Contents and sample chapters for

The service-oriented enterprise
Enterprise-architecture and viable services

Tom Graves
Tetradian Consulting
Acknowledgements

Amongst others, the following people kindly provided comments and feedback on the early drafts of this book: Daljit Banger (White Knight Consulting, GB), Shawn Callahan (Anecdote Pty Ltd, Aus), Charles Edwards (ProcessWave, GB), Geoff Elliott (St Neots, GB), Nigel Green (CapGemini, GB), Charles Millar (Marlow, GB), Kim Parker (AusPost, Aus), Liz Poraj-Wilczynska (Brockhampton, GB), Peter Tsegakof (AusPost, Aus), Jaco Vermeulen (Chaos Based, GB).

Please note that, to preserve commercial and personal confidentiality, the stories and examples in this book have been adapted, combined and in part fictionalised from experiences in a variety of contexts, and do not and are not intended to represent any specific individual or organisation.

Registered trademarks such as Zachman, TOGAF, FEAF, ITIL, ASL, BiSL etc are acknowledged as the intellectual property of the respective owners.
INTRODUCTION

The service-oriented enterprise

Everything in the enterprise is a service. Everything the enterprise does is a service. Everything the enterprise delivers is a service – even products are services, in a sense. And the enterprise itself is a service. *Everything* is a service.

That’s the key idea behind the *service-oriented enterprise*: a view of the enterprise in which everything is seen in terms of services and their interactions and interdependencies, providing consistency and simplicity everywhere, and creating new space for agility and innovation in the enterprise.

This book explores why and how to put that idea into practice, with real business benefits across the entire enterprise.

Who should read this book?

The book is intended for enterprise architects and others who deal with the practical implications of whole-of-enterprise issues.

It should also be useful for strategists and service-managers, and for anyone who works with ‘pervasive’ themes such as supply-chains, value-webs, quality, security, knowledge-sharing, business ethics and social responsibility, health, safety and environment.
What’s in this book?

The text is divided into three main sections:

• **Basics** (see *Basics – An overview*, p. 4) – reviewing key terms such as ‘enterprise architecture’ and ‘service-oriented architecture’, and the core metaphors underlying the service-oriented enterprise

• **Principles** (see *Principles – An overview*, p.27) – describing core ideas about the structure and relationships between services, and their relationship to the structure of the enterprise

• **Practice** (see *Practice – An overview*, p.67) – illustrating how to apply the principles in real-world business contexts, using a Functional Business Model as the anchor for the enterprise service-map, process-tree, costing-model and other variants

Each section contains several chapters, as smaller chunks to apply straight away in your day-to-day work. Although there’s a fair amount of theory, the keyword here is *practice*: the aim is to give you something that you can *use*.

So each chapter includes examples and stories to place the ideas into a real-life context, with references to other relevant resources. Most chapters end with an ‘Applications’ section, with questions to help you apply the material within your own context; and a ‘Resources’ section, pointing to sources for further information. There’s also a glossary at the end of the book, which should help in clarifying the broader meaning of some of the common terms used in the architecture of the service-oriented enterprise.

But first, what *is* the service-oriented enterprise? To answer that, we need to explore some basic terms such as ‘enterprise architecture’ and ‘service-oriented archi-
tecture’; and, behind that, the underlying metaphors that we use to describe the enterprise.
BASICS – AN OVERVIEW

Before we get into the meat of the issues, there are a few background items we need to address.

One of these is the role of enterprise architecture (see Basics – Enterprise architecture, p.6). At present, this field is often regarded as a subset of IT, and specifically of IT governance. But it makes far more sense if we extend it outward to its more literal meaning as the architecture of the enterprise as a whole.

Enterprise architecture is about the structure of the enterprise – how everything fits together in support of the enterprise vision and values and shorter-term goals. Viewing an enterprise entirely in terms of services is one of the more valuable ways to assess that structure, especially as it provides consistency between every part and every level within the enterprise.

A service-oriented architecture also starts to make more sense from that point of view (see Basics – Service-oriented architecture, p.14). As with enterprise-architecture, it initially arose as a way to resolve issues around detail-level IT: but here we expand its scope outward, as an architectural style to understand the entire enterprise in terms of service structures and content, and their mutual interactions and interdependencies.

As part of this, we’ll also need to introduce some basic themes and terminology from formal systems-theory. This isn’t well-known in business as yet: unfortunate but understandable, because much of so-called ‘hard-systems theory’ is described in a complex, arcane mathematics that may well seem impenetrable – and, frankly, irrelevant – to most people in business. The type we need here, though, is the
human-oriented ‘soft-systems’ approach, which is much more approachable, and more easily expressed in business terms for the real business context.

Beneath that distinction between ‘hard-systems’ and ‘soft-systems’ is a fundamental difference in underlying metaphor, in the overall way we describe the enterprise (see Basics – A matter of metaphor, p.26). There’s a key contrast here between the notion of ‘enterprise as machine’ – probably the most common business metaphor since the days of Frederick Taylor’s ‘scientific management’, at the end of the nineteenth century – and the more recent concept of ‘enterprise as living organism’. The mechanistic view has some real value at the detail-level of technology and process-design, but it simply does not work when we try to apply it to the whole enterprise – as many business-folk have discovered the hard way, in expensive debacles such as the failure of so many attempts at IT-based ‘business process re-engineering’. Instead, if we are to gain the full value from a service-oriented architecture, we need to shift towards the more holistic, all-encompassing view of the ‘living enterprise’ model.

Enterprise architecture, service-oriented architecture, and a shift in underlying metaphor: together, these form the basic background for the service-oriented enterprise.

**Resources**

- Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise*, (Tetradian, 2008)
- Tom Graves, *Bridging the Silos: enterprise architecture for IT-architects* (Tetradian, 2008)
Summary

Enterprise architecture is a formal discipline managing a body of knowledge about the purpose and structure of an enterprise. It has its roots in IT systems and technology-management, but has steadily extended its scope towards covering the whole enterprise – and it will need that much broader scope if it is to fully describe the requirements of the service-oriented enterprise.

Details

**What is enterprise architecture?**

What *is* the enterprise? What holds it together? What structures do we need to make it work better, to help it adapt to its changing circumstances and business environment?

These are the core questions underlying the formal discipline of enterprise architecture. It’s still a new discipline: the term itself was coined barely a decade ago, building on previous ideas about ‘information systems architecture’ and the like. There are several standards and formats – TOGAF (The Open Group Architecture Framework), FEAF (US Federal Enterprise Architecture Framework) and the Zachman framework being some of the best-known examples. Which leads us, though, to ask: what *is* an ‘enterprise’? For that matter, what is meant by ‘architecture’ in this context?
An all-too-common answer to that last question – typified by the comments of the architecture manager at a recent client – would be “IT functions and data”. When I explained that the business of her employer – a government department in the social-services sector – was a great deal wider than that, and hence her architecture needed to cover people-based processes, legislation and a great deal more, she replied, “Yes, I agree with you: architecture is all about IT functions and data”. We went round that loop a couple more times before I gave up: there wasn’t much point in trying to discuss real enterprise-architecture with someone who wouldn’t allow herself to hear about any aspect of business beyond her own domain of IT. Oh well…

In every form of architecture – building architecture, naval architecture, process architecture, business architecture, whatever – the real focus is on structure, and the use of that structure. In the architecture and design of a hospital, for example, we would be as concerned with the workflows and other usage-patterns within the hospital as its physical framework of walls and wiring, passageways and plumbing. The same is true of enterprise architecture: it’s about structure – any kind of structure within the enterprise, from data-definitions to downpipes and dumper-trucks - and the principles and guidelines that govern the use of that structure.

So what then is an enterprise? According to the FEAF specification, it’s any kind of entity “supporting a defined business scope and mission”, within which the various members or components “must coordinate their functions and share information” to achieve that purpose. The ‘enterprise’ might be a commercial business; it might be government or non-profit; it might be the local football club or the village New Mothers group; it actually doesn’t matter what the size or purpose might be, as long as there is a defined scope and mission to share.

Note, though, that whilst an organisation is a kind of enterprise, not every enterprise is delimited by the legal and other boundaries of an organisation. An enterprise may be any subset or superset of the organisation: the IT department is an enterprise in its own right, within the overall enterprise of the parent organisation; a multi-partner cross-industry consortium is likewise an enterprise made up of a
shared ‘mission and scope’ to which a group of smaller enterprises choose to align. There’s no inherent restriction there: and in some cases the effective boundaries of the enterprise may well change from minute to minute.

Under those circumstances, governance of the enterprise can get complicated, to say the least. Which is where enterprise architecture comes into the picture:

**Enterprise architecture is a discipline through which an enterprise can identify, develop and maintain its knowledge of its purpose, its structure and itself.**

Enterprise architecture identifies and monitors the structures needed to support that shared mission; it assists in managing change, and responses to change; it provides guidance and consistency across the whole scope.

But here we may hit up against a problem of divergence between what is commonly thought to be the scope of enterprise architecture, and the real scope it needs to have. We need to address that issue of scope before we can move on.

**The scope of architecture**

In most current descriptions, ‘enterprise architecture’ is associated almost entirely with IT – in effect, the term is a kind of shorthand for ‘enterprise-wide IT architecture’. But with each new stage of maturity, the scope has needed to expand further, from low-level technology and interfaces, to structures for data and applications, and then to a somewhat belated recognition that all of this needed to be linked to and driven from business strategy.

Yet all of this is still centred solely on IT: much so-called ‘business architecture’, for example, is often just an IT-oriented summary of ‘everything not-IT’. So a typical view of the scope of that ‘enterprise architecture’ would be as in the TOGAF specification:
The IT-architecture hierarchy

<table>
<thead>
<tr>
<th>Business Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Architecture</td>
</tr>
<tr>
<td>(Information-System)</td>
</tr>
<tr>
<td>Applications Architecture</td>
</tr>
<tr>
<td>Technology Architecture</td>
</tr>
</tbody>
</table>

Note the relative sizes of each area of interest in that diagram. In the original TOGAF-8 methodology, for example, assessment of low-level technology is assigned more than six times as many development steps as for the whole of the rest of the business put together. Kind of an imbalance there…

Every enterprise is different, yet very few are as IT-centric as the standard enterprise-architecture models suggest. And for the rest? Perhaps the simplest way is to describe each enterprise in terms of four distinct dimensions, which we could summarise as the four classic ways an enterprise distinguishes itself from its competitors:

• through products and services – a physical dimension of actions and transactions, “the way we do things round here”
• through knowledge and innovation – a conceptual dimension of ideas and information, “what we know, how we think”
• through relationships and ‘feel’ – a relational dimension of people and their individual skills and experience, “how we relate with each other, and with others”
• through *vision and purpose* – an *aspirational* dimension about direction, focus, morale, belonging, “who we are and what we stand for”

The FEAF reference-model describes these respectively as ‘Other Fixed Assets’, ‘Technology’ (by which they mean IT, though that’s actually only a subset of knowledge-technology), ‘Human Capital’ and ‘Business’. These dimensions interweave as the distinct and distinctive nature of the enterprise as a whole. So the layering of a *real* enterprise architecture would look more like this:

<table>
<thead>
<tr>
<th>People Systems-Architecture</th>
<th>Information/Knowledge Systems-Architecture</th>
<th>Machine / Asset Systems-Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>People (Relational dimension)</td>
<td>Knowledge (Conceptual dimension)</td>
<td>Assets / ‘Things’ (Physical dimension)</td>
</tr>
</tbody>
</table>

*Whole-of-enterprise structure*

So enterprise architecture isn’t about IT – or rather, since IT does play a part in most present-day enterprises, it’s not *only* about IT, but about how *every* aspect of the enterprise interacts with everything else. And it isn’t centred on IT: it isn’t centred anywhere, other than on that guiding theme, the ‘defined business scope and mission’. In short:

**The scope of enterprise architecture is always the whole of the enterprise – the enterprise *as a whole***
We may choose, for practical reasons, to set suitable limits on the scope of an architectural assessment or design. But the moment we forget that that’s an arbitrary choice, not a fact, we’d be in trouble straight away: the real scope – always – is the whole of the enterprise.

**The purpose of architecture**

But what’s the point of all of this? What’s the *purpose* of enterprise architecture? To answer that, we need to return to the FEAF definition, that an enterprise has various members and components that coordinate their functions and share information to express or achieve their shared ‘mission’ or vision. The architecture provides a means to support that process of coordination; it provides consistency across the whole of the enterprise scope, and a consistent *description* of that scope.

That’s the function of reference-frameworks: TOGAF’s ‘Integrated Information Infrastructure Reference Model’ is one such generic example for the IT industry, whilst eTOM and SID (‘enhanced Telecom Operations Model’ and ‘Structured Information/Data’) are a matching pair for telecommunications service-providers, and SCOR (Supply-Chain Operations Reference-model) describes logistics and other supply-chain services.

But these are all fairly static models: they describe structure, but not much about interactions between the components of those structures. And they don’t explain much about the *human* side of the systems – such as the so-called ‘tacit’ knowledge that resides only in people’s heads, and is most often shared through stories and action-based learning. For these, we need a broader view of the structure, the system and the scope.

That’s what a ‘services’ view of the architecture can provide for us. Services present a consistent frame through which to understand structure; and service-
choreography describes the use of that structure. Which leads us, in turn, to a service-oriented view of the enterprise as a whole – the service-oriented enterprise.

That in turn would depend on a view of ‘service-orientation’ that’s as broad as the scope of the enterprise itself. But as with ‘enterprise architecture’, the term is at present strongly associated with IT alone – too strongly for our needs here. To make this work, we need first to take a detour to find the real meaning of ‘service-oriented architecture’.

**Application**

- What are the bounds of ‘the enterprise’ in your own business context? Are they a subset of the organisation; synonymous with the organisation; or do they extend beyond the organisation itself? What are the dynamics of those bounds – in what ways do they change, and for what purposes?
- Does your organisation maintain some kind of enterprise architecture? If so, what is its current scope? Where within the organisation does it reside? Who has responsibility for its scope and governance?
- If your existing enterprise-architecture is restricted to an IT-centric scope, what would need to be done to extend that to a true enterprise-wide scope? What can you do to assist in that change?
- What support does enterprise-architecture provide to enhance consistency across the enterprise? In what ways could that support be further enhanced within your enterprise?

**Resources**

✨ TOGAF (The Open Group Architecture Framework): see www.opengroup.org/togaf
Zachman framework: see www.zifa.com
Agile architecture development: see www.agileea.org
Whole-of-enterprise architecture: see Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise*, (Tetradian, 2008)
Adapting existing architecture-tools for whole-of-enterprise architecture: see Tom Graves, *Bridging the Silos: enterprise architecture for IT-architects* (Tetradian, 2008)
BASICS – SERVICE-ORIENTED ARCHITECTURE

Summary
As with enterprise architecture, service-oriented architecture has to date been associated almost exclusively with low-level IT. We can break free from that restriction, and open out to the whole scope of enterprise architecture, by recognising that everything in the enterprise provides a service.

Details

Services and IT-architecture
If we read the existing literature on service-oriented architecture – the Wikipedia summary, say – we’ll find the same as with enterprise architecture: it seems it’s all about low-level IT, and nothing else. And as with enterprise architecture, it’s not the whole story: the IT is only a small part of the enterprise. Yet it’s often the only part that most people know, so it’s probably the best place to start: we just have to remember to translate any IT-centric assumptions to the broader whole-of-enterprise scope.

In IT-architecture, a service is a composite made up of small blocks of functionality that collectively present a unified interface – often but not always a single interface – to the ‘external’ world, and are otherwise autonomous, ideally with no ‘hard-wired’ links to any other services. Some of the other key ideas include:
• **loosely coupled** – services have minimal dependence on each other: “services share schema and contract, not class”, to quote one of the original services theorists, Don Box

• **contract-based** – the service to be delivered is defined in a ‘service description document’, or some other kind of ‘contract’ or service-level agreement

• **discoverable** – the service and its interface-contract – including information on the quality available in the contract, to aid process optimisation – should be identifiable automatically

• **abstraction** – as long as the service delivers in accordance with the specified interface-contract, how it does what it does should be a ‘black box’, internal detail that is (mostly) irrelevant to the user of the service

• **reusable and composable** – the service should be designed for re-use in a variety of different ways and in aggregations into larger blocks of shared functionality

• **interchangeable** – it should be possible to switch between services on different systems and from different service-providers that adhere to the same nominal service-contract specification

Overall, this gives some great advantages:

• we break free from monolithic, hardwired ‘applications’ that are cumbersome, fragile and hard to change

• developers can optimise the internal mechanisms of a service without affecting its exposed interface

• we can ‘mix and match’ between services to create new ‘mashups’ for rapid response to business change

• services can be described in business terms, enabling people without an IT background to create their own personalised mashups
we can call on the facilities of multiple service-providers for the same nominal service, for load-balancing and risk-management, or for business reasons such as reduced costs or faster response.

All good: but there are some significant downsides, which at times may be overlooked amidst the flood of marketing-hype. There’s a sense sometimes of a wishful attitude of ‘build it and they will come’: yet in a true service-oriented architecture, services have no inherent context, they just are. So issues around ‘discoverability’ – self-advertising, in effect - suddenly become non-trivial: a concern that barely existed with the old monolithic applications, where every function was hardwired into the run-time code.

Also unlike those monolithic applications, there’s no inherent built-in mechanism that links services together into processes: that choreography has to come from somewhere beyond the services themselves. And the need for service-contracts can bring a hefty overhead in terms of bandwidth and execution-time, especially where service-connections need to be created in real-time.

A service-orientation for IT-architecture enables new options, new possibilities: one such example is ‘cloud-computing’, in which IT-services can be exchanged and accessed from anywhere across the web. Yet all of this is still solely from the IT perspective. Once we look wider, and extend the scope of a service-oriented architecture, we start to see that there are other advantages, and other challenges – some of them subtle, even strange, perhaps, but all of which do need to be addressed.

**Security and other service challenges**

Probably the greatest challenges to a service-oriented architecture revolve around security. Every access to anything will require some kind of identity-management:

• is this requester who they claim to be?
• if they are, do they have the appropriate ‘need to know, need to use’?
• if it’s a commercial service, do they have access to the funds to pay for the service?
• how do we confirm all of this, within the response-time specified in the service-contract?

In a typical monolithic application, we would control access to information or functions via some kind of token. Once someone has logged into some system, we can use that token to track anything they do and ask for within the bounds of that system, and respond accordingly. Often all we’ll need to do is something like a lookup to a list of passwords and a list of matching access-rights, and we’d be all but done: problem solved. For the enterprise as a whole, almost the only complications are the need to get people to change their passwords from time to time, to reduce the risk that security may be compromised; and the need to align access-controls for the various different monolithic systems that people need to use – hence the ‘holy grail’ of a so-called ‘single sign-on’ for every system, which sounds simple enough in theory, if not so simple in practice…

But in a service-oriented model, identify-management is suddenly much more complicated – especially on the internet, which has no built-in mechanism to track ‘stateful’ concerns such as identity and security. ‘The system’ – such as it is – consists of layer upon layer of near-arbitrary collections of “small pieces loosely joined”: we can pass around a token, perhaps, but as yet there’s no quick, simple, automatic means to link that token to any central register of access-rights. And once we’re outside the enterprise firewall – as we would be with most cloud-computing services, for example – we have to deal with all the security nightmare of the internet, with hacking, spoofing, interception, forged identity and the rest: it’s not a pretty picture.
The technical challenges of SOA are almost trivial compared to its security challenges: there’s good reason why all of the recent Open Group conferences on enterprise-architecture have also featured a major stream on security and service-architecture. The catch is that many of the issues are have little or nothing to do with IT – a fact which can be difficult for the many people who turn up at such conferences with an IT-only background, a fixed IT-centric view of the business world, and an equally fixed assumption that every possible problem has an IT-based solution… But unless we do think wider about security than just the IT, we’re likely to cause more problems than we solve: when security is our responsibility, we’re responsible for all of it, not just the easy part that’s well within our comfort-zone!

Another classic problem in service-choreography is what’s known as ‘deadly embrace’. An example from the hardwired computer-systems of a few decades ago would be where one service gains exclusive access to a datastore, and then requests the printer, at the same time as another service locks the printer for its use and then requests the datastore. Each service is then stuck waiting for the other: which would not be popular with the services’ users...

In principle, and in direct form, at least that kind of problem has long since been resolved with tactics such as asynchronous printer-queues and optimistic record-locking; and the sheer speed of most present-day data-processing also helps. But each service-transaction takes time to set up, especially through all those worldwide routing-mechanisms of the internet; and when we have to wade our way through layer upon layer of access-control services and suchlike to get anything done, the total set-up time may well exceed that for the actual transaction many times over. And that’s assuming that everything works: it’s not always obvious what will happen when any service-transaction fails within an overall choreography. So even if it’s not a true ‘deadly embrace’ in a technical sense, it may well feel like it… And when such a widely-distributed system fails in that way, it’s often far from obvious as to whose responsibility it would be to fix it: which is guaranteed trouble at a
business level. So cloud-computing may sound almost perfect in principle, perhaps, but may not be so perfect in practice.

And then there are a whole swathe of other issues that have little or nothing to do with IT as such, but can have serious impacts on IT-based services. For example, almost every country has its own distinct and different rules around privacy, record-keeping and the like. So whilst in principle data could be stored anywhere in the world, many countries mandate that some (but only some) information should never cross national borders – which means that service-users must have proof from service-providers that that is indeed the case.

Other countries – particularly the US – assert legal concerns about restricting information to their own citizens only – who may, however, be anywhere in the world. Then there are the thorny issues around pornography and the like, where access may be limited not by nationality but by age, or other personal attributes. All of which lead to chaotic confusions about who can and can’t see what, and who has authority and responsibility to control such limitations – let alone, again, dealing with all the hackers and spoofers and suchlike. Messy indeed.

A true service-oriented architecture needs to be able to deal with every one of these issues. Which is not straightforward at all. Which is why a service-based view of the enterprise needs also to be strongly linked to a solid structure for service-management – not just for IT, but for the whole of the enterprise.

**Services and service-management**

Whether in ‘cloud-computing’ or in old-style applications, IT services need to be linked together into end-to-end business processes – which means there needs to be something above the services themselves to create that choreography. In the same way, there also needs to be an IT infrastructure on which the services will run. The
latter would be more the realm of IT-architecture, perhaps, but all of this would come under the general heading of ‘service management’.


ITIL v2 uses a classic IT-centric ‘them and us’ model for its overview. On the far left there’s ‘The Business’; on the far right, ‘The Technology’ Between them sits the core of service-management, Service Delivery and Service Support, with an interface of ‘The Business Perspective’ on one side and ICT Infrastructure Management on the other. Above and below we have a useful focus on planning, on application management, on ever-essential security-management. That’s it, apart from an emphasis throughout on ‘IT/Business alignment’ – though it’s never very clear about who is supposed to align with whom.

ITIL v3 is radically different: there’s almost no IT-centrism at all. In principle it’s all about service-management for IT, but apart from a few low-level technical details we could remove the term ‘IT’ throughout and it would still make perfect sense. It starts with Service Strategy; moves on to Service Design; then Service Transition, about implementing those designs; and Service Operation. Interweaving all of those is Continual Service Improvement. Each is “a set of specialised organisational capabilities for providing value to customers in the form of services”, with an emphasis on managing the overall service lifecycle as much as service delivery and support. And though never stated explicitly, perhaps the real key point in ITIL is this:

Service-management is itself a service: it delivers the services of service-management
The ITIL standard describes those service-management services, and the relationships between those services, both in terms of their interdependencies and choreography, and in terms of the mutual responsibilities over time. But it also helps us to think about services in much broader terms than those of IT architecture alone. Which matters, because ultimately, everything is a service.

**Everything is a service**

Whatever it looks like, and whatever form it may take, *everything* in an enterprise delivers a service. That’s the real implication of a service-oriented architecture; and also the real reason for its importance to the enterprise.

Everything is a service. Even products are proto-services, in a sense, because they provide the end-customer with the means to deliver a self-service: a vacuum-cleaner provides the service of cleaned floors, the grocery-shopping provides the self-service of meals, and so on. “Customers do not buy products”, says the ITIL v3 specification, “they buy the satisfaction of particular needs”. And we satisfy those needs through the services we provide.

Shifting the perspective from products to services can be more profitable, too. Interface Inc is one of the world’s leading manufacturers of flooring materials, and some two decades ago, in the words of its founder Ray Anderson, was consuming “an ocean of petroleum” each year to make its products. It was a profitable enterprise, if perhaps not as profitable as it could have been – though with the ‘business as usual’ mindset of the time, it was far from clear as to what needed to change to make that quantum shift in profitability. But in the late-1990s Anderson had what he called “a spear to the chest” – a kind of double-epiphany about sustainability. One side of it was a recognition that every scrap of waste – at that time a huge problem for the corporation – was something that they’d paid for but couldn’t sell. Hence zero-waste was not merely a ‘feel-good’ goal, but a business matter with serious business impacts: waste-reduction leads directly to more profit, and *everyone* wins. At the present time, Interface is about halfway towards its ‘Mission Zero’ goal for the year 2020 – zero waste, zero
environmental impact, everything recycled – and its profitability continues to climb with each passing year.

Anderson’s other ‘epiphany’ was a realisation about a link between products, services and waste. Interface had sold carpet and other flooring products; but what its clients really wanted – ‘the satisfaction of a particular need’ – was the service of covered floors: and to the end-customer, that service had higher value as a service. So Interface started to shift their business that way – and discovered that to make it work, they also had to shift their attitude to waste. When the company was oriented towards products, it was in their interest to get their customers to be careless about waste: more waste equals more product sold, and built-in obsolescence seems like a good idea, too. But it’s the other way round when the company embeds the product in its own fixed-fee service: suddenly it’s in the company’s interest for customers to minimise their waste, and for products to be as durable as possible. So for the service-oriented enterprise, sustainability is good business in every sense – and once again, everyone wins.

In IT service-oriented architecture, we rethink IT functions and the like in terms of services. But in the service-oriented enterprise, we take that idea a stage further, and rethink everything in service terms. Service-management is a service; management is a service; so is strategy, architecture, the office canteen. Everything in the entire enterprise is a service.

In ITIL v3, a service is defined in rather abstract terms, as a means of delivering value by facilitating the customer’s desired outcomes. The service is both valued and valuable because we take on some of the complexity and risk on the customer’s behalf. And the service is accorded a value by the customer to the extent that it makes those outcomes easier to achieve.

Product is about ‘what’; service is about ‘how’ and ‘who’ and ‘where’ and ‘when’; but what the customer is really interested in is their ‘why’ – and often don’t greatly care about how that ‘why’ can be achieved, as long it is achieved with the minimum of difficulty and fuss. As a service-provider, that’s our desired outcome: a satisfied customer. In a commercial context, satisfaction matters – because a satisfied customer is likely to come back for more.
Incidentally, this is one point where ‘black box’ encapsulation of service-internals may cross over with demands for transparency, giving a much more subtle definition of ‘customer satisfaction’. When the Australian ‘Ghan’ railway line from Adelaide to Darwin was upgraded some years back, one of the tenders was presented by a Chinese consortium. Unlike the other tenders, which assumed a relatively small workforce and large amounts of heavy machinery, the Chinese proposed instead to use a vast manual workforce of half a million labourers. The delivered service-results would have been exactly the same, created to the same nominal service-level agreement. The Chinese proposal even worked out cheaper – even allowing for the logistics of serving that workforce in difficult near-desert conditions – hence would have been much ‘better’ than the other tenders if the price had been the only criterion.

But an end-customer may well be concerned with more than just the end-result. That was certainly true in this case: the politics were way too scary, for a start – especially as that imported workforce would represent a significant percentage of the country’s entire population. So sometimes the simple black-box encapsulation can risk reducing ‘customer satisfaction’. In an IT service-oriented architecture we might not need to take account of such issues, but at the whole-of-enterprise level we certainly do.

Something else to think about, anyway.

In the service-oriented enterprise, every activity has an explicitly-identified customer to whom that service has value; and each of those customers has an outcome that they want to achieve. Those two points define the service-need, and hence the structure and nature of the service.

Perhaps more to the point, if we can’t identify the customer and their need, we’re probably delivering a service that has no value – a situation which is far from uncommon in larger enterprises, particularly in the myriad of performance reports created with so much effort but that no-one ever actually reads…

Do beware, though, of the common mistake of attempting to define that value solely in monetary terms. We can’t eat money; we can’t travel with money alone; we can’t get a hug from it, or a greeting home; it doesn’t even make a good floor-covering. Money is a means to an outcome – not the outcome itself. To make a service-oriented architecture work, we need to be clear about what that outcome is – not just the means by which we get there.
Products enable services; and services in turn enable outcomes. As a service-provider, delivery of the service is the endpoint of all our work; but from the customer’s perspective, the service is only a means to an end – it’s not in itself their intended outcome. To understand the service-need, we must describe that outcome in business terms, and usually distinct from the service itself. To use an example from the ITIL v3 specification, we ourselves might think of our service as delivering ‘an online sales-information process’ or suchlike; but what our service-customer actually wants is something that will enable their sales-people to spend more time face-to-face with their own customers. That’s the real need; that’s what we need to design our service for; that’s the real measure of ‘success’ in service terms – a satisfied customer. How we get to that desired outcome is up to us: to a large extent, the client doesn’t want to know, as long as the outcome is achieved.

Which brings us back to those core principles of service-oriented architecture: loosely-coupled, contract-based, discoverable, interchangeable, abstraction and so on. Those support best-value for the customer. Other principles support best-value for us as service-providers: particularly reusability, compositability and, again, a layered abstraction. But these depend in turn, on that fundamental shift in perspective: that everything in the enterprise is a service.

It’s not an easy shift in perspective, perhaps, but the payoffs can be huge – especially in terms of simplicity, clarity, resilience and speed of response to change. Yet to make it work well, we need another parallel shift in perspective, about the way we view the enterprise as a whole. And that shift is what we need to look at next.
Application

• What to you is ‘service-oriented architecture’? How would you apply such ideas in your existing enterprise?
• How do you address the security and identity-management issues in service-oriented architecture? And the synchronous-transaction issues?
• What structures and processes do you use for service-management? Do they only apply to IT-services, or to other services as well?
• Who are the clients for your services? How do you identify and model their business needs and business outcomes? How do you ensure that your services deliver against those outcomes?
• What distinctions do you currently draw between products and services? Who are the users and ‘consumers’ of each? In what ways do those users differ, in terms of expectations, responsibilities and so on?

Resources

✨ Service-oriented IT-architecture: see Wikipedia summary at en.wikipedia.org/wiki/Service_oriented_architecture
✨ ITIL (Information Technology Infrastructure Library): see www.itil.org.uk and www.itil-officialsite.com
✨ BiSL (Business Information Services Library) and ASL (Application Services Library): see www.aslbislfoundation.org
BASICS – A MATTER OF METAPHOR

Summary
The most common metaphor for the enterprise is to view it as some kind of complicated yet controllable machine. But to succeed with a whole-of-enterprise architecture that is based on services, we have to shift our perspective, and view the enterprise more as a kind of living organism in its own right, with its own self-chosen direction and purpose.

Details

Machine and organism
[see published book for further details]
PRINCIPLES – AN OVERVIEW

The core principles of service-oriented architecture and design are well-documented from the work on IT-services:

- encapsulation of sub-components into bundles of more usable functionality
- loose-coupling via messages and standards
- service-contracts between ‘requester’ and ‘provider’ roles
- ‘black box’ abstraction of the internal workings of services
- design for reuse and re-purpose
- self-advertising of capabilities and interface needs, to enable ‘discoverability’

As we move beyond detail-level IT, though, other concerns may also come into play. One is that there may need to be multiple implementations of a service, to allow for operational issues as such up-scaling, down-scaling and disaster-recovery. Another is that we need to be aware of the different types of services and how they interrelate with each other – see Principles – The structure of services, p.29. And we also need to be clear about the distinctions between the service-types themselves:

- services that provide the deliverables of the enterprise, or directly support those deliverables – see Principles – Delivery services, p.44
- services that manage the enterprise and its services – see Principles – Management services, p.45
- services that provide the coordination for end-to-end processes – see Principles – Coordination services, p.62
services that promote, protect and preserve the pervasive values of the enterprise – see Principles – Pervasive services, p.64

We need to remember, too, that none of these services exists in isolation: every service has its own set of relationships and interdependencies, all interweaving through the ‘value-web’ that is the whole enterprise. To make sense of all of this from an architectural perspective, we need some means to clarify the complexity. The best way to do this is describe the interactions in terms of architectural patterns, typically derived from formal systems-theory – see Principles – Properties and patterns, p.65.

This combination of structure, categorisation and patterns gives us a framework that we can apply in practice to real everyday architectural issues in the enterprise.

Resources

_services in IT-architecture: see Wikipedia summary at en.wikipedia.org/wiki/Service_Oriented_Architecture
_services in whole-of-enterprise architecture: see Tom Graves, ‘The Viable Services Model: Service quality, service interdependence and service completeness’ in Jan van Bon [ed.], IT Service Management: Global best practices (itSMF/Van Haren, 2008)
PRINCIPLES – THE STRUCTURE OF SERVICES

Summary
Starting from the machine-metaphor notion of a strict separation between management and workforce, ‘brain’ and ‘brawn’, we use Stafford Beer’s ‘Viable System Model’ as a basic structure to describe services and their interrelationships within any layer of the enterprise.

Details

**Brain and brawn**
Architecture is about structure, and about the way that the various components fit together within that structure: so it’s easy to see how most current styles of enterprise-architecture would naturally align themselves with a machine-metaphor for the enterprise. In that sense, the nature of architecture itself is part of the problem here.

Yet living entities have their structure too – though we do need to remember that it *is* a living entity, not a dead machine! So to extend that metaphor a little further, what is the anatomy of a service? Or its physiology?

Perhaps the simplest place to start is the machine-metaphor’s split between ‘brain’ and ‘brawn’. Even the simplest bacillus will show some internal specialisation, and by the time we get to a true stand-alone cell such as an amoeba, there’s some
definite structure there. In particular, there’s a single distinct nucleus somewhere near the centre, which acts much like the entity’s ‘brain’; and a variety of smaller structures such as mitochondria, lysosomes and vacuoles scattered throughout the cytoplasm that provide various support-functions and act as the cell’s ‘brawn’.

The cell itself exists within a broader environment, in which it will serves some purpose, in an ecological sense at least. So in the abstract, hierarchical terms typified by classic ‘scientific management’, we could summarise this visually as follows:

![Diagram](313x376)

′Brain′ and ′brawn′

Extending the ‘scientific management’ view further into the business context, the cell-like service exists to serve the purpose set by the external ‘owners’:

![Diagram](527x361)

*Machine-metaphor model of a service*
In anything larger than the most minimal organism, the cell exists within a hierarchy of larger cell-like structures that replicate on larger scales the specialisations that exist within each individual cell. So too in the enterprise: services are aggregated into larger-scale services, each encapsulating some specific functionality for the enterprise. Each service has its own ‘customers’, and its own role within the overall purpose; and the same is true for every sub-service, and sub-sub-service, and so on.

Yet the layering of services is such that the ‘brain/brawn’ pair of a single service becomes part of the ‘brawn’ for the next layer upward in the hierarchy. So here we meet one of the fundamental flaws in ‘scientific management’: it asserts an absolute split between white-collar and blue-collar, ‘brain’ and ‘brawn’, but in functional terms ‘brain’ and ‘brawn’ together form a single, indivisible unit. ‘Brain’ cannot be regarded as something separate, but in effect is distributed throughout the entire enterprise, interwoven with its ‘brawn’. If we try to treat them as separate, we break down that ‘interwovenness’: the service ceases to be viable – and with it, perhaps, the overall enterprise.

By the nature of cell-anatomy, the machine-metaphor’s separation of ‘brain’ and ‘brawn’ does sort-of work at the lowest ‘cell-like’ levels of the enterprise, and perhaps gives a pleasing illusion of control at the highest levels. Yet the metaphor’s insistence on treating all ‘brawn’ as brainless – and ‘brain’ likewise as ‘brawnless’ – will guarantee intractable problems anywhere in the middle, and ultimately renders the entire enterprise unworkable. As we can see all too easily in practice – but perhaps not so easily see why…

Looking at a single cell in a microscope, it’s quite easy to identify the nucleus and the other individual sub-services. But it can be easy to forget that all of these sub-entities are enclosed within the one cell-wall, and operate as a single unit; and also easy to forget that each cell will usually be part of a larger structure, which itself will be part of a larger structure, and so on. And it’s also easy to become distracted...
by the fact that within each cell, there will be many subsidiary services, and often many instances of a particular type of subsidiary service, yet just one nucleus, one ‘brain’.

There are a few rare examples, such as the slime-moulds mentioned earlier, in which there may be multiple nuclei within the one overall cell – but we can safely leave them out of this discussion for now!

Given that apparent uniqueness, it’s easy to see how management – the business equivalent of the cell’s nucleus – would tend to think of itself as ‘special and different’. That delusion does even sort-of make sense at the highest levels of the enterprise, and at the very lowest, because in both cases there’s just the one nucleus on the chart – at the top, nothing below than brawn; at the bottom, nothing above but management. But in the middle, the delusion falls apart: up, down, sideways, there has to be both ‘brain’ and ‘brawn’. There, management is not and cannot be ‘special and different’: it’s just one more service amongst many, and nothing more than that.

Breakdown of ‘special and different’ delusion in the middle layers
Something else that’s easy to miss in the microscope – in fact very hard to see – is the process of communication and coordination that occurs both between the subsidiary services within the cell, and between the cell and other cells with which it’s associated. To make sense of viable systems, we need to understand not just the visible services, but the more subtle web of connections between them.

**Service-relationships**

At this point we can drop the cell-analogy, and go straight to business. What we’re mainly interested in here is the differences between *intra-service* relationships – those that occur within the service itself – and *inter-service* relationships that link between services, vertically, horizontally, and in other ways.

The main intra-service relationships are those between the ‘brain’ and ‘brawn’ – the subsidiary *delivery-services* that do the concrete work of the overall service unit. In classic business terms, there are requirements and definitions of ‘critical success factors’ (CSFs) going downward from ‘brain’ to brawn’; and performance-records – particularly the ‘key performance indicators’ (KPIs) – that go upward from ‘brawn’ to ‘brain’.

More precisely, they pass between the ‘brain’ of the service and the ‘brain’-services within the subsidiary ‘brawn’ services – a point we’ll come back to later.

The most visible inter-service relationships are those between the overall service and the ‘customer’ for the service – usually called the ‘requester’, in service-oriented architecture. The requirements for these relationships are typically documented in some form of ‘service-level agreement’ (SLA) or ‘operational level agreement’ (OLA), which should – but sometimes doesn’t – also define the metrics by which the quality of service would be monitored. So if we view a typical business-function as a service, we would expect to see a structure something like this.
A simple example of business function as encapsulated service

That’s probably all of the relationships that would be described in a typical IT-centric version of service-oriented architecture. In fact in most cases we’d be lucky to be presented with anything more than the basic ‘brawn to brawn’ service-to-service transactions, without any quality-of-service metrics at all – which is a serious problem in itself for IT service-management.

But in addition to those missing metrics, there are also other key inter-service relationships of which we need to keep track. Although there are others, as we’ll see later, the most obvious of these are the extensions of the same intra-service relationships – requirements, CSFs, KPIs and so on – up and down the service-hierarchy tree. Requirements cascade downward; report-metrics cascade upward, much as described in Balanced Scorecard and the like.
Hierarchy of services, each with their own performance-measures

Each ‘brawn’ within the service is really a complete service in its own right, so to some extent those ‘intra-service’ relationships and messages are actually inter-service relationships between the ‘brain’ of the ‘parent’-service and the ‘brain’ of the respective ‘child’-service. The ‘brain’ of the service is itself a service: in most cases its main ‘customers’ are other ‘brain’-services upstream and downstream in the service-hierarchy.

In addition to these, there are also a variety of ‘horizontal’ relationships that focus not just on individual transactions, but coordination of the full end-to-end processes in which the service takes part. And when we remember that in a real enterprise this will include human contexts as well as the machine-to-machine messages, we also need to include some other, often more subtle relationships and ‘transactions’ around auditing, monitoring, quality, and trust, that keep each service and the overall enterprise on track and ‘on purpose’.
When we look at all the requirements of an enterprise, viewed in terms of the living-organism metaphor, we can see that every sub-unit in the enterprise would need to have some means to do all of the following:

- to do its task – in other words, deliver its services
- to sense and report on its perception of its internal and external environment
- to remember, using some kind of repository of knowledge about its past
- to coordinate its activities with other systems and services
- to plan its activities in some way, coordinating strategies and tactics with others
- to adapt to and, where possible, improve its own environment and operation
- to maintain a sense of purpose, to contrast its present condition with a desired future condition

Every living system does all of this in some way: even an amoeba has a sense of purpose and direction, if only in terms of a simple reflex response to its chemical context. Every part of a functioning enterprise does all of this, too – because if it doesn’t, or if it doesn’t do it well, the enterprise won’t survive. Which means that if we want to build a true architecture of the service-oriented enterprise, this is the true scope of what we have to be able to describe, and to model.

But if this all seems a lot to include in a service-oriented architecture, don’t worry. The principles, and even the practice, have all been well-documented for decades, in the formal discipline of management cybernetics, ‘the science of effective organisation’. And for this we can turn to the fundamental tool in that field: Stafford Beer’s ‘Viable System Model’.
Services as viable systems

A *viable system* is any organised system that combines, and resolves to its benefit, all of the ‘survival demands’ in a changing environment. In business terms, that means an enterprise that not only survives change, but *thrives* on it; for a service-oriented architecture, that implies a structure that is not only capable of monitoring its environment, and any changes in that environment, but also *self*-adapting to change wherever practicable.

So the Viable System Model (VSM) describes the fundamental structure and organisation of any viable system. In essence, it’s a service-oriented architecture, in much the same sense as in the previous section.

We won’t need to go into the full detail of the VSM here – for that, see Stafford Beer’s book *The Brain of the Firm*, or Patrick Hoverstadt’s excellent *The Fractal Organization*. One of the model’s most important characteristics, though, is its recursive structure: the same basic pattern repeats at every level of the enterprise. As a result, it has extraordinary *scalability*: it has been used for every type and size of organisation, from small co-operatives in Britain, and mid-sized factories in Spain, all the way up to the management of the entire economy of a country – Allende’s Chile, way back in the early 1970s.

Beer’s ‘Cybersyn’ system for Allende not only created a true ‘balanced scorecard’ technique two decades before Balanced Scorecard, but showed how to use that information to make real-time decisions about the operations of a huge enterprise. (The system was destroyed by the military junta shortly after their assassination of Allende, not because it didn’t work, but apparently because it had been *too* successful in creating resilience against the covert ‘games’ of the American companies that had bankrolled their coup…) The Chilean website which describes that project – see Resources below – is a real eye-opener about what can be achieved when even very limited resources are combined with a few truly innovative ideas. Many of those ideas will still seem advanced even by today’s standards, several decades later. Go see for yourself: *very strongly* recommended for any enterprise architect or IT-architect.

The VSM was originally intended as a way to describe the actions and information-flows needed for management of the system – the metaphoric equivalent of a
nervous-system for the ‘viable enterprise’, but not much more than that. So whilst it’s a powerful model from an IT-architecture perspective, it does need some expansion to make it work well as a generic model for all types of ‘viable services’, rather than solely for ‘viable IT-systems’. We’ll explore that later: for now, the main point is that it starts from the same basic ‘brain/brawn’ split that we’ve seen earlier as a base to understand structure of services.

In the model, Beer represents these by a rectangle for ‘brain’ – the management-services – and a circle for ‘brawn’ – the delivery-services. Each ‘brain’ can manage any number of ‘brawn’ units – each of which in turn contains its own ‘brain’ and any number of subsidiary ‘brawns’. Crucially, Beer splits the ‘brain’ into three distinct subsidiary functions, and includes two other functions which are, in part at least, outside of the ‘brain/brawn’ pair.

Specialised sub-systems with each service in Viable System Model
All these functions or ‘systems’ are assigned numbers, as follows:

- **system-5**: maintain *policy, purpose and identity*
- **system-4**: *research* and report on the external environment, and develop *strategy*
- **system-3**: *plan* and *manage* the operations activities
- **system-3**: *monitor* and *verify* by sporadic audit of activities
- **system-2**: *regulate* and *coordinate* activities with other systems and services at a tactical level
- **system-1**: *do* the allotted task of the overall service, and sense and report on the internal environment

In the original model, what is now labelled ‘system-3*’ was first considered to be part of the system-3 management tasks; but practical experience, such as with the Chilean ‘Cybersyn’ project, showed that it did need to be handled separately. As we’ll see later, the VSM assignment for system-3*, as ‘sporadic audit’, is actually only one of a whole category of subsidiary ‘pervasive services’ with this relationship to the management-services.

A glance back at that list at the end of the previous section would show that the VSM ‘systems’ cover most of those requirements: the only items missing are the two about knowledge and memory, and adaptation and improvement. Although both of these are partly covered by the definition for system-4, they’re actually better handled by an expansion of the roles of system-2 and system-3*.

Although the VSM itself focuses on management (the system-3, -4, -5 cluster) and the information-flows needed for management, the real tension of importance in the enterprise is between purpose (the single top-level system-5) and the expression of that purpose in practice (the multitude of system-1 entities). *Everything else – including management – is just a support-service towards that end.*

Without those support-services, the purpose won’t happen: hence management does matter, and matter a lot. But we do need to keep a perspective here, and
perhaps a better perspective even than the VSM itself: we need to give equal attention to all of the services and service-categories, at every level and of every type, and avoid any over-emphasis on just one or two at the expense of all the others.

In effect, every service, whatever its level in the service-hierarchy, has the same internal structure and relationships; yet each has its own specialist role within the organism or, at the enterprise scale, the superorganism. Each service has its set of subsidiary services or ‘child-services’ with four possible categories of functions:

- task-delivery or delivery-support (system-1) – see Principles – Delivery services, p.44
- service-management (system-3, system-4, system-5) – see Principles – Management services, p.45
- ‘horizontal’ coordination with other services (system-2) – see Principles – Coordination services, p.62
- functions to keep the overall service on track and aligned with enterprise purpose (system 3*) – see Principles – Pervasive services, p.64

Classic management texts would bundle everything other than task-delivery into the service-management services. But as the VSM diagram shows, coordination and audit do need to be addressed separately – not least because they need to operate under a different type of management-structure, and operate in a completely different way to the vertical hierarchy of the main service-management.
Service-functions and the tetradian architecture-dimensions

The four types of services also align with the four architecture-dimensions of the tetradian model described in other books in this series:

- delivery services: physical dimension – actions and transactions
- management services: conceptual dimension – information, planning and reporting
- coordination services: relational dimension – connections across between people and across the silos
- pervasive services: aspirational dimension – enterprise purpose, identity and values

We also need to explore how the different types of services communicate with each other – see Principles – Properties and patterns, p.65. In most cases these would pass through predefined ‘normal channels’ – though note that many of these, particularly with people-based services, could use a range of means that may be much broader than just the usual IT-type transactions, records and messages. But in some cases, an urgent message or other exception – what Beer describes as an ‘algedonic’
communication - will need to jump from any service to any other: and these must be able to bypass those ‘normal channels’ completely – whether middle-management like it or not! So whilst the VSM does describe all the information-channels needed by a complete viable-system, it’s not always either the same or as simple as conventional IT-centric descriptions might expect.

For now, though, let’s look at those service-function types in more detail – starting with the main task-delivery functions.

Application

• What structures would you identify in your existing enterprise? In what ways are these structures layered, repeating the same kind of structures at different hierarchical levels?
• In what ways are these structures divided between ‘brain’ and ‘brawn’? Who or what sets the overalls purpose for each structure?
• In what ways do these structures communicate with each other? Via what mechanisms? And for what purposes?
• How are the activities of the various services coordinated? At each level? Between levels? And as nodes within end-to-end processes?
• How would you identify the efficiency, reliability and overall effectiveness of any service? Of communication and coordination within and between services? How would you improve any of this? What support – or lack of it – do you gain from the structure itself in doing so?

Resources

⁂ Balanced Scorecard: see Wikipedia summary at en.wikipedia.org/wiki/Balanced_scorecard

Viable System Model: see Wikipedia summary at en.wikipedia.org/wiki/Viable_System_Model

Cybersyn project, Chile, 1971-73: see www.cybersyn.cl/ingles/cybersyn/index.html


The ‘tetradian’ as architecture-dimensions: see chapter ‘Dimensions of Architecture’ in Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise* (Tetradian, 2008)
PRINCIPLES – DELIVERY SERVICES

Summary
The ‘delivery’ services are what are most likely to be understood as the services of the enterprise – the visible work done by the organisation’s workforce and systems. In addition to those which deliver services or products to external customers, there are many which provide internal support in a multitudinous variety of forms. Some of these latter services may be outsourced – especially those ‘non-core’ services which provide only a kind of background or infrastructural support.

Details
Description
[[see published book for further details]]

Sub-units or variants
[[see published book for further details]]

Interfaces
[[see published book for further details]]

Application
[[see published book for further details]]
PRINCIPLES – MANAGEMENT SERVICES

Summary

The management services focus on information-flows and on the future of the enterprise. As such, they represent the ‘brain of the firm’ – though for that reason may perhaps gain an excess amount of attention compared to the needs or natures of other service-types. Their interfaces tend naturally towards hierarchies and silo-structures, creating a risk of the ‘Taylorist trap’ of over-escalated issues, and sometimes serious difficulties in coordination of end-to-end processes. These can be resolved by treating management not as ‘special and different’, but as just another set of services within the overall enterprise.

Details

Description

In the service-oriented enterprise, management is about what the enterprise knows, how it thinks, and how it decides. Although the details will differ with each enterprise, there is actually very little difference in overall function or content or even in structure – hence one reason why managers may move from one enterprise to the next with relative ease, and relative success.

There’s a subtle trap here that has killed many a company, and is very difficult to spot if you’ve been trained only in the Taylorist tradition of ‘scientific management’. This is that whilst the management skills themselves may be portable, and the information-content for management is much the same everywhere, that which is managed is not the same – it’s different with every context.
Taylorism assumes that the ‘brain’ always knows best; and the higher up you are in the hierarchy, the more knowledge you’re presumed to have. Hence the ‘higher-ups’ are assigned more authority, and more decision-making responsibility. So whenever there’s a problem, an exception to the predefined work-instructions in The Book, it’s ‘escalated’ upward in hierarchy for decision; and ‘actions’ passed back down again from on high as to what to do to resolve the problem.

But the catch is that this only works when the problem is one of conceptual knowledge. Often the real need is not ‘thinking with the head’ – the theory – but what we might call ‘thinking with the hands’ – which can only be gained through skills, through what may well be years of hard-won hands-on practical experience. Yet the Taylorist principles give preference and precedence to people who are strong on abstractions and strong on theory – but give little or no credit for competence in the practice. Which means the ‘decision-makers’ are less and less likely to be competent to make practical decisions – a situation that gets worse with each step higher up the tree.

As Deming demonstrated, the proper place for practical decisions is at the point of practice: sometimes we need to ‘escalate’ down in the hierarchy, not up. But in Taylorism, the workers at each level are deemed to be ‘brainless brawn’, and have no power to make decisions. In effect, they have the nominal responsibility, but no authority. Instead, the authority is ‘owned’ by the manager: and if the managers insist on holding on to responsibilities for detail-concerns for which they don’t have the practical competence, everyone is stuck.

The only way to get ‘unstuck’ is to recognise that ‘higher’ knowledge is not necessarily ‘better’ knowledge: what matters is whether it’s the appropriate knowledge. And the only way we can find out what type of knowledge is appropriate in any given context is by creating a respectful partnership of peers. Management is just a service, just like any other in the enterprise: theory serves practice just as much as practice serves theory.

I saw this at work time and again in my years as a systems-analyst in aircraft research. Each year a new generation of bright young graduates would appear on the site, waving their shiny new engineering degrees, certain that they knew everything about aircraft design. And each year, with the full agreement of the foreman, the machinists and fitters would smile, and go into a subtle work-to-rule: they would do exactly what the new engineers asked – which meant that nothing worked. Components were made for assemblies that couldn’t be assembled; materials mated that wouldn’t last more than a matter of minutes in the field; all precisely as per the design-specification… Some of the young engineers took longer than others, perhaps, but they all learnt soon enough that there was another knowledge they needed here: a knowledge of how to make things work in the real
world, rather than only the comfortable world of theory. If they wanted their designs to succeed in practice, they needed to respect that other knowledge, and work with it, no matter how different it was from their own.

Would have been good if some of the managers I’ve met in my time had learnt that lesson too. Oh well…

Management-services form natural hierarchies, with each level representing a different layer of abstraction. Enterprise-architects would be familiar with some of these, such as from the classic Zachman set, with its five, or six, or seven distinct layers, from unchanging ‘universals’ at the top to the real-time maelstrom of operations at the bottom. Real enterprises will provide many variations on that theme, such as further splits on boundaries of geography or market or function, but the base-principle of a layered hierarchy of abstraction will remain the same.

Echoing the same layering, the Viable System Model embeds three distinct – and functionally different – layers within the ‘brain’ of each system-1 delivery-service:

- **policy** – purpose, identity, imperatives, vision and values
- **strategy** – ‘outside / future’
- **direction** – ‘inside / now’

The different functions of these subsystems also align to different abstractions of time, from far future all the way through to near past. Note, though, that these apply in their own ways within every system and service – not just at the top of the enterprise management hierarchy.

**Viable system context**

In viable-system terms, these are the *VSM system-3*, *system-4* and *system-5* services.
In VSM diagrams, these are represented by a rectangle for the management-services collective set, containing three squares or rectangles for the three sub-systems, with system-5 uppermost.

In the tetradian architecture-model, these represent the *conceptual* dimension; system-5 also represents the management expression of the *aspirational* dimension.

In the organism-metaphor, these represent the brain and nervous-system of the enterprise – action-requests going out to the whole enterprise, and sense-information returning back, with decision-making taking place both centrally and distributed throughout the enterprise.

**Enterprise context**

At the most abstract level of the enterprise – as shown by the bounding rectangle around the system-3, -4 and -5 entities in the top-level VSM diagram – there will be exactly one overall management service.

At board-level, the system-5 (‘policy’) should be represented by the Chief Executive Officer (CEO) or Managing Director (MD). The system-4 (‘strategy’) would typically be represented by the whole board, with CEO or MD as chair; or perhaps ideally by an explicit role of Chief Strategy Officer (CSO). The system-3 (‘direction’) would typically be represented by the MD or COO.

In the body of the enterprise, the management-services would be represented by distinct groups or individuals with the respective assigned roles. For example, a formal strategy-group may take on responsibility for the system-4 ‘strategy’ services directly below board-level; various staff-members will take on the system-3 ‘direction’ role – often with a job-title of ‘manager’ – at most levels within the enterprise. Further down in the hierarchy-tree, the respective tasks may well form
only a part of the responsibilities of a work-role – especially for the system-4 ‘strategy’ and system-5 ‘policy’ tasks.

Sub-units or variants

In the VSM, as described above, there are three distinct services that combine to make up the overall management-service within each system-1 unit: system-5 ‘policy’, system-4 ‘strategy’ and system-3 ‘direction’.

Whilst working on this chapter, I came across a point about the VSM that I hadn’t noted before, and which may be important for anyone applying systems-theory principles in enterprise-architecture.

I base much of my own enterprise-architecture work on a rethink of Tuckman’s well-known five-phase ‘Group Dynamics’ project-lifecycle, as an overview-model of the overall workings of an enterprise:

- **forming**: purpose, identity, strategy; also far-future
- **storming**: people-issues; anywhere from far-future to far-past
- **norming**: plans and schedules; also near-future
- **performing**: production; also ’now’!
- **adjourning (or mourning)**: completions; also near- to mid-past

Yet when we look at the ‘management’ section of the VSM, it seems it covers only three of the five Group Dynamics phases: VSM system-5 ‘policy’ aligns to ‘forming’; system-4 ‘strategy’ aligns to the later part of ‘forming’, plus most of ‘norming’; and system-3 ‘direction’ aligns to the later part of ‘norming’, plus most of ‘performing’.

But there’s no explicit VSM coverage at all of the ‘storming’ phase, the people-issues – which seems strange considering Beer’s own strong personal bent towards ‘left-ish’ participatory politics. And although the original VSM system-3*, ‘random-audit’, does sort-of touch the ‘adjourning’ phase, it only does so on an occasional basis – not the continuous processes needed for completions and lessons-learned.

This gap may stem from the VSM’s history as a model of information-flows for management and the like; but it still seems a huge hole in the coverage of what’s actually needed for systemic design of management processes. Odd…
We do cover these ‘missing’ issues here, via our extended system-2 as ‘coordination services’ and system-3* as ‘pervasive services’. But if we use only the original VSM to describe management-services, we’ll need some alternate means to cover those missing issues – or we’ll once again end up with intractable issues that we can’t see.

Management-services at any level should always include exactly one of each of the following subsidiary services.

**Policy, purpose, identity**

The VSM system-5 ‘policy’ service is responsible for ‘holding the vision’ for the enterprise: anything to do with principles, policy, values, identity and so on. Ultimately, it’s also responsible for compliance to regulation and legislation, and defining how such should be interpreted in the enterprise – and that includes the more subtle ‘regulations’ of the social milieu or ‘corporate culture’ in which the enterprise operates.

At the topmost level, the ‘policy’ service sets the anchor – or perhaps ‘guiding star’ – for the entire enterprise:

**Purpose defines the enterprise priorities**

Within each layer – each ‘system-1’ instance – it’s the role of the local ‘policy’ service to interpret those priorities, and to assess and evaluate all activities in terms of those priorities. The function here is the enterprise’s means to handle uncertainty: whenever we encounter a real-world context in which the usual rules don’t apply, and analysis is either too misleading or too slow, the principles and purpose define the priorities to guide those decisions we need to make. Purpose *matters*.

But here we hit up against a nasty problem: there’s almost always a difference between the principles and values that the enterprise would espouse in public, and the often unconscious ones used in its actual decision-making. And sometimes that
difference can be huge – so huge that no-one in the enterprise can see it, or dare see it. Under those circumstances, what we’ll need most is some perhaps-painful honesty, for which one useful guide is Stafford Beer’s oft-quoted acronym POSIWID:

“The purpose of a system is what it does”

If we don’t like what the system does, or if what it’s doing doesn’t align to the espoused values, we need to look closer at how we’re managing purpose in the enterprise. The actual ‘purpose’ of the system is what it does: not just what we hope or believe or would like to pretend it does. If we say that safety is a high priority, yet we have a high accident-rate, what we’ve built is a system in which safety is actually a low priority: the problem is systemic, right at the core of the enterprise – and that misalignment is simply reflected in the work-practices, not caused by them.

To change that misalignment on safety – to use the same example – we would need to review everything in terms of safety: not just processes, but performance-metrics, work-incentives, knowledge-sharing and the rest. So the metrics we need to run the business are defined by the business-purpose: the priorities define the content that we need in the enterprise ‘Balanced Scorecard’.

| Those metrics pass up and down the tree via the system-3 ‘direction’ service – but to make sure that we can actually trust the figures and values reported in those metrics, and that they really do mean what they claim to mean, we need to verify as well, by some kind of audit. We also need to support the root-level practices, and any changes needed in those practices, that end up being reported in those metrics. More on both of those concerns when we look later at the ‘pervasive services’ – see Principles – Pervasive services, p.64. |

In the same way, purpose also acts as the guide and final arbiter for decision-making in strategy (VSM system-4), in direction (VSM system-3), and in the balance between them. As we move further down the tree, this ‘purpose’ also includes constraints imposed by any applicable legislation and the like; and also any standards either chosen by the enterprise, or mandated or recommended by the
market and milieu in which the enterprise operates – such as ISO-9000 certification for quality-systems, or an industry-specific XML-based messaging format. Purpose defines the standards – in every sense – by which the enterprise will measure itself.

**Strategy (‘outside / future’)**

The VSM ‘system-4’ service is responsible for assessing and interpreting the milieu and market in which the enterprise operates, to enable the enterprise to respond **proactively** to its environment. Typical themes include:

- scanning the business environment for context and for signs of future change
- strategic foresight and forward planning, through tools such as scenario development and causal layered analysis
- assessment of strategic risk
- technical development, service development, product development and capability development

Strategy is another whole field in itself – far more than we could cover here, given that the emphasis in this book is more on enterprise structure and architecture – so for now it’s probably best just to leave it at that summary above. But there are plenty of good texts that do address those issues from an architecture-like perspective: see the Resources section below for some examples.

Another key role of the ‘strategy’ service is its engagement in ‘strategic conversations’ between past and future for the enterprise: for example, sales (past) versus marketing (future). Note, though, that this kind of ‘conversation’ needs to occur at **every** level of the enterprise, as appropriate to the respective context – and not solely in a separate ‘strategy unit’ that reports only to senior management.
Direction (‘inside / now’)

The VSM ‘system-3’ service is responsible for coordination of all the activities in this service’s ‘downstream’ delivery-services – its collection of ‘system-1’ instances. In short, this is the classic role of middle-management: a myriad of day-to-day decision and detail. This is the most visible aspect of the overall management services, and usually accounts for the bulk of their work, too.

In poorly-designed organisations, ‘direction’ is often the only part of the management-services that exists lower down in the hierarchy-tree – which literally leaves the organisation with only half a brain at those levels, capable only of dealing with the present and nothing else. Not a good idea…

In addition to directions to the ‘child’ delivery-services, the two main tasks here are resource management, going downward, and performance management, coming back up the tree. Balancing the conflicts within and between these two tasks is rarely easy.

Resource-management sounds simple enough until we remember that those resources have to come from somewhere first: and then it’s suddenly not so simple… As we’ll see later in the discussion on coordination-services, there’s another Taylorist trap that hits overly-hierarchical organisations, because many of the resources have to come from other silos: and if we don’t have good horizontal ‘silo-busting’ links, we’ll be stuck with an unwieldy bureaucratic mess. Which is no-one’s idea of fun – and far from effective, too.

The other catch is that to know what resources we need for the ‘child’ services, and how to divide the resources between them, we not only have to be clear about what we’re asking them to deliver, but also how we measure what they’ve delivered – in other words, the performance-metrics. If we get the design of the performance-metrics wrong, or if they don’t actually mean what they claim to mean, we’ll again be stuck.
It gets even more complicated when – as is usually the case – there are delays in the system, because the performance measures will always be somewhat out of sync with the timing of the resource-allocation. By the time these constraints echo across even a simple supply-chain – such as the four-partner chain in the classic Beer Game simulation – it takes a great deal of skill, experience and mutual trust to avoid an inefficient, ineffective, wasteful, frustrating, blame-filled mess. There are a fair number of online implementations of the Beer Game, so it’s well worth having ago at it yourself: it’s not easy.

The Game was originally developed by Peter Senge and his colleagues at Sloan School of Management, from the 1960s onwards, as is explained in more detail in Senge’s book on systems-theory, *The Fifth Discipline*. Both strongly recommended, anyway.

Yet another complication occurs with aggregation. In this section of the management-services, part of our task is to take all of the performance-metrics from the ‘child’-services, and aggregate them into a single set of metrics to pass upward in the tree. The catch here is that we need to ensure that the same definitions and transforms are used in each of the ‘children’, otherwise we can no longer compare like with like, and the metrics risk being meaningless. The same applies further up the hierarchy-tree, particularly when we have to compare metrics from different silos. Once again, there’s plenty of room for confusion here…

I came across a spectacular example of this in my work with a government department, when ministers and opposition almost came to blows over conflicting performance-figures that were supposedly derived from the same reporting-systems. They weren’t, but the fact that all the files and data-fields had exactly the same names did kind of add to the confusion!

I’ve described this incident in other books in this series, so I won’t expand on it here: suffice it to say that this is where abstract-sounding notions such as ‘database of record’ and ‘single source of truth’ start to make real sense to the business-folks… More detail on this later when we look at ‘business systems’ – see the section ‘Services and business-systems’ in *Practice – The knowledge of services*, p.87.

And there is one more potential Taylorist trap of which we need to be aware: the tendency of managers to subsume all control of the coordination-services, and even
the pervasive-services, into their own role. In part this is because classic ‘scientific management’ barely even acknowledges the need for such services; in part this is because rigid silo-structures often lead to a ‘need’ to control and limit any ‘horizontal’ connections, even though they’re essential for any cross-silo end-to-end processes; and in part it’s because, if there’s no explicit support or resourcing for such services, the ‘direction’ management-service is probably the only place they can go, whether they fit there or not. But placing them there does cause all manner of intractable problems: more on those later in the respective sections, Principles – Coordination services, p.62, and Principles – Pervasive services, p.64.

Overall interaction

The overall management-service is delivered by the combined actions and interactions of the embedded system-3, system-4 and system-5.

Management services: internal interfaces

System-3 ‘direction’ and system-4 ‘strategy’ need to engage in continual ‘strategic conversation’ on a variety of themes such as:
- present sales ↔ future marketing
- present staff ↔ future needs
- present funding / resources ↔ future needs
- present products and production ↔ future needs

System-5 ‘policy’ acts as the arbiter in this conversation, and also guides decision-making within each of the other two services. There would be some feedback into ‘policy’ from both, but the main drivers to policy-change will tend to occur more from higher up in the hierarchy-tree.

**Interfaces**

Management-services have symmetrical *vertical* connections, as both *provider* and *requester*, with the matching service-type above and below in the hierarchy-tree: system-5 ‘policy’ connects to the ‘policy’ management-services in the ‘parent’ above and within the ‘child’ system-1 delivery-services below; and likewise for system-4 ‘strategy’ and system-3 ‘direction’.

At the top of the tree – in other words the executive-management level – the ‘upward’ connections link to the operating milieu and business-environment. Policy comes ‘down’ in the form of legislation, regulation, standards and social expectations; reporting goes ‘up’, such as in the classic example of the company Annual Report.
Management services: interfaces with other systems

Descending downward, the hierarchy-tree terminates wherever management finally connects with actual service-delivery. In those cases, ‘policy’, ‘strategy’ and ‘direction’ do not connect with a matching counterpart, but with the service-delivery itself.

This is also the effective source of the key distinctions between ‘staff-management’, ‘middle-management’ and ‘line-management’.

‘Staff’ are at the top of the hierarchy-tree: they connect to other management-services below, but must deal with the real-world business-milieu above.

Middle-managers are literally in the middle: they connect primarily with other managers both above and below.
But although line-managers report to other managers upward, they connect with service-delivery below – the horizontal links through which the ‘real-world’ work actually happens. So there’s a subtle danger here that middle-managers need never see anything other than the abstract, vertically-oriented world of management-services – an unfortunate trait whose disastrous consequences can be seen all too clearly in many organisations...

Because all of the primary connections are vertical, management-services have a tendency to form into hierarchies and silos. This is a natural and necessary consequence of these interfaces: the only key point to bear in mind is that the hierarchies are not the whole of the story.

The management-services also have secondary horizontal interfaces with coordination-services and pervasive-services. These interfaces are described in more detail in the respective sections – see Principles – Coordination services, p.62 and Principles – Pervasive services, p.64.

Application

• What management services exist in your enterprise? How would you distinguish between the different types of services at different layers of the enterprise?

• What happens to your perception of management in general when you choose to view them as services, rather than as mechanisms of control? What happens when you view each not as ‘a thing apart’, something inherently ‘special and different’ – as in the Taylorist machine-metaphor – but as just another type of service delivered to the enterprise?

• By what means are the system-5 services – purpose, policy and identity – defined in your enterprise? In what ways – if at all – is that process of definition distributed throughout the enterprise?

• In what ways and by what means are the expression of policy and identity distributed through the enterprise? Would you be able to use the term
‘celebrated’ to describe that policy and purpose, or does it seem more that it’s something *imposed* on the rest of the enterprise by senior management? If the latter, what impact does this have on the enterprise as a whole? What, if necessary, could or should you do to change this?

- By what means are the system-4 services – business-intelligence, strategy, market-knowledge and the like – defined and determined in your enterprise? In what ways – if at all – are those processes distributed throughout the enterprise, or are those responsibilities and capabilities reserved to a specific team somewhere near the senior-management levels?

- In what ways and by what means are the *expression* of strategy and the like distributed through the enterprise? Is it co-developed, ‘owned’ throughout the enterprise, or does it seem more that it’s something *imposed* on the rest of the enterprise by senior management? If the latter, what impact does this have on the enterprise as a whole? What, if necessary, could or should you do to change this?

- What detail-level tactics are used for direction and management of the ‘downstream’ delivery-services? What requirements and criteria are passed down the tree? In what forms? How are the requisite resources identified, obtained and distributed? What performance-metrics are passed back up from the downstream services? In what forms? How is the balance between requirements, resources and performance identified, verified and maintained? And in what ways, and by what transforms, are those performance-metrics aggregated to pass up the tree?

- How are the competing demands and priorities of policy, strategy and tactics resolved within the management-services? How does this vary at different levels and in different areas of the enterprise?
• What differences can you see between the espoused values and priorities of the enterprise, and the priorities which actually apply in practice? If “the purpose of the system is what it does”, what is the actual ‘purpose’ of the enterprise? In what ways does this vary at different levels and in different silos of the enterprise? What could you do, in architecture terms, to bring the espoused purpose and actual purpose into closer alignment?

• If management-services tend naturally to form hierarchies and silos, because of the nature of their interfaces and interdependencies, what advantages and disadvantages to the enterprise as a whole can you see arising from this fact? What could you do to address the disadvantages without impacting the advantages?

Resources

- Tuckman ‘Group Dynamics’ as project lifecycle and as organisation-model: see chapter ‘An overview’ (and overall book-structure) in Tom Graves, *Real Enterprise Architecture: beyond IT to the whole enterprise* (Tetradian, 2008)
Environmental scanning for business context: see Wikipedia summary at en.wikipedia.org/wiki/Environmental_scanning

Causal Layered Analysis: see www.metafuture.org/Articles/CausalLayeredAnalysis.htm


Scenarios and ‘strategic conversation’: see Kees van der Heijden, Scenarios: the art of strategic conversation (Wiley, 2004)

Online implementation of the Beer Game: see www.masystem.com/beergame


Online resources on ‘Fifth Discipline’: see www.fieldbook.com
PRINCIPLES – COORDINATION SERVICES

Summary
Where management-services build vertical hierarchies and silos within the enterprise, the ‘coordination’ services bridge across the silos, creating links for end-to-end processes, and enabling cross-functional integration. The services typically provide coordination in three distinct yet related areas: the routine running of business processes; changes to the business structure and capabilities; and changes to the overall purpose and direction of the business. The enterprise cannot operate without such services, though few will be formally recognised in management structures – an issue which itself can have serious impacts on enterprise effectiveness.

Details

Description
[[see published book for further details]]

Sub-units or variants
[[see published book for further details]]

Develop the Business
[[see published book for further details]]
Change the Business  
[[see published book for further details]]

Run the Business  
[[see published book for further details]]

Interfaces  
[[see published book for further details]]

Application  
[[see published book for further details]]
Summary

The pervasive services extend the enterprise imperatives from the topmost ‘policy’ management-service through to every part of the entire enterprise. One specific example, as described in the VSM, is sporadic audit, to verify reported results against actual activity. In practice, though, there would usually need to be a much broader range of equivalent pervasive-services, to ‘audit’ against other enterprise principles, policies and values – including themes such as safety, security, quality and corporate social-responsibility – and create awareness of the issues, and the capabilities needed to support those themes throughout the enterprise.

Details

Description
[[see published book for further details]]

Sub-units or variants
[[see published book for further details]]

Interfaces
[[see published book for further details]]

Application
[[see published book for further details]]
Summary

No service exists in isolation. It exists to serve; it has its own specific nature and properties, each has links and dependencies with other services. A ‘systems’ perspective helps in making sense of the complexity of all these concerns, by describing patterns of inter-relationships which are known to work in practice. These relationships also clarify options for service re-purpose and re-use, through a concept of ‘completeness’ as a guide for service design and implementation.

Details

Services as systems
[[see published book for further details]]

Service quality
[[see published book for further details]]

Service interdependence
[[see published book for further details]]

Service completeness
[[see published book for further details]]
Application

[[see published book for further details]]
PRACTICE – AN OVERVIEW

The service-oriented enterprise is composed of a myriad of other services at many different scales. We can describe each of these as belonging to one of four categories: delivery, management, coordination and pervasive quality:

![Four categories of services](image)

But in itself this doesn’t tell us much about how the services interrelate with each other in practice; and it also doesn’t tell us much about how to describe that real-world practice. For those, our best tactic would be to merge that four-fold service-structure with another five-phase approach to modelling and mapping the architecture of the enterprise:
This methodology is described in more depth in another companion volume in this series, Real Enterprise Architecture: beyond IT to the whole enterprise – see the Resources section below.

So this Practice section is split into five chapters, each describing aspects of service-architecture and the service-oriented enterprise in terms of the respective phase:

- **purpose**: an emphasis on pervasive-services: “what purpose does the enterprise serve? whom does it serve?” – see Practice – Service purpose, p.70
- **people**: an emphasis on management-services: “how do services relate with each other?” – see Practice – Services and functions, p.72
- **preparation**: an emphasis on coordination-services: “what do services know and need to know?” – see Practice – The knowledge of services, p.87
• **process**: an emphasis on delivery-services: “what does each service do? how does it do it? how do we coordinate services to get real work done?” – see *Practice – Services in action*, p.89

• **performance**: an emphasis again on pervasive-services: “what have we done? how can we improve what we do?” – see *Practice – Optimising services*, p.91

Although this cannot cover every possible aspect of service architecture, of course, it should provide enough to begin to put the principles into real practice within the architecture of your own present enterprise.

**Resources**

- Recursive five-phase enterprise-architecture: see chapter ‘An overview’ and overall book-structure in *Real Enterprise Architecture: beyond IT to the whole enterprise* (Tetradian, 2008)
PRACTICE – SERVICE PURPOSE

Summary

The first question for any service-architecture is “whom do we serve?” The answers identify the overall purpose for the services provided by the enterprise-as-service, and the anchors for the architectures that support that need.

Details

Business as a service

[[see published book for further details]]

Application themes: enterprise as service

[[see published book for further details]]

Application themes: stakeholders

[[see published book for further details]]

Application themes: stakeholder imbalance

[[see published book for further details]]

Application themes: vision and values

[[see published book for further details]]
Application themes: policies and constraints
[[see published book for further details]]

The structure of purpose
[[see published book for further details]]

Application themes: vision, role, mission, goal
[[see published book for further details]]

Application themes: mission and service
[[see published book for further details]]

Application themes: mission, goal and metrics
[[see published book for further details]]

Application themes: staying on purpose
[[see published book for further details]]
Summary

With purpose identified, the next architecture requirement is the definition of the services needed to implement it. In doing so, we need to map how these services would interrelate with each other – particularly the people-aspects of the enterprise. One proven tactic is to create a Functional Business Model as a base-map for subsequent explorations.

Details

Function as service, service as function

In mathematical terms, a function is something that can create a change in accordance with intent. The same is true in business too: and especially at the upper levels of the enterprise, ‘function’ is almost synonymous with ‘service’. Business-functions are services.

So how do we describe how these functions or services interrelate with each other? The layering of delivery-services – the business functions – is also a layering of the management-services and their resultant silos; and the various ways in which those delivery-services link together across the silos also describes the layering and interactions of the ‘run the business’ coordination-services. In that sense, functions describe the enterprise itself.

So a Functional Business Model is one of the most valuable tools for architecture of the service-oriented enterprise. Once we have the vision and values clearly
described, as the anchors for everything else, the next item we’ll need is that function-model.

**The Functional Business Model**

A Functional Business Model is a layered list of business functions – a visual summary of what the enterprise does in functional or service terms. So the aim here is to create a model that remains much the same as long as the enterprise does that kind of work.

For that reason, the content of the model needs to be independent of the current structure of the enterprise. So in a sense we run the usual enterprise-architecture process backwards here: we start with the ‘to-be’ model, the description of the idealised enterprise, and then work backwards from there to link it to what actually exists in the present day. That in turn will give us the gap-analysis that we’ll need for future change towards that ‘to-be’ ideal.

The aim is that the model should summarise the whole enterprise on a single page. Although it would fit best on a larger sheet (A3 or 11x17), it should still be readable on a standard single sheet (A4 or 8½x11). In this form, and at this size, it supports understanding across the enterprise, at a ‘big picture’ level. In effect, it provides a high-level visual anchor for the enterprise’s business-architecture.

---

Remember that an ‘enterprise’ here can be at any scale: any subset or superset of an organisation. But when the model works well, it provides an instant overview of the whole enterprise, at whatever scale that may be.

In one case, we saw a copy of the model pinned up on the wall in almost every manager’s office. It guides conversations about end-to-end processes, or any process that crosses boundaries between organisational silos – a process being in effect a choreographed path that links business services together to do something that contributes towards the organisation’s goals.
With a completed function-model, you can allocate functions to *business systems* – clusters of activities that do similar things and share similar data in different areas of the enterprise. This simplifies mapping of applications and systems across the whole of the enterprise. You can also map projects and applications to functions, allowing you to see gaps and overlaps in the system coverage, and identify potential for new projects. And by cross-mapping to activity-based costing, you can map costs to functions, highlighting the targeting of project spend. That’s a lot of value from one relatively simple model.

The model is laid out in whatever format best describes the business. It could be laid out as a left-to-right flow from ‘Customer’ to ‘Delivery’, with supporting activities above and below, as in the first of the examples shown later in this chapter, on p.82; it could be laid out as an open box; or any other layout that feels right and makes sense as a business description. But whatever layout is used, it’s typically organised as a visual nested hierarchy, usually with three or four tiers of functions:

- **Tier 1**: major categories of business functions – key aspects of what the enterprise as a whole actually *does*.
- **Tier 2**: clusters of related activities – the major support-missions for the tier-1 functions
- **Tier 3**: ‘activities’ or clusters of related tasks – typically the emphasis of a team’s or a person’s work.
- **Tier 4**: the individual tasks within business processes – the actual delivery-processes

The tier-4 functions are usually not shown on the diagram, for lack of space, but instead would be listed within a supporting text document. Although there is real value to be gained from gathering the full information at every tier, the cost of doing so rises with the level of detail. The first three tiers are probably essential for
any enterprise-modelling exercise, but it’s likely that tier-4 information may be worth gathering in depth only when an appropriate project touches the respective area of the enterprise.

**Creating a Function Model**

A few key points need to be kept in mind throughout the process of creating a function-model:

- the model needs to be independent of the current organisational structure or business units – for example, an org-chart is effect an *overlay* on the Function Model
- the model needs to be independent of any existing applications – because they are likely to change in future
- each function or activity in effect delivers a service

To identify candidate functions and activities, trawl through existing documents such as org-charts, strategies, plans and annual reports, and also intranet sites and the like. Look for:

- org-chart entries: each role implies one or more business functions – though they may overlap, or be repeated in multiple locations, or aggregate several distinct functions
- references to projects: each is likely to imply a new or upgraded capability, which again implies a function
- references to phone-lines or other contact-points: these imply business-functions behind the points of contact

It’s also a good idea to build a network of business acquaintances to help you fill in any gaps. One of my standard tactics is to sit in whatever café is used by staff to get their morning coffee, with a cluster of interesting-looking diagrams casually laid out on the table as if working on them, and use
Another source of pointers to functions is the business data. If an enterprise data-model is available, trawl through that, looking for the implied functions that would create, read, aggregate, update or delete the information-items.

Every business function does something, so each function-label on the model should begin with or include a verb. For the tier-1 and tier-2 functions it would be acceptable to use abstract verbs such as ‘Provide’ or ‘Manage’, but wherever practicable, and for tier-3 or tier-4 especially, it’s advisable to use more specific, descriptive verbs such as ‘Receive’, ‘Assess’, ‘Monitor’ and so on.

**Application themes: tier-1 functions**

- What are the major categories of business functions?
- How do these functions relate with each other, in terms of service-categories and service-layering?

![Diagram of tier-1 functions]

**Functions: tier-1 example**

Aim to define some six to twelve tier-1 functions. These will usually be obvious in the structure of the enterprise: for example, every organisation will have a set of
‘business support’ functions such as HR and finance, a set of contact-points for customers and other stakeholders, a layer for strategy and management, and a set of core business-processes.

One common split is on divisional boundaries, but beware of anchoring the model too tightly to the current structure of the enterprise, because it could well change next week. Instead, aim to make your categories reflect key aspects of what the business actually does.

Some industries already have their own generic function-models, such as eTOM for telcos, and SCOR for supply-chain and logistics. They’ll need adaptation to the specific context of the enterprise, even within the respective industry, but they’re useful as guidelines in any case.

**Application themes: tier-2 functions**

For each tier-1 category:

- What are the main clusters of related activities that occur within this category?
- How do these functions relate with each other, in terms of service-categories and service-layering?

Expect to identify around 40 to 50 of these in total. They can sometimes be found from job-titles: a truck driver, for example, or a warehouse manager, who each do a range of related business activities and tasks. The org-chart will also give some pointers on this, though again take care not to tie the list too tightly to anything that’s likely to change.
Functions at tier-2 can be harder to identify than those at tier-3 and below: the latter can be detected readily via a trawl through documents and the like. To derive tier-2 functions, look for natural clusterings of tier-3 functions: often these will be implied by higher-level entries in the org-chart, or by groupings of functions that reappear together in different geographical locations.

Application themes: tier-3 functions

For each tier-2 cluster of related activities:

- What are the main ‘activities’ or clusters of related tasks?
- How do these functions relate with each other, in terms of service-categories and service-layering?
Tier-3 ‘activities’ typically represent the main emphasis of a single person’s work. Aim to identify 200 to 300 of these – perhaps less in a smaller organisation, but not more, in case the model becomes too complicated to make sense.

Functions: tier-3 example

A warning: human nature being what it is, every manager will want you to list every one of their business functions on the diagram… You’ll need to apply some fairly strong negotiating skills to get the balance right across the whole organisation, because there’ll never be enough space to list everything! As a guide to negotiation, set yourself a strict limit of 250 or so tier-3 functions for the whole enterprise.

All of the functions they describe will be valid, of course, but most will be tier-4 ‘tasks’ which can be listed in the supporting document. What you’re after here is the small handful of – to them – higher-level tier-3 functions that summarise their work in relation to everyone else: probably no more than half a dozen functions each, in practice.
Application themes: tier-4 functions

For each tier-3 activity:

- What are the tasks within the activity?
- How do these functions relate with each other, in terms of service-categories and service-layering?

Unlike the other tiers, which are usually rather more abstract collations or aggregations of business-functions, tier-4 ‘tasks’ will either be, or be close to, actual tangible work. Since they don’t have to be squeezed onto the diagram, there can be any number of these: document them in any way that seems appropriate to the business need.

Verifying a Function Model

As the model is developed, it needs to be verified for completeness. The simplest way to do this is to think of each function as a service, with each component intimately interlinked with and dependent on all of the other services within the enterprise, and (certainly at tier-3 and above) containing other subsidiary services which it coordinates to deliver its overall ‘product’ – whatever that may be. In effect, the whole enterprise delivers one or more services to its clients, and in turn ‘consumes’ services from other entities such as suppliers, partners, government and other stakeholders.

To verify ‘completeness’ in a viable-services sense, the essential test is that each of the Viable System Model ‘systems’ is present in some form or other – either ‘contained’ within the respective service, or provided via a link from some other
service. In addition to routine audit in the specific terms for that service, it’s also advisable to verify that quality-management of some kind is applied consistently throughout the enterprise, as a mandatory pervasive-service for every context. The checklist that follows addresses all of these concerns, but in practice is only the minimum check that should be applied: add further context-specific tests to the list as appropriate.

**Application themes: verifying a Function Model**

Apply the following checklist to each function, and at each tier:

- **policy**: what is the service’s purpose? who or what defines policy?
- **strategy**: what is the current strategy? outside relationships? who defines this?
- **manage**: how are the service’s tasks defined, managed and monitored?
- **verify**: what random checks or audits are used to verify performance?
- **coordinate**: how is the service coordinated with other services?
- **tasks**: what does the service do? how does it do it? how does it support its ‘downline’ services (if any)?
- **exceptions**: how does the service identify and resolve any run-time exceptions?
- **quality**: what corrective-action does the service undertake for causes of issues?
- **track**: how does the service track and manage quality-issues and other issues?
- **improve**: how does the service manage improvement of its processes?

Each of these checklist-items should reference services that this function ‘consumes’, either from other functions, or from subsidiary functions and tasks within itself. In principle, each of these services must exist somewhere, otherwise the organisation would be unable to operate.
Or operate well, at any rate: hence the notion of a ‘viable service’. A key part of ‘to-be’ functional design is to identify support-relationships for services that the function does not contain within itself. Serious viability problems can occur if those support-links are either missing or poorly-maintained – the absence of explicit support for corrective-action and quality-management being classic examples that I’ve seen in all too many business-environments.

Document any identified gaps, and resolve these gaps wherever practicable.

In many cases, the required support-services or links will in fact exist, but are not recognised as such: they may be implied in what is actually done in practice, but no explicit formal procedures exist to call for them. In others, the links might be subsumed under a generic task-heading of ‘responsibilities of line-management’ or suchlike – but there may be no explicit means to ensure that the work required within the respective service-interrelationship is actually done.

And in some cases, the links may be erratic, or entirely absent, often with results whose symptoms are all too obvious, but whose cause is not. High-pressure production environments, for example, are notorious for losing track of any future-focussed business-purpose, or of failing to understand the crucial quality-management distinction between ‘correction’ – “fix it up to get the job out of the door” – versus ‘corrective action’ – “make sure it doesn’t happen again”. All of these concerns – most often from inadequate integration of pervasive-services – need to be resolved before the enterprise can become fully viable, and fully alive.

**Function Model examples**

Two examples of completed function-models follow.

The first example is described below as for a ‘plastics factory’ – actually adapted from the function-model for one division of an organisation in another industry, de-identified and re-captioned to preserve commercial confidentiality. This shows the relatively simple case of an enterprise with a single straight-through value-chain,
from customer-order – in most cases received via another division of the organisation, hence only minimally addressed here – through to transfer to fulfilment-processes, which were again handled by another division.

**Functions: manufacturing division of (fictional) plastics firm**

The organisation had several distributed work-sites, hence inter-site transport was an important factor. The zone shown above the main value-chain provides the policy and strategy management-services and, to some extent, the pervasive-
services; the zone below the ‘transport’ function shows typical support-services provided by central management.

The second example is adapted from an early version of a model developed for a state-government department in the social-services sector. This is considerably more complex than the previous example, because it has separate value-streams for each of its five distinct ‘services to citizens’, plus an additional value-stream for its formal relations with government and its many other societal stakeholders.

|----------|--------------------|-----------------|-----------------|--------------------|----------------------------------|

*Functions: government department in social-services sector*
All value-streams share many of the same support-services, which makes it inappropriate to use a simple layout of straight-through value-chains, as in the previous example. Instead, a two-tier Viable System Model layout is used, with the main delivery-services shown above as one of the tiers, and the support-services shown below as the other.

The same vertical axis is used in both tiers, other than that the ‘Contact’ (customer) row is not used in the second tier, because the customer-facing delivery-services are implicitly ‘customers’ for the support-services. The categories in the vertical axis are:

- **channel** – delivery-services (customer-facing only)
- **deliver** – delivery-services
- **fund to deliver** – delivery-services
- **educate and train** – delivery-services
- **coordinate** – ‘direction’ management-services, ‘run the business’ coordination-services
- **monitor and improve** – pervasive services, ‘change the business’ coordination-services
- **research, prepare, plan** – ‘policy’ and ‘strategy’ management-services, ‘develop the business’ coordination-services

Some of the apparent gaps are intentional – for example, some value-streams do not deliver community-education services, and others do not provide internal training. Other gaps, though, are genuine, indicating potential or actual problems for the enterprise.
Resources

Function-model examples: see Tom Graves, ‘The Viable Services Model: Service quality, service interdependence and service completeness’ in Jan van Bon [ed.], IT Service Management: Global best practices (itSMF/Van Haren, 2008)

Requirements-gathering (as an analogue for gathering information on business-functions): see Suzanne Robertson and James Robertson, Mastering the Requirements Process (Addison-Wesley, 1999)


PRACTICE – THE KNOWLEDGE OF SERVICES

Summary

With an appropriate Function Model in place, we can cross-link other concerns onto that standard reference-map: these might include information, information-flows, costings, projects, assets, applications, business risks, and much else besides. Such cross-references between the domains create new options for whole-of-enterprise optimisation.

Details

Services and business-systems
[[see published book for further details]]

Application themes: identifying business-systems
[[see published book for further details]]

Application themes: identifying single-source-of-truth
[[see published book for further details]]

The value of services
[[see published book for further details]]
Application themes: Function-Model overlays
   [[see published book for further details]]

Verifying viability
   [[see published book for further details]]

Application themes: viable relations
   [[see published book for further details]]

Application themes: viable communications
   [[see published book for further details]]
PRACTICE – SERVICES IN ACTION

Summary

The practical purpose of a service is to get something done – it delivers a service of some kind to the customer or consumer of that service. Practical concerns that arise from this include the need to ask what the service does, how it does so, and how the various functions and services are coordinated into a seamless business-process that provides the final service that the customer receives.

Details

Actions and transactions
[[see published book for further details]]

Application themes: process-choreography – processes
[[see published book for further details]]

Application themes: process-choreography – services
[[see published book for further details]]

Breaking free from IT
[[see published book for further details]]

Application themes: challenging IT-centrism
[[see published book for further details]]
Application themes: consultancy as service

[see published book for further details]
PRACTICE – OPTIMISING SERVICES

Summary
From the customer’s perspective, the service ends once the results specified in the service-contract have been delivered. But for the service-provider, there’s a need for a final phase to review what has been done, and how well it has been done – and what else could be done to improve the overall effectiveness of the service-oriented enterprise.

Details

Audit and improvement
[[see published book for further details]]

Application themes: audit to improve
[[see published book for further details]]

Improvement and optimisation
[[see published book for further details]]

Application themes: lessons-learned
[[see published book for further details]]

Optimisation and engagement
[[see published book for further details]]
Application themes: architecture engagement

[see published book for further details]
APPENDIX: BUILDING A FUNCTION MODEL IN VISIO

Preliminaries

Unpack the template
[[see published book for further details]]

Application

Create a new function-model file
[[see published book for further details]]

Create the function-model
[[see published book for further details]]

Add a new page
[[see published book for further details]]

Create reports
[[see published book for further details]]

Allocate Business Systems
[[see published book for further details]]
This section summarises some of the key terms and acronyms used in the book.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appropriate</td>
<td>matching the intended overall purpose; a REAL/LEARN effectiveness-assessment theme associated with the aspirational dimension of the context</td>
</tr>
<tr>
<td>aspirational</td>
<td>aspirational and intentional aspects of work and the workplace, expressed in collective and individual identity and purpose, and in issues such as ethics, values and codes of conduct; also commitment-assets and aspirational capital such as organisational morale, health and fitness</td>
</tr>
<tr>
<td>conceptual</td>
<td>mental and conceptual aspects of work and the workspace: beliefs, attitudes, knowledge, procedures and process specifications; also knowledge-assets and intellectual capital</td>
</tr>
<tr>
<td>effective</td>
<td>‘on purpose’, producing the intended overall result with an optimised balance over the whole; requires broad generalist awareness of the whole, rather than the narrow focus required to create local efficiency, hence often contrasted with efficient; see also REAL/LEARN</td>
</tr>
<tr>
<td>efficient</td>
<td>‘doing more with less’, creating the maximum result with minimum use or wastage of resources in a specific activity or context; improved incrementally through active learning and related techniques for feedback and reflection, although major improvements usually require a change in paradigm; a REAL/LEARN theme associated with the conceptual dimension of the context</td>
</tr>
<tr>
<td>elegant</td>
<td>human dimensions of effectiveness, such as feelings, emotions and ergonomics, expressed in issues such as usability, simplicity and personal preference; a REAL/LEARN effectiveness-assessment theme associated with the relational dimension of the context</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>emergence</td>
<td>context within which cause-effect patterns can be identified only retrospectively, and in which analytic techniques are usually unreliable and misleading</td>
</tr>
<tr>
<td>enterprise</td>
<td>a systematic process to model and guide integration and optimisation of the entire enterprise</td>
</tr>
<tr>
<td>architecture</td>
<td></td>
</tr>
<tr>
<td>FEAF</td>
<td>acronym for Federal Enterprise Architecture Framework, a framework and methodology developed for enterprise architecture by the US government</td>
</tr>
<tr>
<td>goal</td>
<td>a specific objective to be achieved by a specified point in time; emphasis on the behavioural or physical dimension of purpose, contrasted with mission, role and vision</td>
</tr>
<tr>
<td>integration</td>
<td>contextual awareness of all the interactions between the physical, conceptual, relational and aspirational dimensions of work and the workspace, and the active process of linking them together into a unified whole</td>
</tr>
<tr>
<td>mission</td>
<td>a desired capability or state to be achieved, usually within a specified timeframe, and to be maintained indefinitely once achieved; emphasis on the emotional and, to a lesser extent, the conceptual dimensions of purpose, contrasted with goal, role and vision</td>
</tr>
<tr>
<td>narrative</td>
<td>personalised and often emotive expression or interpretation of knowledge, as history, anecdote or story; link-theme between mental dimension and relational dimension</td>
</tr>
<tr>
<td>optimisation</td>
<td>process of integration in which efficiency in different areas is traded-off and balanced for maximum effectiveness over the whole, between different layers and sub-contexts such as departments, business processes and business units</td>
</tr>
<tr>
<td>physical</td>
<td>physical aspects of work and the workspace: skills, competencies, physical processes, behaviours, actions; also tangible assets and work-environment</td>
</tr>
<tr>
<td>dimension</td>
<td></td>
</tr>
<tr>
<td>principle</td>
<td>a conceptual or aspirational commitment or model, the conceptual-dimension or aspirational-dimension equivalent of value</td>
</tr>
<tr>
<td>purpose</td>
<td>an expression of individual and/or collective identity - the aspirational theme of “who we are and what we stand for”; incorporates distinct dimensions of vision, role, mission and goal</td>
</tr>
<tr>
<td><strong>R⁵</strong></td>
<td>collective term for five complexity-science principles used with the <em>tetradian</em>, namely <em>recursion</em>, <em>rotation</em>, <em>reflexion</em>, <em>reciprocation</em> and <em>resonance</em>.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>REAL / LEARN</strong></td>
<td>acronym for four keywords to evaluate effectiveness: <em>reliable</em>, <em>efficient</em>, <em>appropriate</em>, <em>elegant</em>; the LEARN acronym includes <em>integration</em> in the evaluation-set</td>
</tr>
<tr>
<td><strong>reciprocation</strong></td>
<td>overall balance in transactions, especially <em>power</em>-transactions; reciprocal balance between entities may not be direct or immediate, and in many cases balance may only be achieved over time at a system-wide level, with energy-transfers occurring between <em>physical</em>, <em>conceptual</em>, <em>relational</em> and/or <em>aspirational</em> dimensions; an <em>R⁵</em> principle for assessment of <em>effectiveness</em> and relevance</td>
</tr>
<tr>
<td><strong>recursion</strong></td>
<td>patterns of relationship or interaction repeat or are ‘self-similar’ at different scales; permits simplification of otherwise complex processes; an <em>R⁵</em> principle for assessment of <em>effectiveness</em> and relevance</td>
</tr>
<tr>
<td><strong>reflexion</strong></td>
<td>corollary of <em>recursion</em>, in that the whole, or aspects of the whole, can be identified within the attributes and transactions of any part at any scale; an <em>R⁵</em> principle for assessment of <em>effectiveness</em> and relevance</td>
</tr>
<tr>
<td><strong>relational dimension</strong></td>
<td>relational and emotional aspects of the work context: feelings and <em>values</em>, internal relationships and interpersonal transactions, relationships with external stakeholders; also assets such as reputation and trust</td>
</tr>
<tr>
<td><strong>reliable</strong></td>
<td>high degree of certainty and predictability for a desired outcome; a <em>REAL / LEARN</em> effectiveness-assessment theme associated with the <em>physical dimension</em> of the context</td>
</tr>
<tr>
<td><strong>resonance</strong></td>
<td>concept of positive- or negative-feedback (increasing or damping) in a system; permits simplification of otherwise complex processes; an <em>R⁵</em> principle for assessment of <em>effectiveness</em> and relevance</td>
</tr>
<tr>
<td><strong>rotation</strong></td>
<td>systematic process of assessing a context from multiple perspectives; an <em>R⁵</em> principle for assessment of <em>effectiveness</em> and relevance</td>
</tr>
<tr>
<td><strong>role</strong></td>
<td>a declared focus or <em>strategic</em> position within the ‘world’ described by a <em>vision</em>; emphasis on the <em>conceptual</em> and, to a lesser extent, the <em>relational dimensions</em> of <em>purpose</em>, contrasted with <em>goal</em>, <em>mission</em> and <em>vision</em></td>
</tr>
<tr>
<td><strong>scenario</strong></td>
<td>an imagined future context, developed for the purpose of understanding both the present context and options for action in the future context</td>
</tr>
<tr>
<td><strong>strategy</strong></td>
<td>‘big picture’ view of an action-plan for an organisation to implement a <em>purpose</em>, usually emphasizing its <em>vision</em>, <em>role</em> and <em>mission</em> components; contrasted with the <em>tactics</em> required to execute the plan</td>
</tr>
<tr>
<td><strong>tactics</strong></td>
<td>detailed <em>missions</em>, <em>goals</em> and other step-by-step activities to execute a <em>strategy</em>, or some segment of an overall strategy</td>
</tr>
<tr>
<td><strong>tetradian</strong></td>
<td>depiction of the <em>physical</em>, <em>conceptual</em>, <em>relational</em> and <em>aspirational</em> dimensions of a context as four axes in a tetrahedral relationship, usually also showing the respective link-themes as the edges between the vertices of the tetrahedron</td>
</tr>
<tr>
<td><strong>TOGAF</strong></td>
<td>acronym for The Open Group Architecture Framework, an IT-oriented framework and methodology for <em>enterprise architecture</em> developed collectively by members of the Open Group consortium</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>an emotional commitment</td>
</tr>
<tr>
<td><strong>vision</strong></td>
<td>description of a desired ‘world’, always far greater than any individual or organization; described in the present tense, yet is never ‘achieved’; emphasis on the <em>aspirational dimension</em> of <em>purpose</em>, contrasted with <em>goal</em>, <em>mission</em> and <em>role</em>; also link-theme between <em>aspirational dimension</em> and <em>relational dimension</em></td>
</tr>
<tr>
<td><strong>visioning</strong></td>
<td>generic term for the process of identifying, developing and documenting <em>vision</em> and <em>values</em>, leading towards <em>strategy</em> and <em>tactics</em></td>
</tr>
<tr>
<td><strong>Zachman framework</strong></td>
<td>a systematic structure for categorisation of models within an IT-oriented <em>enterprise architecture</em>, developed by John Zachman</td>
</tr>
</tbody>
</table>