helps

to identify the elements that need to be carefully tracked, and which can have looser management oversight.

So using the services described in Figure 17 a categorisation can be obtained for these services that helps understand how they will be delivered. Clearly before many of these decisions can be made there has to be greater refinement of the services, most often involving the architectural definition of those services and an initial perspective on the requirements. In a project environment however it is quite often the case that once the services have been defined it is possible to attempt a "first pass" delivery categorisation which helps with later phases, which may themselves refine that categorisation.

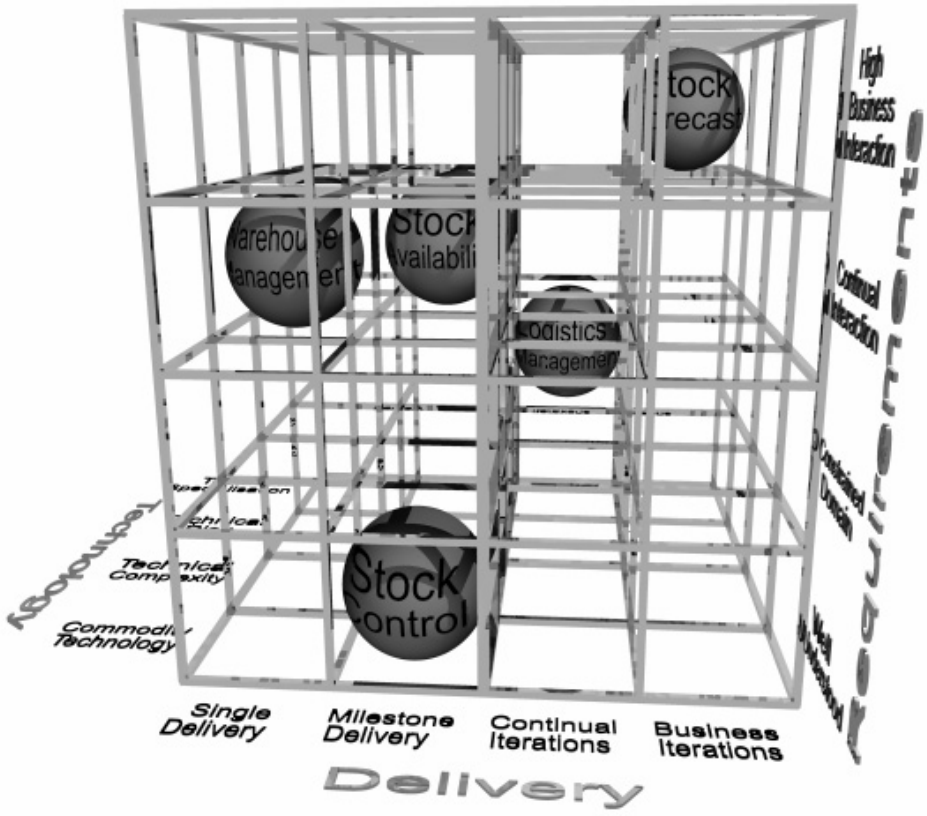


Figure 31 Categorisation of Services for Delivery

Here we have 5 Services being categorised. The most complex and interactive element is the Stock Forecasting, with specialist skills required for delivery. The "simplest" service is that for stock control as it represents a well understood area of the business and is on a commodity technology platform. With these services we can then make decisions about how best to deliver them. Splitting the service delivery into 4 elements, Requirements, Design, Development and Transition we can determine what is done where. These creations of visual representations of the work can help enormously in planning and estimation work, and also in communication around the business and technology areas in understanding why a particular delivery approach has been taken.

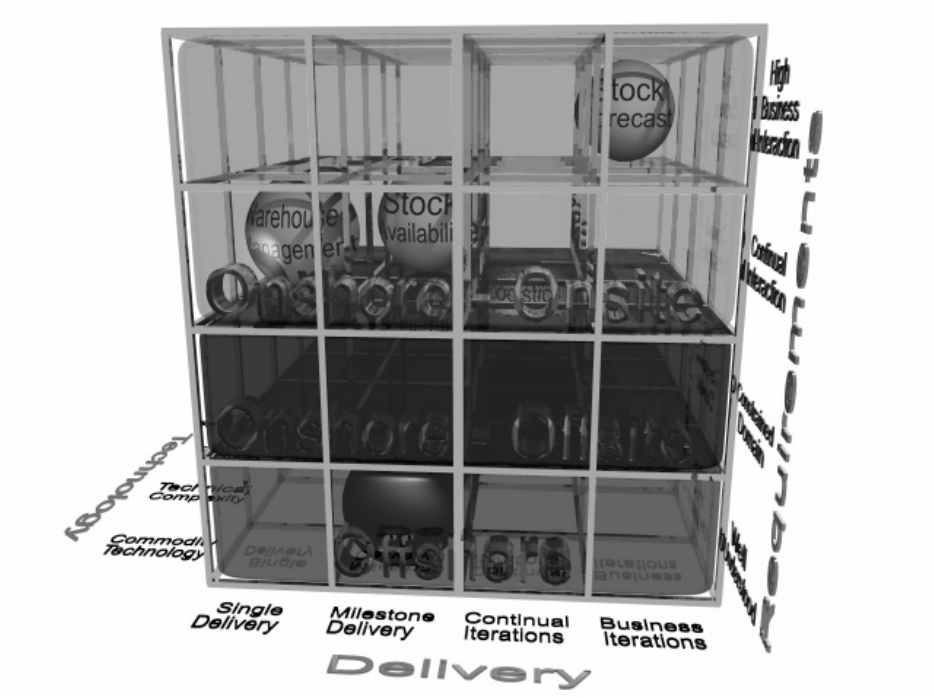


Figure 32 Requirements Classification

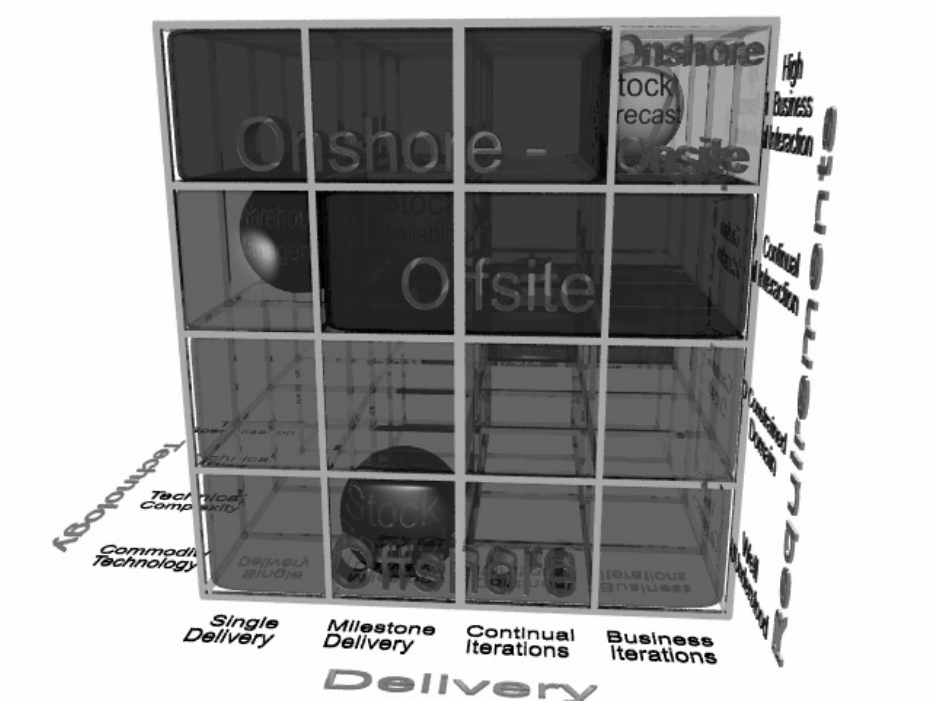


Figure 33 Development Categorisation

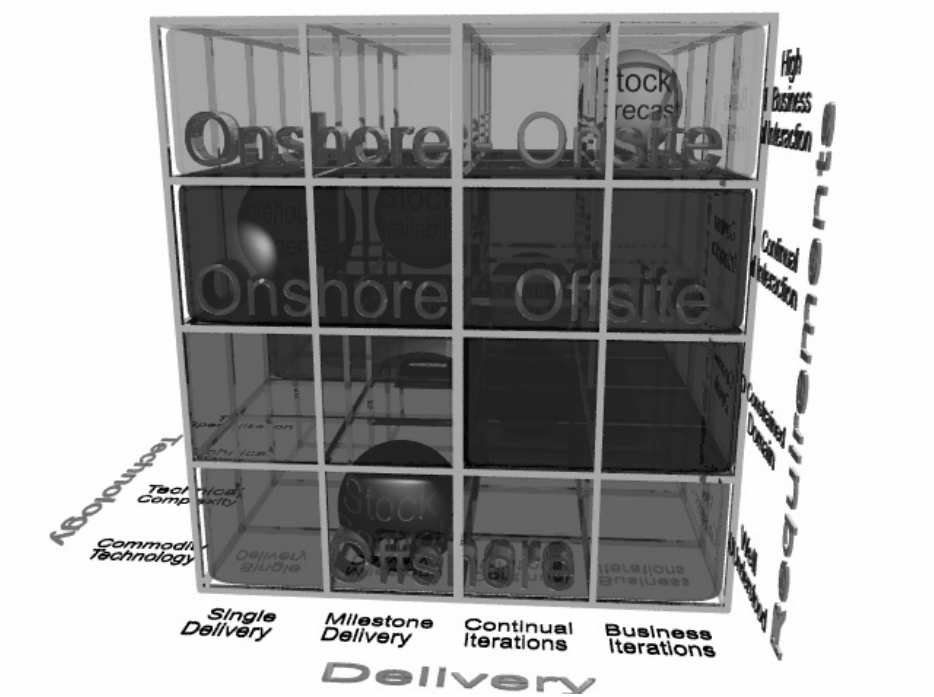


Figure 34 Design Classification

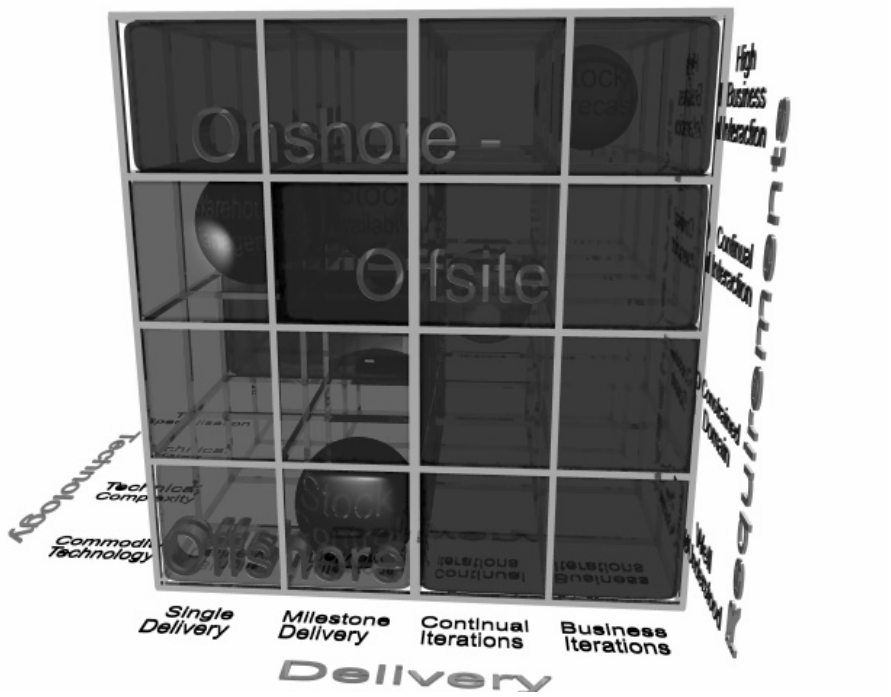


Figure 35 Transition Classification

These simple classifications can quickly convey to people on a project exactly what is being done where, and why. This then drives into the project planning, estimation and costing elements for the project and turns the costing of "one large blob" into the costing of smaller discreet pieces of work. One element that is required in an SOA delivery is the integration team, normally quite small. This should be considered as a technical service and categorised in the same way as other technical services in the delivery.

10

Measuring KPIs

Once you have a clear service architecture one common objective is to understand the valid measures on a set of services and service interactions. Traditionally this is one area that IT has been weak on with IT departments, and tools, relishing in the ability to show "all" the information to users. The reality is that different users need different information, and technical information such as queue depth is rarely of use to business users. Focusing on the business KPIs there are two simple types of measurement that can be quickly identified from service diagrams, firstly key volume metrics, and secondly key business ratios.

Identifying and measuring pure volume elements is never particularly difficult, service diagrams only make it slightly easier by identifying the flows that are measured and not just the end-points. These can also then be given weightings or scores based on their importance.

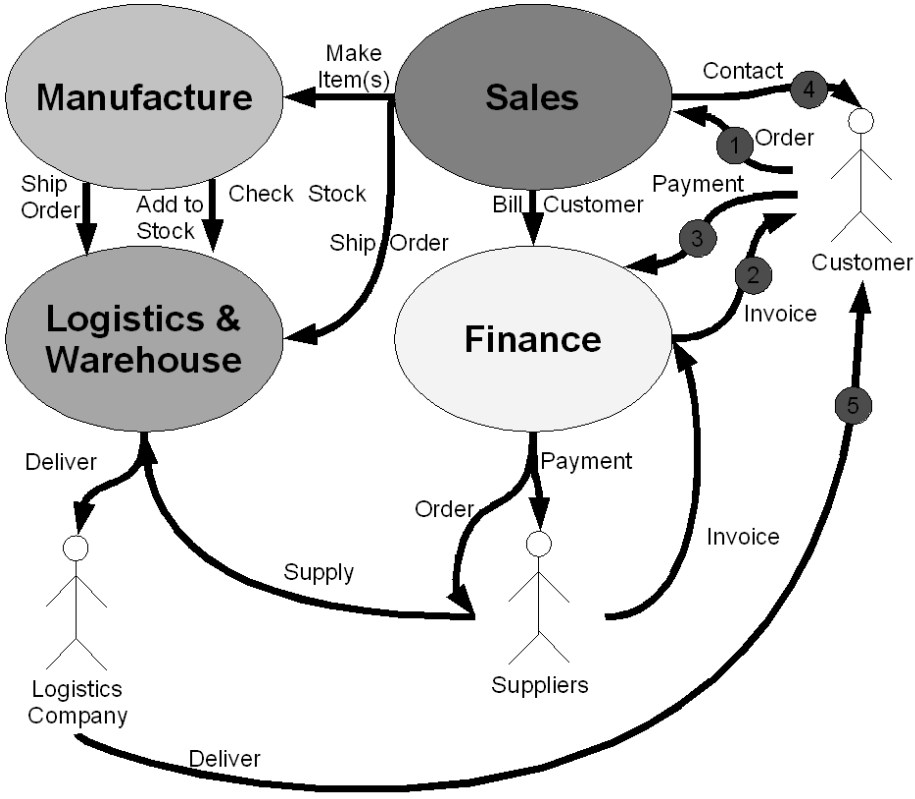


Figure 36 Volume Metrics

So taking the Level 0 enterprise Model as an example, and focusing on Sales and Finance, we have five key volume metrics ranked in order.

1. Orders placed by customers
2. Number of Invoices sent to customers
3. Payments received from Customers
4. Number of customer contacts
5. Number of deliveries made to customers

For each of these we would be looking at capturing the number of transactions, the value of those transactions and the unique customers involved in those transactions. And of course these metrics can be tracked over time to give trends.

The second and potentially more powerful approach is to identify key ratios between these interactions. There are several elements that are considered to be key business processes, in reality though the desire is to track the key KPIs rather than the entire process. So when people talk of "Book to Bill" they don't care too much about the actual steps required to deliver that process, but on the timeliness of Book to Bill and of the orders converted into invoices. The same can be said of many elements talked of as key processes, the important element is to track between two activities, the actual process itself is secondary to that tracking of KPIs. If there is an issue then the process becomes important as you want to indentify what is causing the problem, but again this could be viewed as a series of related KPIs rather than as an actual executing process.

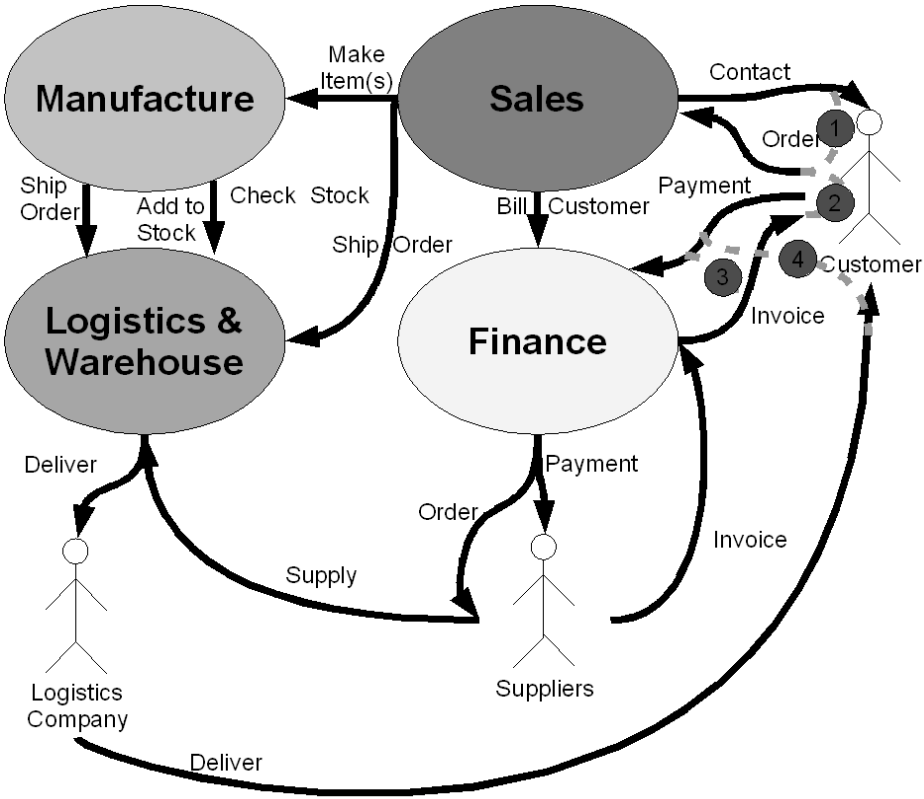


Figure 37 Relationship Metrics

Figure 37 shows examples of some of the key indicators that are defined using relationships between capabilities. These are

1. Contact to Order
2. Order to Invoice
3. Invoice to Payment
4. Delivery to Payment

The purpose here is to demonstrate that these elements are linked in delivering KPIs for the business.

Its important to remember that these are not attempting to define some large business process but to link two business capabilities into a clear business KPI.

Unlike simple volume metrics these relationship based metrics require additional documentation to detail what the relationship means and how it should be measured. For Contact to Order for instance there could be several options.

- Total Contacts against Total Orders
- Type of Contact that led to an order
- Number of contacts against size of order
- Time between contact and order

There are of course many more options, and combinations of options, so its important to document what these relationships between capabilities actually mean. If using a collaborative tool such as a Wiki, or a more formal approach, these definitions should just be links from interactions to the KPI, which should then have the definition. The aim of this is to highlight that the KPI is associated not with the service itself, but between two interactions, this means it can be measured *outside* the service boundaries.

11

Understanding The Technical Language For "How"

The stage at which you have a clear Service Architecture is when it becomes sensible to consider what is the most appropriate way to define "how" services will be built. As with classification for delivery or value it is not sensible to assume that one size fits all. Some services will not be built using technology, some will require detailed specialisation in either technology or business areas and others will come "off the shelf" from a COTS or package vendor. There is a myth around from technology vendors with Business Process tools that all elements should be done as business process definitions before moving on to what ever stages are next. For some services this is correct, but in the main adding this extra step adds little or no value and just means the project takes longer from business vision to realisation and the clarity of communications is reduced.

The following is only a guide on the right "how" tool for a service, but there are a few clear rules that apply no matter what technique is selected.

1. The Service boundaries are absolute "how" techniques can
    1. Alter internal state and information
    2. Invoke external services
2. and cannot
    1. Contain activities that are internal to another service

Simply put, a "how" can only define and deliver capabilities within the bounds of one service, though it may call capabilities that are made external by other services.

In selecting the appropriate approach for "how" its important to understand the type of interaction that actors will have with the service, and the existing elements that will drive its delivery. Rather than considering every potential delivery approach only five broad areas will be considered here: Business Process, Human Workflow, Software Development, Package, Application Service Provider(ASP). Software Development here is taken to refer to all those methods, like DSDM, XP, RUP etc, which take a requirements to implementation view.

Application Service Provider is the easiest to consider and will therefore be done first, consider using an ASP approach for an individual service, or set of services when:

- Service, or set of services, are in the "utility" quadrant for business value
- Service is part of the "cost" quadrant for value, and front bottom left (well know, well understood, commodity technology) for delivery.

If you are using an ASP approach then your "how" should be defined by that ASP, selecting a commodity solution means that they will be telling you what is possible to change and how you should change it. Do not try and engineer your ultimate solution for something being provided outside your firewalls and which will have been designed to deliver fixed approaches, *you* must change to those approaches and not the other way around. Think of ASPs like Telcos, you don't tell them how to run their business or what they should provide, you just buy what you want from them and use it how they want you to.

The next simple choice is when to use "pure" package, this is defined as a standard off-the-shelf product with customisation

using configuration, not via development. Choosing here is about understanding how it fits most normally this is where:

- Service, or services, are in the cost or utility quadrants for value
- Service, or services, are in the bottom left for delivery

Some packages can operate in other quadrants but there is a common underlying theme in buying a package, namely you have stated a desire to change your organisation to work in a standardised way, this way is represented by the package. As with the ASP approach do not try and do what the package does not want you to do, or in a way the package does not want you to do it. Consider the packages to be providing the basic capabilities in these areas, this does not mean that you cannot work in other ways, but that you should not try to do that inside the package.

This idea brings us to the remaining areas of Business Process, Human Workflow and Software Development. These are the areas where you need to define *exactly* how you want to operate. These can exist in any value quadrant, but as a result have different drivers in each of those quadrants. The key difference between these and ASP or standard package implementations is that they are for the business and IT to define in their own manner. These three approaches are about doing those things that cannot, or should not, be bought off-the-shelf. There are however different reasons for selecting each of the approaches.

Human Workflow is a good technique for those areas which are task oriented and have significant amounts of asynchronous interaction with users, or which are solely about people interacting to deliver capabilities. These services are often then delivered using user centric applications such as Portals and the focus is on enabling fast and efficient task management for the users.

Business Process is a good technique when you *need* to model service interactions and where there is real value in

understanding each of the steps required in delivering a capability. Processes are also good at modelling interactions with Human Workflow capabilities or other activities that are asynchronous in nature. This later area is where business process modelling has significant advantages over traditional software development approaches as it represents a clearer way of modelling the synchronous and asynchronous elements.

Software Development is a good technique where the business doesn't *care* how you implement a capability as long as it is implemented correctly. These are areas where the business sets clear goals and external requirements for services but isn't concerned either for the technology or its implementation. These elements can occur in all quadrants in the value chain and are most visible in new technology or business areas where these is a desire to do something new, the top right of the value quadrant. Example of this would be the drive to get an "e-business" application at the turn of the millennium, or the desire to "fix" stock forecasting where the business will define only what "fix" means and leave the implementation to the delivery team.

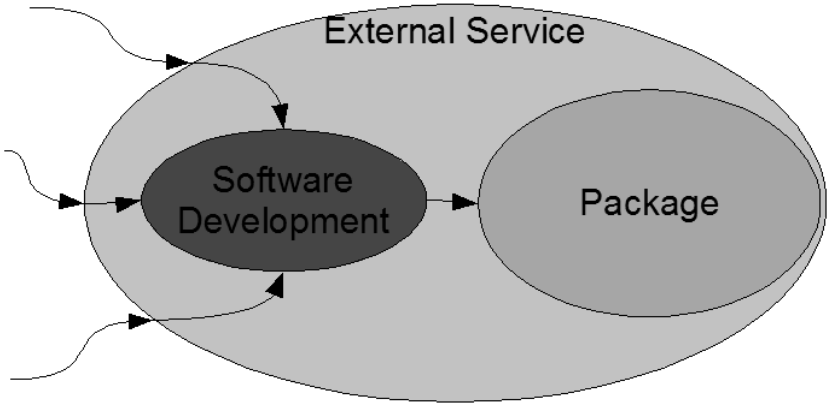


Figure 38: Package and Software Services together

None of these elements are mutually exclusive in a services programme, but each service should have a consistent approach, you don't want a service to mix the approaches. For instance if you have a package but want to do customisation externally using software development tools you should break down the service into two technical areas, most often with the software development one acting as the true externally facing service.

Understanding how services are best to be defined, and therefore implemented is a key step in the delivery of any service architecture. Avoiding the "one size fits" all approach is critical to making sure that services are defined in the correct way. This decision should be taken in conjunction with the question on "where" the delivery should take place but the two decisions are not always directly related, particularly for non-package or ASP solutions.

Service Oriented Business Processes

One of the biggest impacts of Service Orientation is on how business processes are captured and defined. Traditionally there has been a clear separation between Business Process exercises and IT implementation, and where the two have been brought together it has often led to issues. One truism is that the first time that the business knows its processes are wrong is when IT documents them, and by the time it works out what is right IT have implemented them in a way that can't be changed. The aim of service architecture is therefore to provide boundaries for these business processes.

In basic terms *an activity **in** a process is the invocation of a capability **on** another service or the alteration of the state **within that** service.*

When enterprises talk about processes at the top level of the business the words "service" and "process" are some what interchangeable, at this highest level process is used to just define a key area like "buy" or "sell" for example. The trick

therefore is to recognise the point at which processes stop being decomposed and start becoming actual orchestrations of activities. While process defines clear groupings and generic terms, and the word service or process can be used interchangeably then things are okay, at the point where it is clearly the orchestration of a set of Capabilities then service needs to become the dominate modelling approach.

The simplest solution is of course to always use Service and Capability as this removes the risk of architectures becoming process oriented. But this isn't always viable for several reasons including process work already done, and the time it takes to wean people away from process work. The aim therefore is to identify the tipping point between when Service and Process can be considered equals, and when Service must become the dominant factor.

For enterprise service architectures the tipping point is often, as shown in the diagram, at level 3 in the service model. The first three levels aim to define broad areas of control and critically do not seek to define the ordering of steps. At Level 3 however process models start to define orderings, and crucially often look at cross domain orderings, and it is here where Service *must* become the dominant factor. Unfortunately Level 3 is not universally the tipping point, some Level 2 processes may actually be about ordering, but again that should be seen as an indicator of when Service becomes dominant. Language is also important here, process elements that are purely descriptive, often only using nouns are less likely to cause issues when considering them as services than ones which imply end to end action, most notably those that include the word "to". Identifying the right tipping point for your enterprise or project is critical in ensuring that there is a single consistent view of what the enterprise estate should look like and how it should operate. Do not try and maintain two distinct models with complex mappings between them, this will undoubtedly fail.

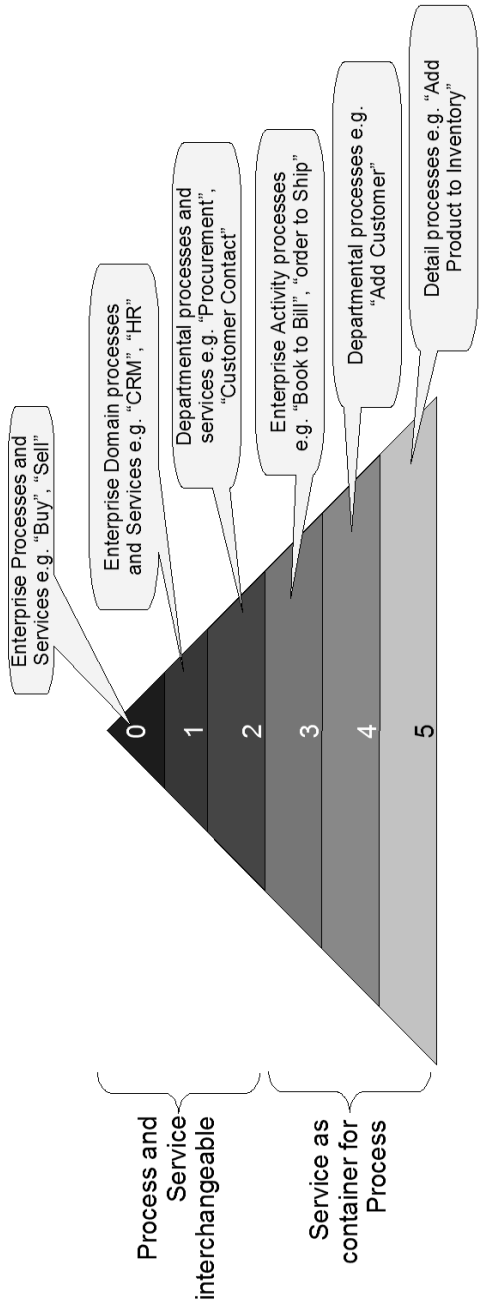


Figure 39: Finding the tipping point for Service/Process relationship

Service as a container for process

So what does it mean for Service to be a container for process? As mentioned earlier there are a couple of rules which come down to the statement that processes can only define and deliver capabilities within the bounds of one service, though it may call capabilities that are made external by other services. The other impact of using Services as a container is to provide the ability to retain control within a overall execution "thread" without having to have it be a single unified process. This enables dynamism and flexibility to be retained at each stage, with the service hierarchy providing the overall control and governance.

Figure 40: The process wheel

The process wheel aims to describe the difference between enterprise processes, which tend to be fairly static, and departmental processes that have to be faster to adapt. A single process that spans both the enterprise core and the detail of the department will have vastly differing objectives. The solution therefore is to clearly differentiate between the core enterprise processes, the departmental processes and the links between them.

This is where services really help as they are able to clearly delineate these different areas and enable the appropriate implementation, measurement and modelling techniques to be used for each one. By modelling services first, and then process we are able to ensure that we have the visibility and stability at the core, while enabling the flexibility required at the edge.

So when planning and mapping business processes it is essential to do this in the context of the service hierarchy. Taking an example of the process from initial customer contact to final payment for goods we would first look to understand what the process looks like at the very highest level possible, using the examples earlier this would be Level 0.

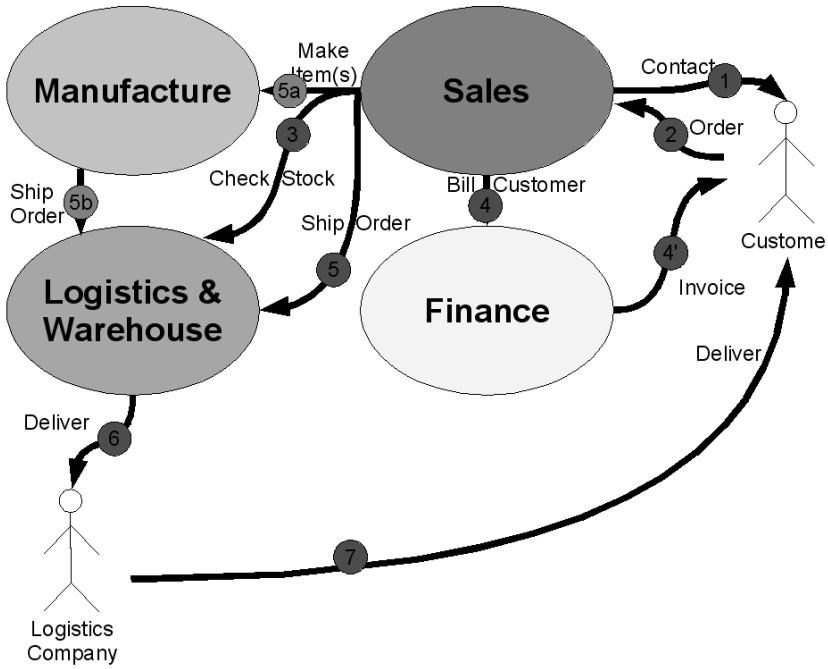


Figure 41: Level 0 Process Flow

Modelling process using the Service Maps gives a reasonable overview of how things should work, but doesn't give a clear enough picture when considering ordering or decisions. In the example shown there are two examples of these challenges. Firstly the modelling of concurrent flows, namely the finance and logistics related processes. Secondly is the modelling of an alternate flow, in this case the calls via manufacturing.
What this model does show however are the KPI points and their ordering, which enables us to define the tracking of a process rather than its actual execution. You could choose this approach to define execution, but it isn't recommended.

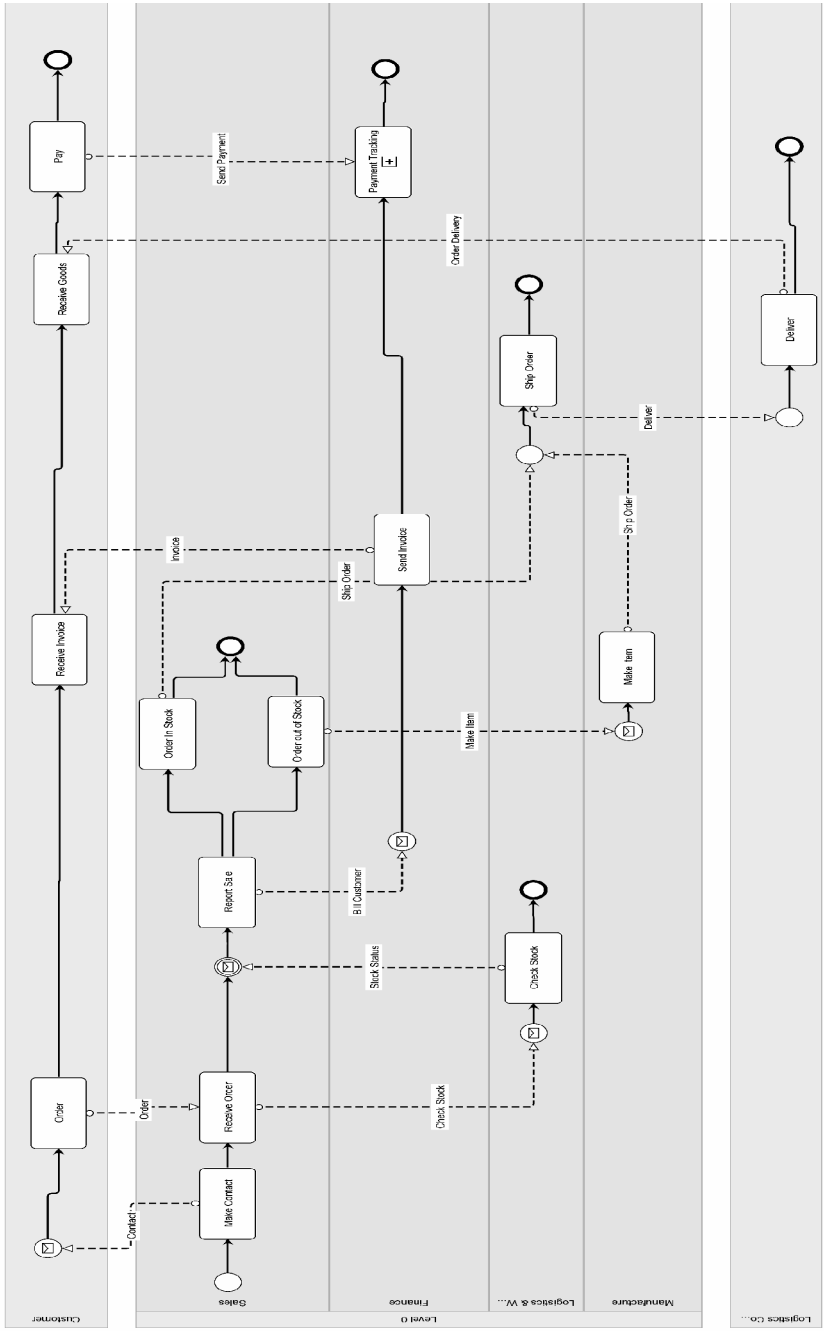


Figure 42: BPMN representation of the Level 0 process

This represents the simplest Business Process view of what is happening, this isn't an execution process as it doesn't include the decision making but gives a broad overview of what is happening. There are a few specific elements to note, which highlight why process modelling independent of service modelling is a dangerous thing. Firstly Logistics and Warehousing contains two distinct processes, one to obtain the stock information and the other to actually ship the order. Secondly to turn this into an execution process breaks one of the core roles, namely not having internal detail on multiple services, building this as a single cross-domain process would result in exactly the amorphous blob we are trying to avoid.

This highest level process is relatively simple, and in reality is liable to be fairly static and unlikely to change, representing as it does just the broad ordering of key capabilities. The next stage would be to drill down into each of these areas and understand what the capabilities themselves will be doing.

This execution process contains additional detail about the internals of the services and is beginning to model at Level 1 rather than Level 0. These diagrams can help in understanding how different processes are coordinated together to deliver a larger goal, but shouldn't be defined in this manner. Instead the recommendation is that each *Lane,* where a Lane represents a service, should be modelled independently, with messages being sent to other Lanes rather than to specific activities. For the model above for instance this would lead to the Sales process looking as follows.

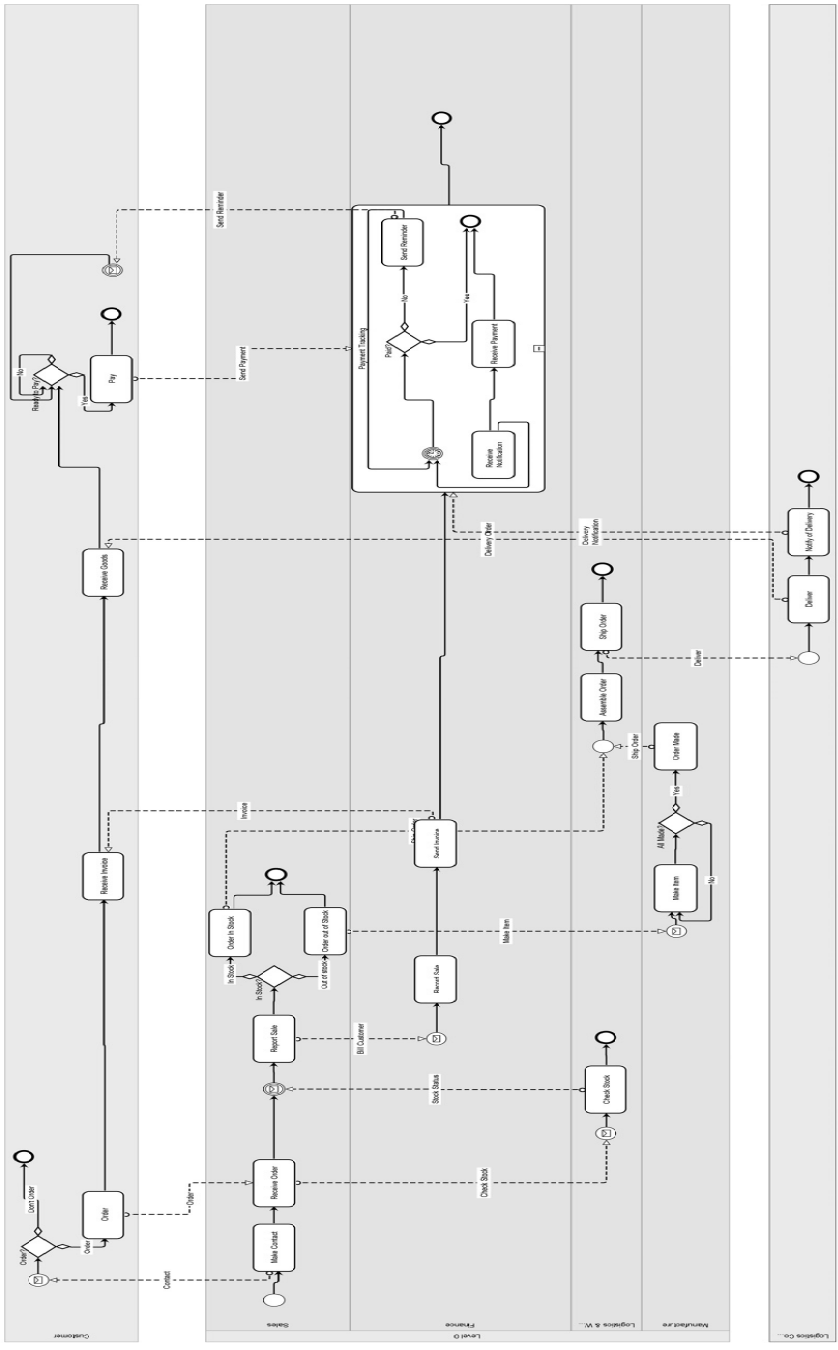


Figure 43: BPMN Execution process at Level 0

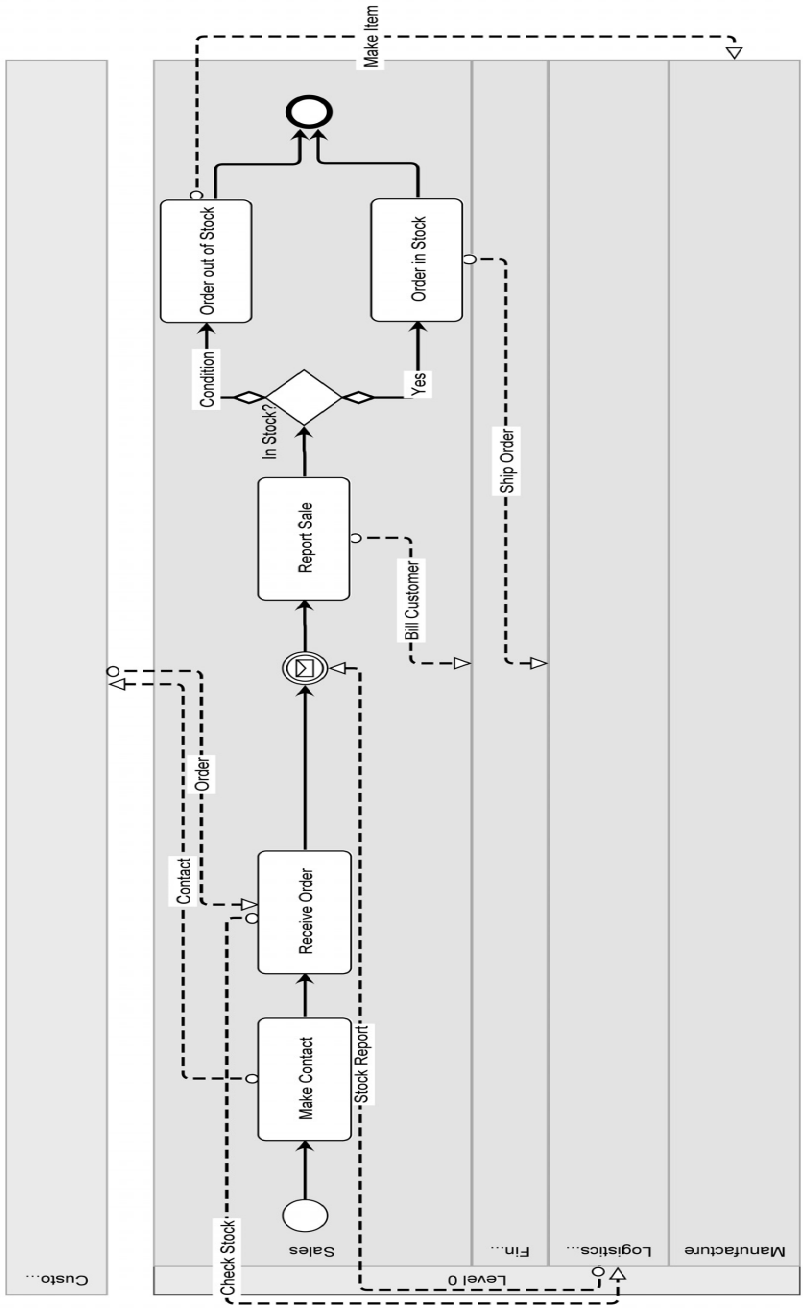


Figure 44: Level 0 Sales Process

The next element is of course to then consider whether these steps are parts of the service itself or are provided by lower level services. Taking the sales element above it might be that the "Make Contact element is actually provided via a call to another service. There are two critical elements here, firstly the difference in modelling an *intra-service* rather than *inter-service* invocation, the second in making the high-level view simple. The recommendation is that *intra-service* invocations are modelled as sub-processes, while *inter-service* invocations are modelled as invocations between Lanes.

With intra-service the aim of sub-processes should be to have a reference to the lower level service, an invocation, rather than having the whole process modelled in a traditional BPMN sub-process.

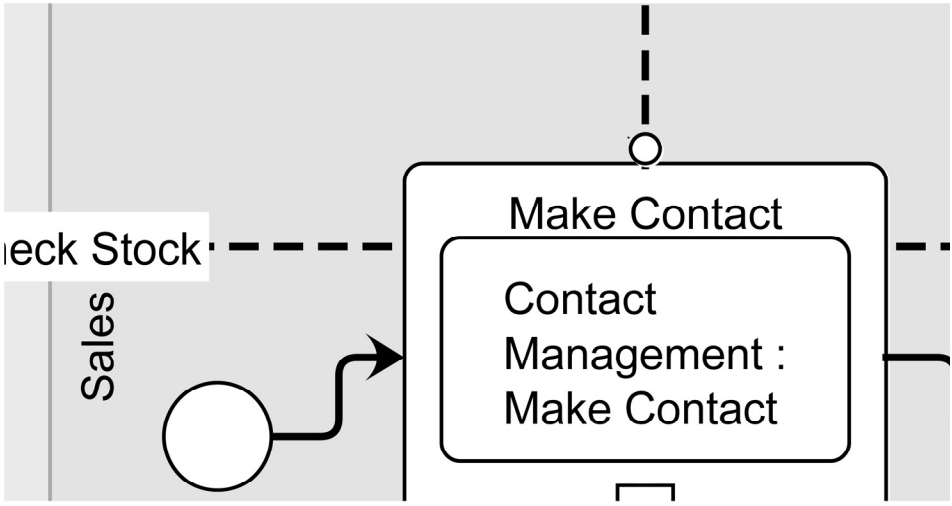


Figure 45: Referencing an intra-service call

This process can then continue as required, often resulting in changing notation if the best mechanism for capturing the requirements is no longer business process. As a result we have created an "end to end" process from contact to payment from which we can extract all of the KPIs required. By doing it

within the context of services however we have not fallen into the trap of creating an uber-process that is large and difficult to change.

Business Process v Execution Process

Another reason for using services is to help provide a mechanism of separating the business process from its actual IT execution. In December 2005 there was an example used in the OASIS SOA Adoption Blueprints group to help outline the difference. It was called "Christmas SOA", this was describing the difference in the perceived business process and the actual executed process for Christmas in my house. At the time Louis Jones was too young to really understand Christmas, but Lana was really into the swing of it. The business service architecture for Christmas for a 4 year old is pretty simple.

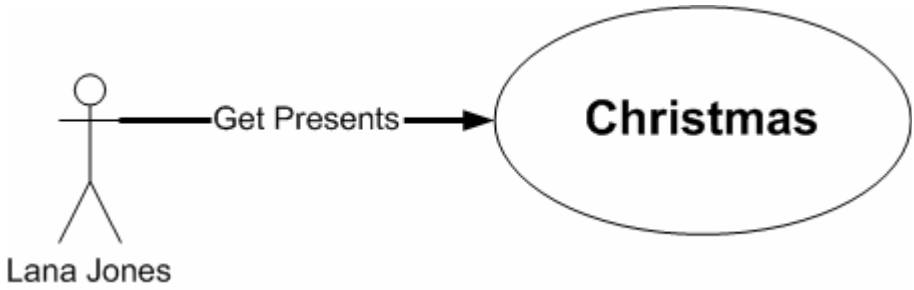


Figure 46: A kids SOA view of Christmas

There is only one reason for the Christmas service existing, and that is so Lana can get presents. Breaking it down further there is a slightly increased complexity in how this present supply will be undertaken.

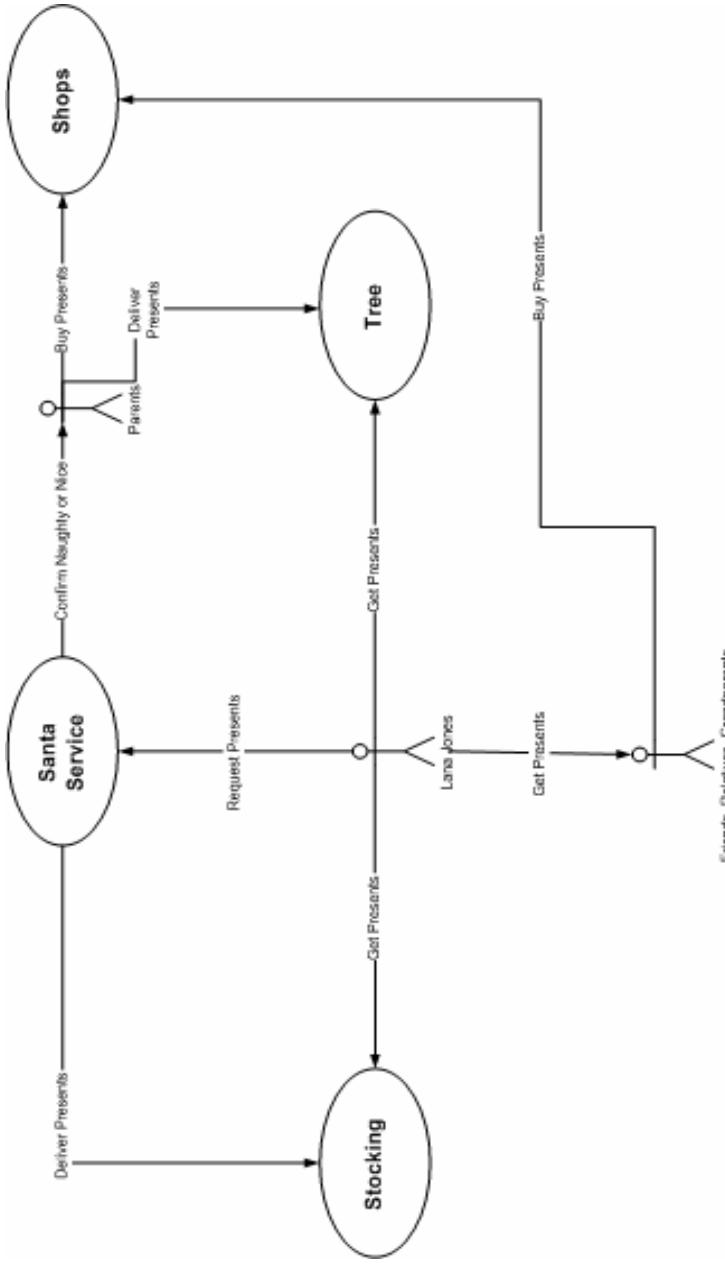


Figure 47: Level 1 Christmas Model

So there are three present delivering mechanisms, stockings, trees and anyone that she meets over Christmas. Key within this however is the ability to make present requests to Santa who is checking with the parents whether she has been naughty or nice (a powerful device in the lead up to Christmas). Lana, and Louis, represent the business for her parents at Christmas it is the kids season and getting those big eyes and the excitement is worth the investment by us on the other (IT) side. So to understand the business process we have to look at Christmas through Lana's eyes.

It is a pretty straight forward process really, she asks Santa for presents, he checks with the parents and then lots of presents arrive via the tree and stocking. Shops are not important to the process from Lana's perspective.

So down in IT they have a slightly different view on Christmas.

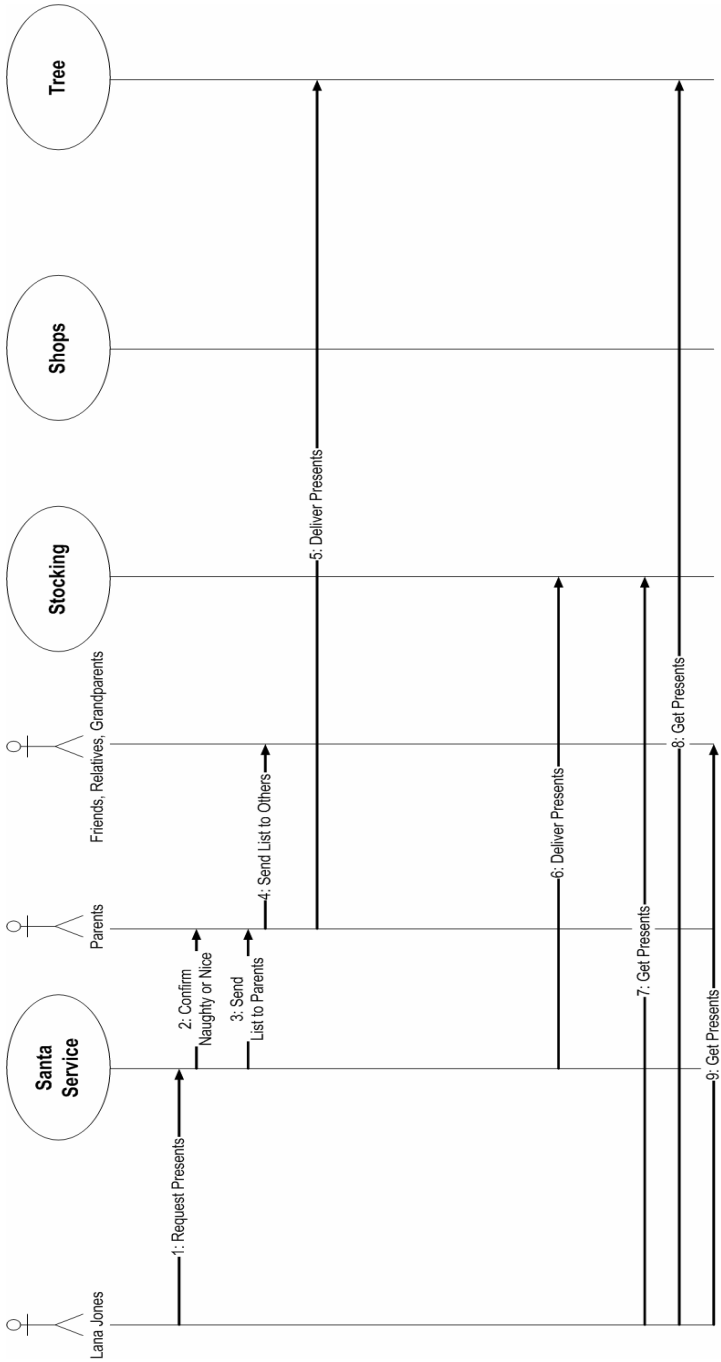


Figure 48: Lana's view of the Christmas Process

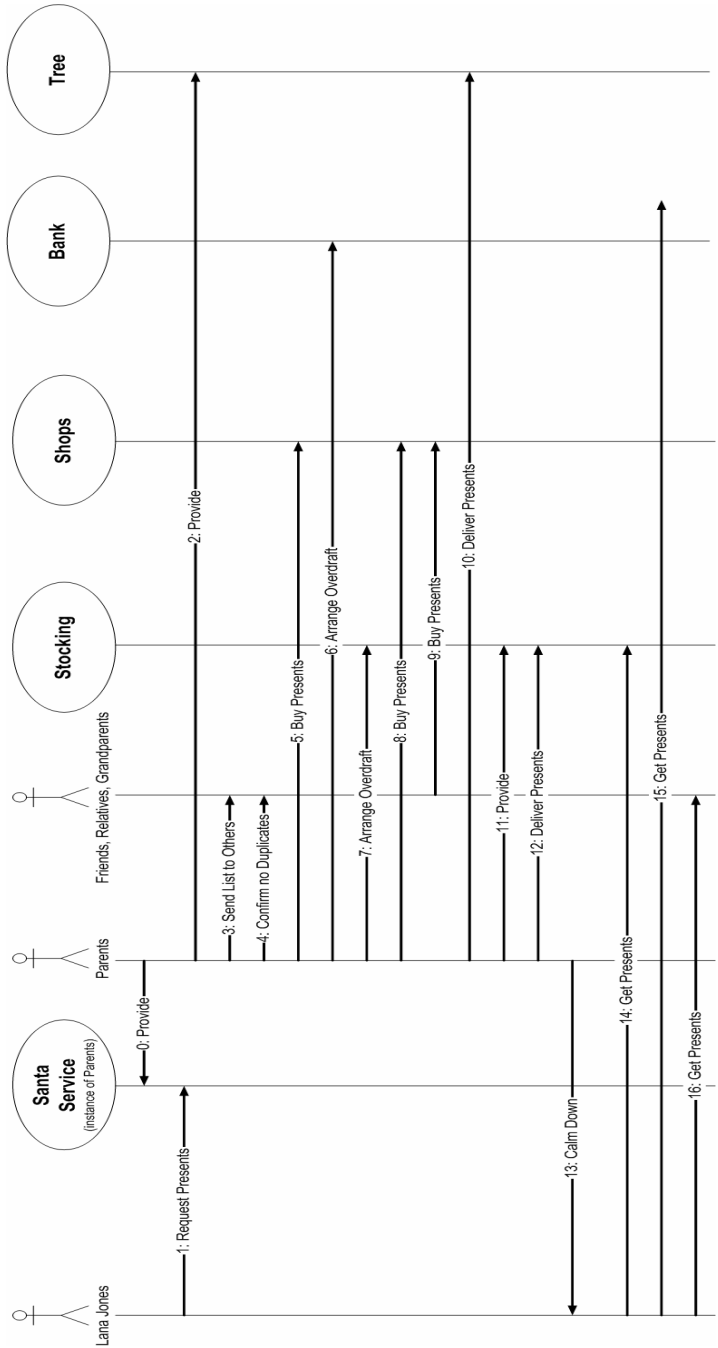


Figure 49: Parent process for Christmas

We have a new technical support service, the bank, and we consider the Santa Service to be something we have to provision in order to make it available to the kids. There is a huge amount of work that goes on behind the scenes to make that simple business process work.

There are two ways that you can manage this difference using Services, the first is where the top level process clearly works "outside" of the lower level services and the technical additions are clearly separated from the business process. This is the case for the Lana "Service" part of the process as there is no change in that interaction. Internal service execution is irrelevant from her perspective. The second is where the actual execution of a service differs in a way the business cares about. For instance here the Santa Service differs hugely in its execution between Lana's view and ours. In this case the best way would be to implement the business process as a set of KPIs which can be measured at their points of execution rather than having to have an actual executing process. So while Lana's process remains the same, her visibility of the progression of other steps, for example the population of elements under the tree or in her stocking, would be exposed via KPI measurements. From her perspective she doesn't care whether those KPIs represent the only steps in the process, just that they have happened.

This example also highlights the dangers of end-to-end process modelling within even in this simple example. With a traditional IT approach the business process would have been seen as the starting point, this would then have been *enriched* by IT to include the actual requirements for execution, and then put live as that process. This would mean that the business view would have become polluted with the technical implementations which have no meaning to the business. By using Services we are able to provide the same high-level process to Lana Jones in both cases (namely those interactions that she has) and are able to hide the technical detail behind the various service façades.

Services and UML

If the right way to model services is to use UML the similar rules to those used for process modelling are applied. Services again act as the absolute boundaries for requirements. Use Cases, activity diagrams or sequence diagram must all obey the fundamental rules:

1. The Service boundaries are absolute "how" techniques can
    a. Alter internal state and information
    b. Invoke external services
2. and cannot
    a. Contain steps or activities that are internal to another service

This means that Use Cases etc area limited to the domain of a single service and *cannot* cross that boundary in terms of definition but can invoke (or use) functionality defined in other services.

One way to enforce this separation is to use UML packages that align to the service hierarchy, this helps people capturing UML requirements to understand and document this information in the same structure as used for the services themselves. This does not remove the need for review or verification of the requirements against the service model but it does ensure that such verification is simpler.

12

Extracting Business Services From Existing IT

Once you have a business service architecture it is essential to start understanding what capabilities and services are currently delivered by the various existing systems. This not a simple process as it requires a real understanding of how these systems operate and regularly this is "known" rather than published information, and often therefore incomplete, incomprehensible or just plain wrong. The other challenge is that the IT estate has often been defined for the benefit of IT and therefore is not predisposed towards simple mappings.

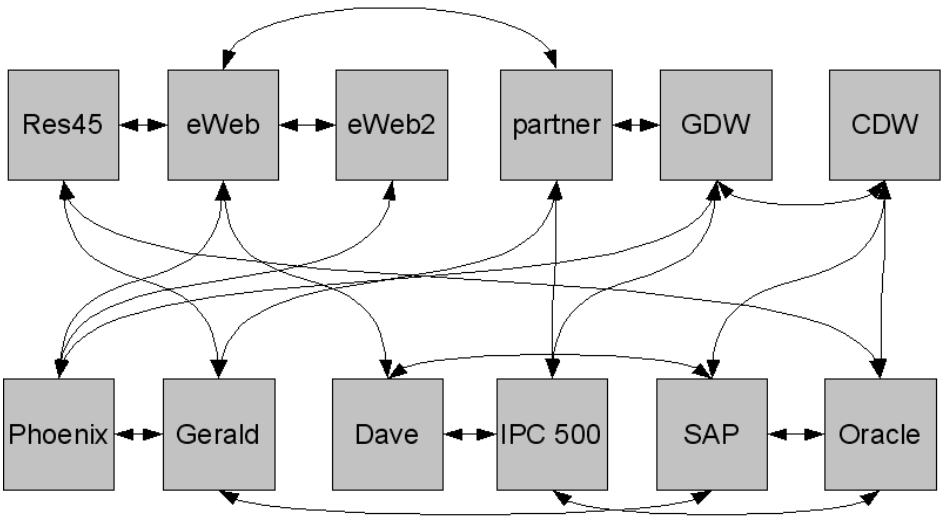


Figure 50: IT traditional view - no business names

This simple picture is in no way representative of the systems I've worked on, I've never seen a diagram this simple. The names though are representative of what I've seen, particularly it has to be said the "phoenix" name which has appeared at least once with every company I've ever worked with. The point is simple, the IT estate is named in a seemingly random fashion to anyone who hasn't grown up with how it works. This means on boarding is both about learning the business and learning the IT systems, though in reality for IT the focus is most normally entirely on the later with the business being considered something you "pick up" over time.

One possible way of starting to move from an IT encoded set of systems towards a business view is to first start with broad strokes by looking to use the high levels of the service architecture as containers for entire systems. Where systems fit into multiple areas there are two options, if its 80% in one and 20% in another then put it just in the one. If its split fairly evenly put it *all* in both.

Service Name	***System Name***
Quality Control	Phoenix
	IPC 500
	SAP
R&D	Huxley
Regulatory Approval	Dave
	Gerald
	Barney

Table 8: Example Service to System Mapping

After you've done that the next stage is to start matching the capabilities of the systems to the capabilities of the services. This will mean you start mapping out capabilities that are in other domains so you need to capture that information against

the right service, the reason for initially doing a broad cut is so you can then target specific areas.

Service Name	***Service Capability***	***System Name***	***System Capability***
Quality Control	Verify Supplies	Phoenix	Supplies IN
		SAP	Compliant Manufacturing
	Verify Regulatory Approval	SAP	Enterprise laboratory information management
		IPC 500	QA Regulation
			US Regulation
	Product Verification	Phoenix	Product Check
			Product OUT
		IPC 500	Product Safety Check
		SAP	Compliant Manufacturing

Table 9: Mapping capabilities to systems

This then gives you the ability to start identifying two key things

1. Whether certain capabilities are duplicated in the systems
2. Whether there are business capabilities not met by the systems

Further iterations can cover elements such as the percentage fit, ability to extend and other detailed elements. Because there is a business value classification for these services it is also possible to use that to drive exactly how you do that work, and in particular to set out the strategy in various different areas. The goal here is to create a map where you can understand where the business services *could* and *could not* come from in the most effective manner. Those that will not be able to meet the business goals should be set for decommissioning, those that are

able to change as the business requires should be scheduled for development based on their business value. Considering these existing systems as "packages" where software development is done externally to create the business view is often a profitable approach, especially for systems that are difficult to change but contain a solid base of capabilities.

13

Impacts Of SOA On Project Planning

There is little point creating a great service architecture that matches the business needs, classifying those services so you understand both their value and the best approach for delivery, and then going about your IT delivery in a big bang waterfall approach because that is what you've always done. Starting with the services is critical for Service Orientation, but to properly realise an architecture it needs to be delivered, and eventually managed, based on those services. This can have far reaching impacts, and benefits, for the organisation as it changes the emphasis from large projects towards large programmes composed of multiple service projects. This management of complexity and clearly defined separation of concerns helps to reduce the amount of documentation and focus the communication required, thus increasing the productivity of the people working on the programme.

As the number of people on a project increases, so does the projects complexity. As the number of communication lines increases, so do the timescales. These are long established facts[16] of project management. Therefore if the amount of communication can be reduced, so can the amount of effort. If the project can be turned for 100 people, into 10 projects of 10 people it becomes significantly less complex. This is where Services deliver immediate value to organisations who seek to use them.

Impacts On Project Portfolio Management

Traditional portfolio management has been about which *projects* to do. Projects are given priorities based on their individual ROI case or strategic drivers and this is then used to determine what should be done and with what budget. Using service architecture helps to focus investment more accurately and to identify ordering of delivery at a finer level of granularity than enabled at a project level.

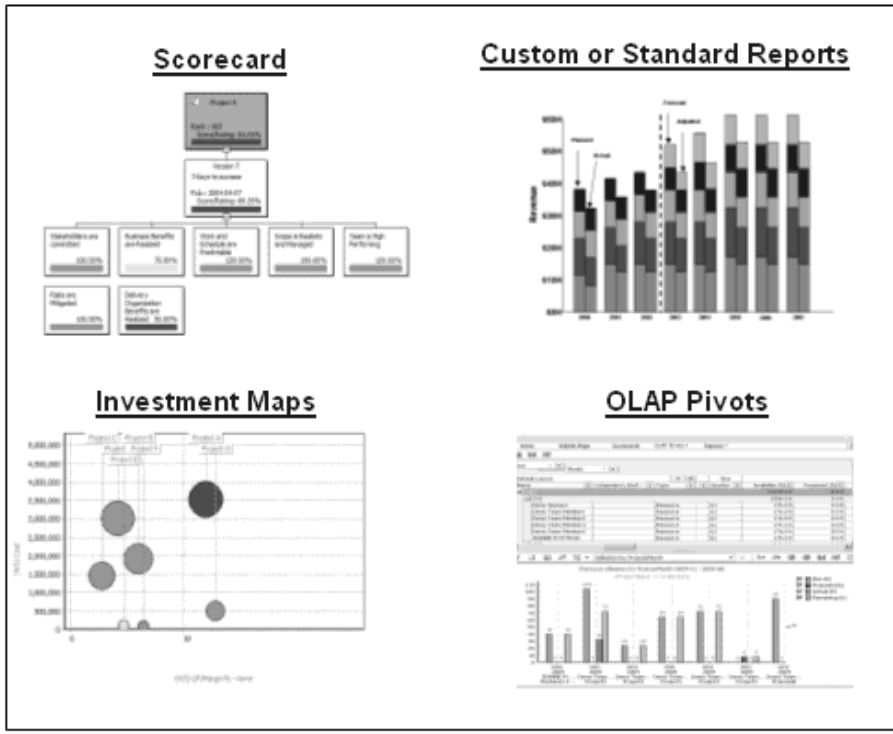


Figure 51 Images from Rational Portfolio Manager

In the traditional approach work is justified on a "large lump" basis with the ROI being based on the totality of delivery. This approach tends to make organisations move towards large code-named projects that deliver systems that become known by those codenames, rather than by the functionality delivered.

Organisations gain systems called “Phoenix”, “Genesis” or “Dave” with the names becoming over time the things which need to change, rather than focusing on the business change. This large lump approach also means that intra-project dependencies or benefits are often missed, with projects being costed based on the totality of their delivery, rather than the most effective way to deliver multiple projects.

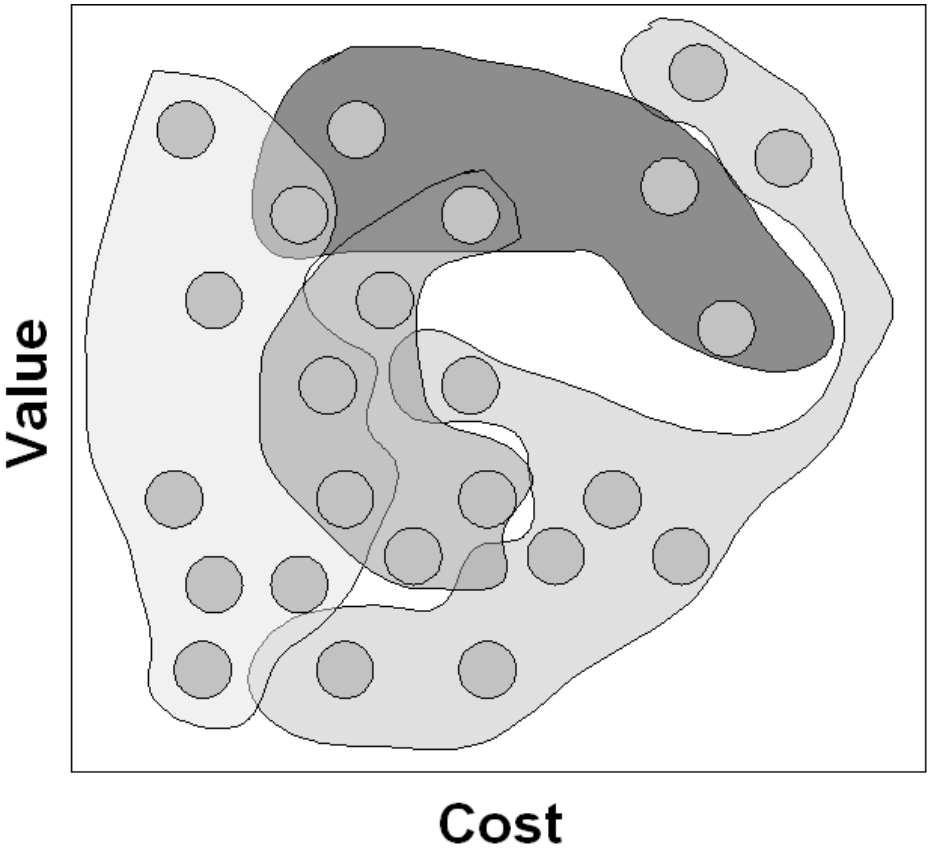


Figure 52 Service Projects with Business Programmes

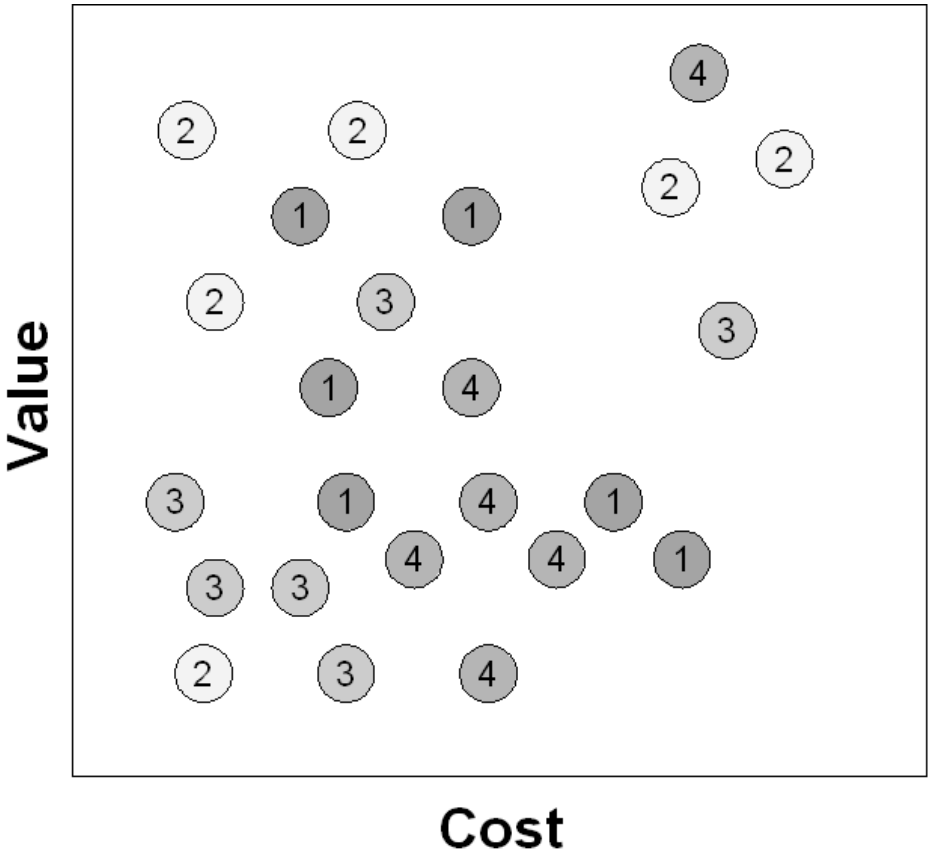


Figure 53 Service Projects mapped into IT delivery phases

Moving away from Project Portfolio management towards taking programme portfolio management as the standard measure, with delivery being based on individual service projects, enables organisations to view all future development requirements a view where the real benefits lie and which things just "must be done" in order for those benefits to be realised. As an example a project that creates a new stock forecasting service, the value, requires more integration than the existing approach, thus the integration is a required element for the value to be delivered. If multiple projects require similar integrations to be done then it is possible to cost this infrastructural IT element based on the clear benefits that it will enable the realisation of.

Linking IT infrastructure to the business benefits, and clearly classifying the difference between the commodity, the infrastructure, and value, forecasting, helps to focus the mind of how such elements should be procured.

The tools that are out there for Project Portfolio management can then be made to work, relatively easily, in a service context with programmes being the combinations of multiple service projects, this ability to manage the portfolio more dynamically helps to create more effective programmes that aim to solve multiple business goals rather than simply one.

Moving From Projects To Programmes

The largest impact on project management in the delivery of SOA is the greater need for programme rather than project managers. The focus becomes less on people and skills required to manage large groups of people and multiple streams of work towards managing multiple sub-projects and their interactions. Projects clearly still exist but they are of a much smaller scale than before.

This shift in approach, combined with the categorisation of services by their most appropriate delivery approach means there is a greater flexibility in delivery process that within a traditional project, rather than the decision being for a single development method for all work there is the option to tailor the ceremony required. And with Service projects being smaller than traditional projects there is a reduced need for documentation as team sizes will be smaller and more manageable.

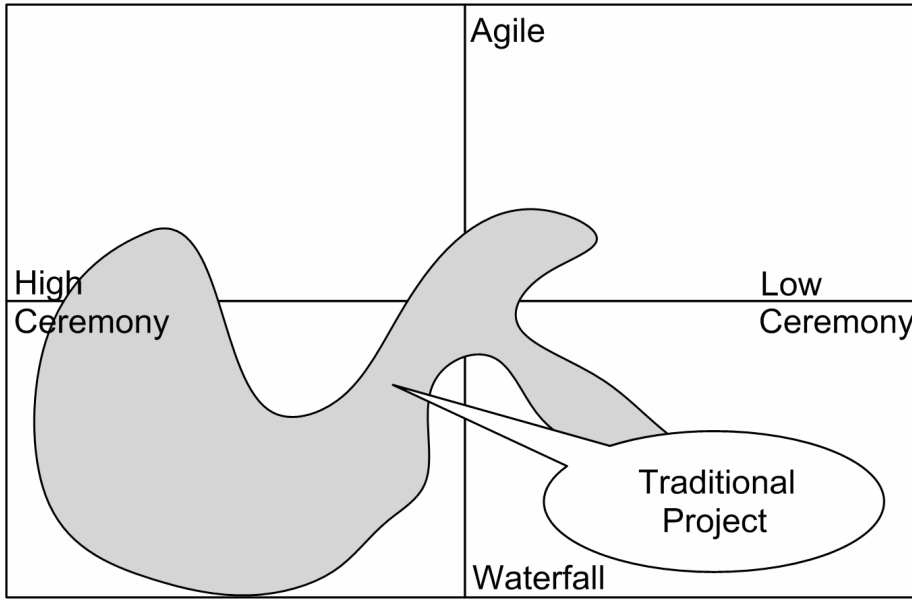


Figure 54 Common Large Project Ceremony Levels

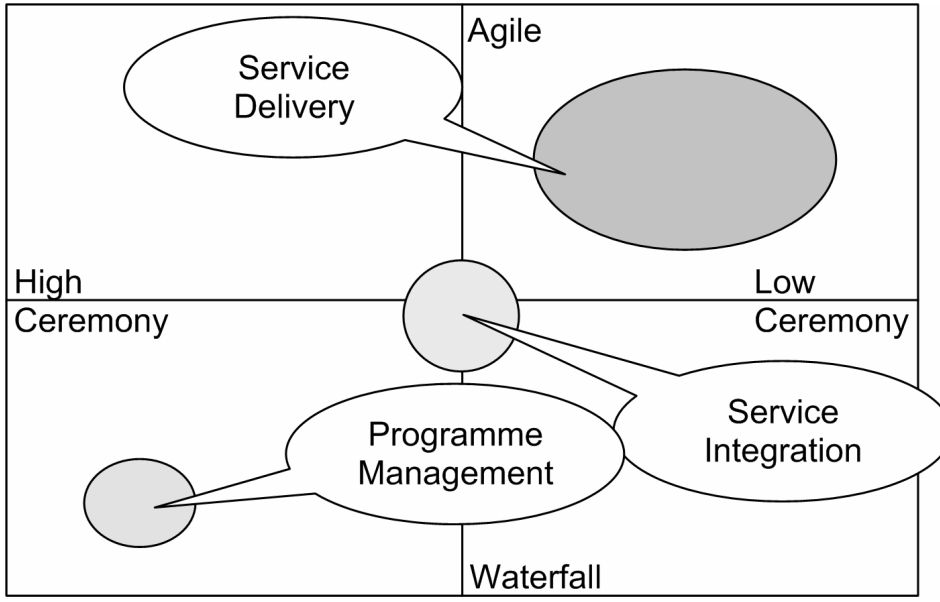


Figure 55 Ceremony Levels in SOA Delivery

While in traditional delivery (Figure 54) there is little ability to tailor the process to fit individual parts of the project there tend to be decisions made on the ground during a project to drop or add certain elements within different parts of the project. This is often done as a result of developer decision making, and not always for the best of reasons. Part of the reason that this happens is that governing a project with several hundred people is extremely difficult and telling the difference between good decisions and bad decisions is often something only done in retrospect.

Service based delivery and the delivery categorisation (Figure 30) also has a significant advantage when considering the right delivery method for each service, rather than having to have a single method that attempts to fit each service. Figure shows the broad areas that are covered within the project. This ability to tune the development of each service results in less overall effort and a greater degree of programme visibility.

This is not to say that SOA solves these problems of developer individuality or plain dumb decision making, its aim is to make them more visible earlier in the project. In this way project management in an SOA environment takes the learning's from approaches such as eXtreme Programming, DSDM[17] and AUP[18] but aims to make these approaches easier for organisations to adopt, and most importantly makes it possible to choose *multiple* different delivery approaches that are *appropriate* to the type of work being undertaken.

SOA based programmes therefore tend to split into three distinct areas based on the type of work being undertaken.

1. Programme Management – High Ceremony almost Waterfall like. The programme management job is to break the work into its services projects and then measure and track progress and dependencies. This requires a high degree of formalism and ceremony to ensure that different service projects do not impact each other. The *volume* of work in this area however is not

large and should be targeted only at the boundaries of service projects. It is the Programme Management's job to ensure that Service projects are not too large

2. Service Integration – Middle ground, requiring a degree of formalism and timed boundaries. This work is about bringing multiple services together and ensuring that they work correctly, as this works across streams it requires an increase in formalism and ceremony over actual service development.
3. Service Delivery – Depending on the categorisation of the service for delivery, as described earlier, a specific delivery process can be picked that is right *just* for that service. This might be extremely agile using XP, or slightly more formal and measured using xUP or DSDM. The purpose here is in small projects with defined bounds to deliver functionality quickly and accurately and then move to the next project, not to have one project that continues on and on.

Aims Of An SOA Programme

One of the first elements that Services help with is splitting the project into a number of phases, and enabling the estimation of those services to be done individually. This helps divide the work up, and makes both top down and bottom up estimating much simpler. By using the Classification for Delivery approach described earlier it is quickly possible to split the estimation tasks between their expected delivery units. If the best delivery method for a given service hasn't been decided this can be done by creating separate estimates, potentially involving competitive tendering, to determine the best solution. By focusing down at this level there is a greater degree of accuracy, and a greater ability to track that accuracy during the project. By working with the project leads, agreeing the most appropriate mechanism for estimation and delivery and mapping out clearly the dependencies between areas the programme manager has the information needed to plan the most effective overall programme plan.

When using metrics driven estimation tools such as "SLIM"[19] it is important to consider the level or parallel development that can be under-taken by using a Service based approach. Because Services are able to define discreet areas the key to understanding their impact on other services is to understand how much the service contract, its interface and associated elements, will change during the project. If the contract is fixed the service development can run perfectly separately once it has been defined, in most cases however there is a certain level of contract change during a project, and the invocations between services, the capabilities, detail the dependencies that exist. What this does mean however is that a services based programme can normally ramp resources up quicker, and sustain more individuals on a project without reducing productivity.

An advantage of a services approach is that different services can be delivered to "different heartbeats" which helps to scale down projects to meet the agile 'sweet spot'[20], this is focused on driving up developer productivity, and thus on driving down cost.

Planning An SOA Programme

Planning an SOA programme is broadly a three stage process, and to be clear here this does not mean that you can throw all good project management practice out of the window, just that an SOA programme has three clear stages in which there are different priorities and focus.

1. Initiation –
    a. Defining the services
    b. splitting down of the programme into its service projects
    c. dependency analysis of the service projects
    d. programme plan
    e. set-up of programme and project environments
2. Delivery –
    a. Monitoring of service projects
    b. Integration of the service projects

    c. Testing and Verification of services
    d. Roll-out of competed services
3. Closedown
    a. Analysis of services against strategy

The key is to think of the work as a programme and then monitor the individual service projects. It doesn't really change either how projects or programmes are managed but changes the emphasis on what a project is and when programmes should be used. This is no different to planning any large programme of work which delivers in multiple phases, the key with Service architecture is that it enables these skills to be used more often to help ensure projects remain an optimum size for productivity and cost.

14

Using A Service Architecture In IT Support

One of the most overlooked areas where SOA can help is that of using Services to better manage existing IT estates. Areas such as the management of services once deployed are beginning to be covered, and therefore will not be covered here, but the area of applying service orientation to the support of legacy systems is something rarely considered. This is a shame as it is one of the quickest wins for any company with a large legacy estate.

Part of the reason for the benefits is the way that most organisations cost the support of their systems. While there are many variations in method and approach to estimating the number of "Full-Time Equivalents" or FTEs required to support a system there tends to be one common thread, namely that this is estimated against a system as a whole. This tends to lead o a number of key elements that can drive up the cost of that support.

1. System availability is driven by the most available area of functionality
2. Quality, response times etc, of the support desk is driven by the most valuable area of functionality
3. Bug and Issue tracking is prioritised at the system level
4. Patching and upgrades is done at the system level

While there has been some work at the transition away from monolithic hardware, there has not been a similar trend away from monolithic support.

Using Services To Align Support Costs

The first way that Services can helping in support is in the same way as they help in delivering, namely aligning people and focus in the same way as the business intends to interact with them. This means logging calls, faults, change requests and help requests against the business domain first, rather than directly against the system.

Making this change will help to identify the business drivers against the systems more clearly rather than viewing systems as drivers in themselves. Ultimately you can use these services, and their business values, to provide specific levels of service to people from different areas of the business who may use the same system for different purposes. As an example, for a key operational system which is used by people who interact directly with clients as part of the core business operation, for instance sales, and also by people using that information to obtain reports for future work, for instance marketing, and by people who are only interested in information reports, e.g. Compliance. Treating these areas differently will help you tune the amount of people required to support the systems, for instance focusing training people on the sales team, and the timeliness of the responses required, marketing might be okay as long as they get the information in the next week.

Changing the support model to become business service, rather than system, oriented is about making support directly related to the business value of the interactions, rather than summing this information and ascribing it to a single system.

Using Services To Reduce Direct Support Costs

Take as an example a set of systems which currently run on a large server. In a desire to reduce costs the company wishes to move it from the large server onto a number of smaller servers running in a grid. The question is whether thinking in terms of

services could enable greater savings and opportunities. Assuming that these systems are not one single executable but a series of series of CICS transactions, or related applications, which cover four basic areas, operations, sales, finance, reporting and that these elements share a common database, and on occasion invoke each other. The other assumption is that these areas can be configured to run on separate servers if required. Given those assumptions the following could be considered instead of a simple monolith to grid shift.

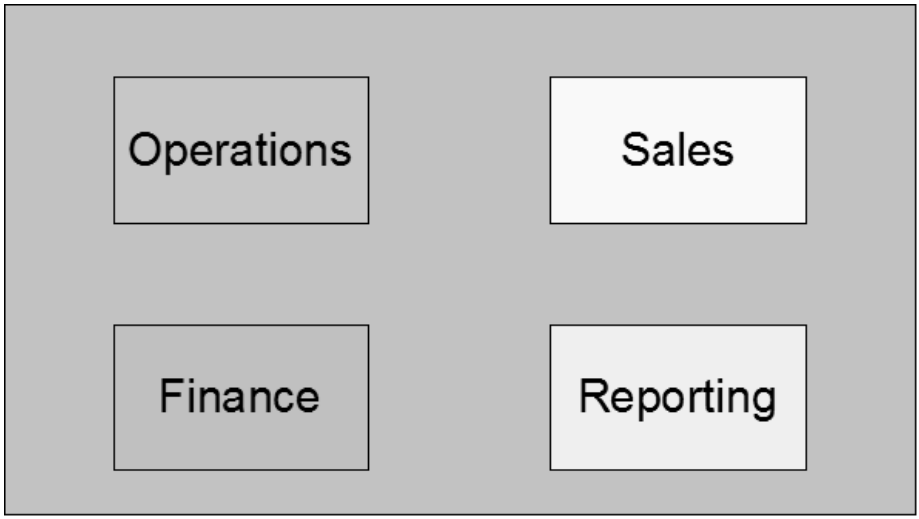


Figure 56: Monolith Server with 4 applications

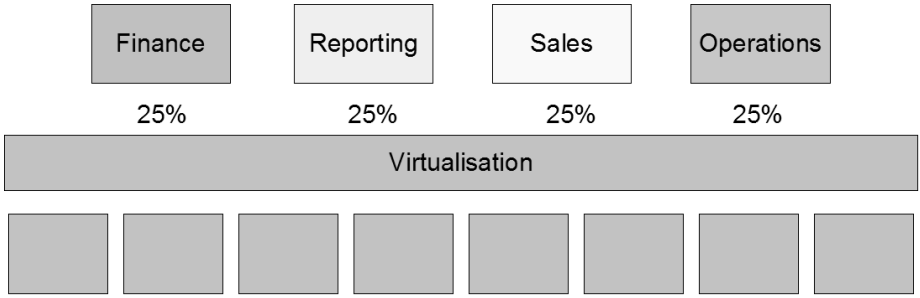


Figure 57: Standard Grid with 4 applications

Firstly understand the services that these areas represent to the business, secondly understand the value that the business ascribes to each on, and thirdly understand the specific support requirements for each of those services. While a standard grid model may elect to give different slices to different applications, often based on their previous capacity requirements, this might not be the most appropriate solution based on the business drivers. Use the opportunity of the migration to understand the different SLAs of the individual elements and use this information to not only set the capacity, but also the priority of that capacity. In this case for instance while Reporting could be allocated 25% of the grid, its priority may be much lower, meaning that in the even of high load on a higher value service, such as Sales, its actual share of the grid will be reduced in favour of an increase in capacity for Sales.

Again linking this back to the original business service architecture and the business value ascribed to the services ensures that money is being correctly spent and can be justified from a business perspective.

Clearly there are other opportunities in support around capacity planning and future changes, but those are already well understood areas, business service architecture just provides them with a clearer perspective and provides a common framework throughout the lifecycle.

15

Summary

This book has concentrated on the start of the service architecture, the most important element. And has briefly explained how other elements can then benefit or be linked in from this approach. The most important thing is to get the *services* right and ensure they represent what the business wants. This then enables other parts of the organisation and delivery to work around a common framework, thus reducing the need for continual alignment between these areas. This book has shown how Service Orientation can impact all areas of business and IT delivery, but has not tried to replace best practice in those areas, merely to demonstrate how Services enable a different set of choices and approaches.

If you are serious about services architecture then it's important to create the services well before you think of technology. Remember what ***Service*** Oriented Architecture is about…

"It's the Services Stupid"

About The Author

Steve Jones is currently CTO of Application Development Transformation at Capgemini, based in the UK. His background is originally in extremely large scale systems in the Air Traffic Control industry and has since worked in almost every sector and continent available. Steve is Capgemini's executive sponsor for their membership of the OASIS and Java Community Process standards bodies and takes an active role in those bodies. A member of the original JAX-RPC group and Mobile Web Services group Steve has been arguing since 2000 that SOA is about more than web-services. Published in several conference proceedings and in the IEEE Software Journal on "Toward an acceptable definition of service". Steve was educated at the University of York and received a BEng(Hons) in Computer Science. Steve continues to present at various conferences, including joining Charles Beckham on stage during the JavaOne 2006 keynote. Steve is currently a member of the JavaSE 6 and JBI groups in the JCP and the OASIS SOA Reference Model group. He currently blogs on Service Architecture and technology at http://service-architecture.blogspot.com.

Table Of Figures

Table Of Tables

Endnotes

1 http://www.oasis-open.org/committees/download.php/15071/A%20methodology%20for%20Service%20Architectures%201%202%204%20-%20OASIS%20Contribution.pdf

2 S. Jones “Toward an acceptable definition of service” IEEE Software May/June 2005 (Vol. 22 No. 3) pp 87-93

3 P. Kroll and P. Krutchen “The Rational Unified Process Made Easy: A Practitioners Guide to the RUP”, Addison Wesley, ISBN 0321166094

4 K. Beck “Extreme Programming Explained – Embrace Change”, Addison Wesley, ISBN 0321278658

5 OASIS Blueprints group http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-blueprints and OASIS Reference Model group http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

6 OASIS SOA Reference Model - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

7 OASIS SOA Reference Model Public Draft - http://www.oasis-open.org/committees/download.php/16628/wd-soa-rm-pr1.pdf

8 Steve Jones, Steve Meyfroidt: Web Services and Java (Position Statement). SAINT 2003: 10-13

9 Jones & Meyfroidt “Mobile Web Services”. JavaOne 2002

10 G. Geurts and A. Geelhoed “Business Process Decomposition and Service Identification Using Communication Patterns” – MSDN Journal 1 ,January 2004

11 Bertrand Meyer: *Applying "Design by Contract*, in *Computer (IEEE)*, vol. 25, no. 10, October 1992, pages 40-51

12 Proceedings. Thirteenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 14-16 June 2004

13 NFL challenge and timeout rules
- http://www.nfl.com/news/990526replaytechnology.html

14 A. Mulholland – "Moving from Big to Small"

15 P.A. Strassman – Business Value of Computers, ISBN 0-9620413-2-7

16 F. Brookes – "The Mythical Man Month: Essays on Software Engineering", Addison-Wesley, ISBN 0201835959 (1975, republished 1995).

17 DSDM - http://www.dsdm.org/

18 AUP - http://www.ambysoft.com/unifiedprocess/agileUP.html

19 SLIM estimation tool - http://www.qsm.com/slim_estimate.html

20 P Kruchten "Scaling down projects to meet the agile 'sweet spot'" - http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/aug04/5558.html

FREE ONLINE EDITION

(non-printable free online version)

If you like the book, please support
the author and InfoQ by
purchasing the printed book:
http://www.lulu.com/content/408923
(only $27.95)

Brought to you
Courtesy of
Steve Jones &

This book is distributed for free on InfoQ.com, if you have received this book from any other source then please support the author and the publisher by registering on InfoQ.com.

Visit the homepage for this book at:
http://infoq.com/books/enterprise-soa